Sparse Contrastive PCA and its Feasibility for Metabolomics Data

Sarah Mensah

Department of Mathematical Sciences
Montana State University

May 10, 2024

A writing project submitted in partial fulfillment
of the requirements for the degree

Master of Science in Statistics

APPROVAL

of a writing project submitted by

Sarah Mensah

This writing project has been read by the writing project advisor and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the Statistics Faculty.

Date	Mark Greenwood Writing Project Advisor
Date	Katharine Banner Writing Projects Coordinator

Abstract

Untargeted metabolomics studies generate high-dimensional data that are challenging to analyze and interpret. Principal Component Analysis (PCA) is a popular dimensionality reduction tool, but its components can be difficult to interpret due to the linear combination of all the original variables. Sparse PCA addresses the issue of interpretability by imposing sparsity constraints on the loadings, resulting in components that comprise only a few key variables. Contrastive PCA, on the other hand, helps to remove any noise in the covariance structure by contrasting the original (target) data with background (control) data. In this paper, I explore a specific implementation of sparse PCA called sparse contrastive PCA (scPCA). scPCA achieves a dual objective of extracting important signals from data and making the principal components more interpretable. The method is applied to three datasets to uncover its strengths, limitations, and feasibility for metabolomics data.

Contents

1 Introduction

Metabolomics is the study of concentrations of different small molecules (known as metabolites) for the sample from each subject. It can provide insights into the function of various systems but also can generate a large number of metabolites to study. Targeted metabolomics involves collecting and analyzing a small or moderate number of metabolites that are thought to be of interest. Untargeted metabolomics studies, where all detectable metabolites are explored, often generate high dimensional data. These studies may generate thousands of metabolites (variables) for tens or hundreds of subjects. The high-dimensional nature of the data generated poses challenges for analysis and interpretation.

One way to address this challenge is to employ dimensionality reduction techniques to project the data into a lower dimensional space and understand both the variables and the subjects in that lower dimensional space. Principal Component Analysis (PCA) is commonly used for dimensionality reduction; however, the interpretability of the principal components (PCs) can be limited due to the linear combination of all the original variables. Additionally, the covariance structure on which eigenvalue decomposition is performed may contain unwanted variation, and as a result, the noise may mask the important signals in the data. To address these issues of interpretability and unwanted variation, researchers have proposed various variants of PCA.

This paper explores a variant of PCA called sparse contrastive PCA (scPCA) recently proposed by Boileau et al. (2020a). Sparse contrastive PCA is a combination of sparse PCA as proposed by Zou et al. (2006) and contrastive PCA as proposed by Abid et al. (2018). scPCA simultaneously removes any technical noise in the covariance structure before eigenvalue decomposition while imposing sparsity constraints on the PC loadings. This method achieves a dual objective of extracting important signals from the data and increasing the interpretability of the PCs. My goal is to explore the scPCA's strengths, limitations, and feasibility for reducing the dimensionality of high dimensional metabolomics data to make them more interpretable and easier for analysis.

Section 2 presents a background on sparse contrastive PCA. Section 3 describes the algorithm and software implementation for sparse contrastive PCA. In Section 4, we show results of sparse contrastive PCA applied to different datasets. Section 5 discusses the method, its application, and the implications of this research and future work.

2 Background

2.1 PCA

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction and visualizing multivariate data. The core concept of PCA is to transform a set of potentially correlated variables into a new set of uncorrelated variables, known as principal components (PCs). The orthogonality of the PCs is a property shared by all other variants of PCA discussed in this paper. These PCs are generated to sequentially capture the maximum variance in the data while minimizing loss of information. The PCs can subsequently replace the original data for further exploration and analysis purposes.

PCA is performed on the covariance matrix, denoted as C, of a data set (n observations on P variables) to create the principal components. The eigenvectors of C (also known as PC loadings) represent the directions in which the data vary the most, while the eigenvalues indicate the amount of variance explained by each principal component. The transformation to the principal components can be visualized as a rotation of the original variables to align with the directions of maximum variance in the data. The first PC captures the most variance, followed by the second PC, and so on. PCA is typically performed on correlation matrices, R, to provide even weight to variables with different units and/or different variances, but the correlation matrix is just a special case of the covariance matrix.

PCA can be performed using either the Jordan decomposition or a singular value decomposition (SVD), as described by Hardle et al. (2003). The Jordan decomposition provides a way to represent a symmetric matrix in terms of its eigenvalues and eigenvectors.

In contrast, SVD extends the Jordan decomposition to non-square matrices. The Jordan decomposition requires at least as many observations as variables, while SVD can be applied even when the number of variables exceeds the number of observations.

The covariance matrix, C, can be reconstructed using a subset of eigenvectors and eigenvalues. This can be achieved by multiplying the matrix of eigenvectors by the eigenvalues and then by the transpose of the eigenvectors. The reconstruction of the original covariance matrix is exact when all eigenvectors and eigenvalues are used and it is approximate when only a subset are used.

Eigenvalues are useful to understand the amount of variance captured by each principal component. They provide a quantitative measure of the importance of each PC in explaining the variation in the data. Typically, higher eigenvalues correspond to PCs that capture more variance in the original data. By examining the magnitude of the eigenvalues, researchers are able to prioritize which PCs to retain based on their contribution to the overall variance.

Everitt and Hothorn (2011) discussed several rules of thumb for deciding how many principal components to retain. One common approach is to retain enough components to explain a specific percentage of the total variation of the original variables, typically aiming for between 70% and 90%. However, the choice of this percentage can vary depending on the research questions. Another guideline is to exclude principal components whose eigenvalues are less than one, especially when extracting components from the correlation matrix. Additionally, researchers may use scree plots, which plot the eigenvalues against the component number. The number of components to retain is often chosen at the "elbow" of the curve, that is, where the slope changes. A change in slope indicates a point where adding more components does not lead to a substantial increase in the explained variance.

After determining the number of principal components needed to represent our data, we typically calculate scores for each individual on these components. These scores, known as PC scores, reflect the values of the i^{th} principal components for the n observations in our samples. PC scores play a key role in constructing biplots.

A biplot is a graphical representation that combines PC scores and variables, offering a visual means to understand the relationships between variables and observations. In a biplot, observations are represented as points, and variables are represented as vectors originating from the plot's origin. The length and direction of the vectors indicate the variable's contribution to the PC scores and their variance. Variables with longer vectors contribute more to the respective PCs. The correlation between the original variables is shown by the angle between the vectors. A cosine of the angle approximates the correlation, with angles closer to 90 or 270 degrees indicating less related variables. The cutpoint of a perpendicular from a specific point to a variable line approximates the value of that observation on the variable. If the cutpoint falls on the origin, the observation's value is approximately the variable's average. Points far from the origin in the direction of the variable line indicate high values, while points far that are far from the origin and project into the negative of the vector represent values far below average. Those suggestions for interpretation do not apply when vectors are very short in the displayed PCs and also may not be as correct when the PCs being displayed explain very low percentages of the total variation in the data.

2.1.1 PCA on a Simulated Data

In order to explore some of the features of PCA and extensions, PCA is applied to simulated dataset consisting of $p = 30$ continuous variables and $n = 400$ observations with four groups. The data are available in the scPCA package (Boileau et al., 2020b)

	PC	Eigenvalue	Proportion_of_Variance
1	1	123.931	0.111
2	2	121.823	0.109

Table 1: Eigenvalues and proportion of variance explained by the first two PCs for PCA on the simulated data.

	PC1	PC2
V1	-0.112	0.117
V2	0.197	-0.426
V3	0.217	-0.297
V4	0.064	-0.274
V5	-0.329	0.074
V6	0.152	-0.104
V7	-0.491	-0.405
V8	0.005	-0.345
V9	0.720	0.031
V10	-0.010	0.581
V11	0.027	0.020
V12	0.031	0.021
V13	0.024	0.020
V14	0.031	0.020
V15	0.020	0.015
V16	0.031	0.018
V17	0.032	0.019
V18	0.024	0.019
V19	0.030	0.010
V20	0.019	0.017
V21	-0.001	0.004
V22	-0.001	-0.008
V23	-0.002	-0.003
V24	-0.006	-0.001
V25	-0.006	-0.010
V26	0.006	0.002
V27	0.000	-0.002
V28	-0.005	-0.002
V29	-0.007	-0.005
V30	0.003	-0.014

Table 2: Loadings of the first two PCs from PCA on the simulated data.

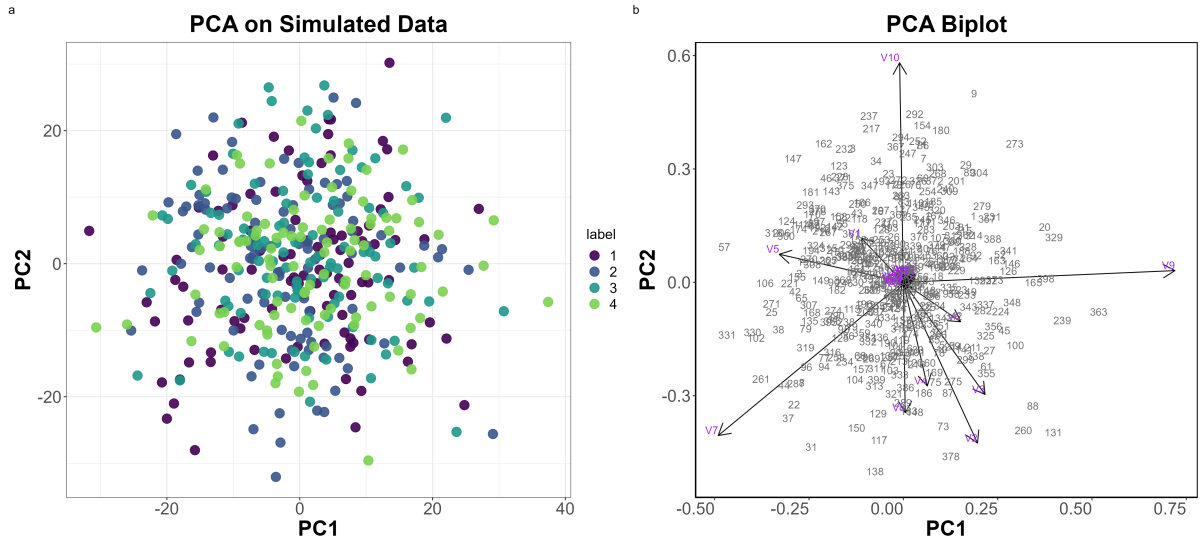

Figure 1: **a**) Plot of first two principal components of simulated data (covariance version adapted from BHDS, 2023). **b**) Biplot of principal components for simulated data.

From Table 1, the first and second principal components account for approximately 11.1% and 10.9% of the variation in the data, respectively. Combined, these two components explain approximately 22% of the total variation in the data. Table 2 shows the loadings of the first two PCs. The first two principal components have no non-zero loadings. Figure 1a illustrates the first two dimensions resulting from PCA applied to the simulated data.The primary aim of using PCA for dimensionality reduction is to identify important data patterns within the first few dimensions. In this case, the data contain distinct groups, which are the signals of interest. However, these groups are not separated in the first two dimensions. The biplot in Figure 1b shows that there is a correlation between variables 8, 4, 2, and 3, and all these variables contribute to PC2. Variables 9 and 10 are uncorrelated and contribute primarily to PC1 and PC2, respectively. Approximately ten variables contribute to the first two PCs, while the remaining 20 variables may contribute to the third or subsequent PCs. Note that this analysis is revisited in section 4.

2.2 Sparse PCA

PCA produces principal components that are a linear combination of all the original variables, and this poses interpretation challenges, especially for high dimensional datasets that have hundreds or thousands of variables. Sparse PCA (sPCA) addresses this issue by imposing sparsity constraints on the loading so that the PCs produced will only be a linear combination of a few key variables. This makes the PCs more interpretable. There are several implementations of sparse PCA proposed by different researchers, but the one we are focusing on is the sPCA proposed by Zou et al. (2006).

Sparse PCA is built on the idea that PCA can be formulated as a regression-type optimization problem with a quadratic penalty (via the elasticnet), as described by Zou et al. (2006). The sparse loadings are found as the solution to the elastic net problem for the k PCs, $j = 1, 2, \ldots, k$, of

$$\max_{\beta_j} \boldsymbol{\beta}_j^T C_x \boldsymbol{\beta}_j - \lambda_0 \left\|\boldsymbol{\beta}_j\right\|_2^2 - \lambda_{1,j} \left\|\boldsymbol{\beta}_j\right\|_1$$

where β_j is a vector of sparse loadings and λ_0 and $\lambda_{1,j}$ are the tuning parameters. The tuning parameters provide a compromise between variance and sparsity. When λ_0 and $\lambda_{1,j}$ are equal to zero, sPCA reduces to PCA. A method described in Boileau et al. (2020a) is used to find the optimal values of the tuning parameters for this and the other related methods, although the sPCA tuning has been proposed in a variety ways across different related approaches.

2.2.1 Sparse PCA on Simulated Data

sPCA is applied to the simulated dataset discussed in Section 2.1.1. The results in Table 3 showed that the first and second principal components explain about 11.1% and 10.9% of the variation in the data, respectively, totaling about 22% together. This is similar to the percentage of variation explained by the first two PCs in standard PCA. However, from Table 4, PC1 has 20 non-zero loadings, and PC2 has 18 non-zero loadings. Sparse PCA shrank the contribution of some variables (10 in PC1 and 12 in PC2) to zero, making the PCs easier to interpret. Despite this, Figure 2a doesn't show the groups separating, indicating that

the variation of interest isn't captured in the first two PCs. The biplot in Figure 2b has a similar pattern as Figure 1, but with fewer variables, particularly in the third and subsequent dimensions.

	PC	Proportion_of_Variance
1	1	0.111
2	2	0.109

Table 3: Proportion of variance explained by sPCA on simulated data.

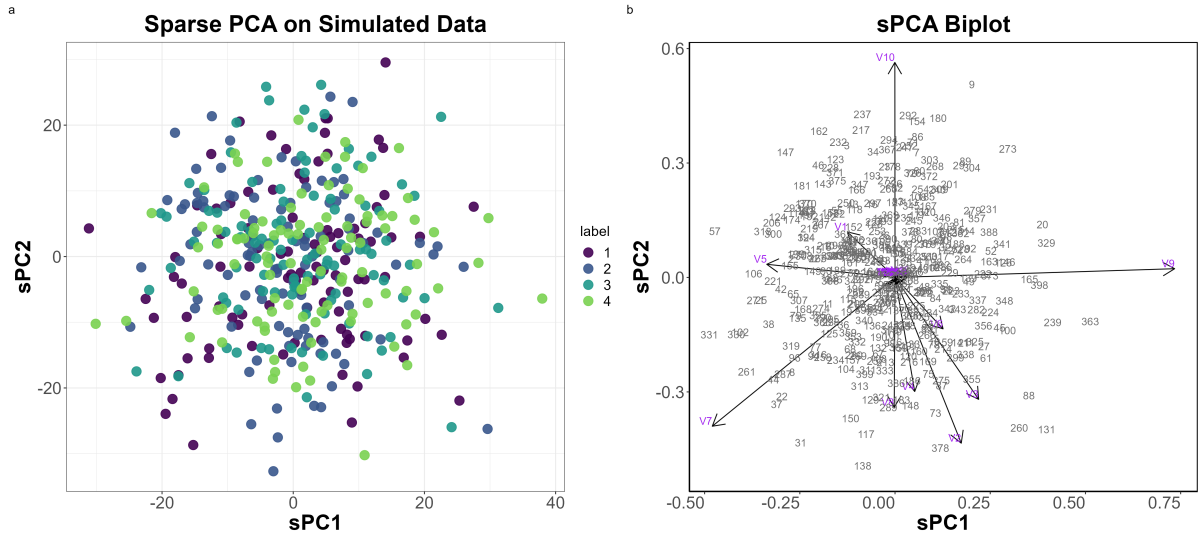

Figure 2: **a**) Plot of first two principal components of simulated data (covariance version adapted from BHDS, 2023). **b**) Biplot for sparse principal components for simulated data.

	PC1	PC2
V1	-0.112	0.117
V2	0.197	-0.426
V3	0.217	-0.297
V4	0.063	-0.274
V5	-0.329	0.074
V6	0.152	-0.104
V7	-0.491	-0.405
V8	0.005	-0.345
V9	0.720	0.031
V10	-0.010	0.581
V11	0.026	0.020
V12	0.031	0.020
V13	0.023	0.019
V14	0.031	0.020
V15	0.019	0.014
V16	0.031	0.018
V17	0.033	0.019
V18	0.023	0.018
V19	0.029	0.010
V20	0.019	0.017
V21	0.000	0.000
V22	0.000	-0.007
V23	-0.000	-0.001
V24	-0.004	0.000
V25	-0.002	-0.008
V26	0.000	0.000
V27	0.000	0.000
V28	-0.003	-0.002
V29	-0.004	-0.002
V30	0.000	-0.012

Table 4: Loadings of the first two PCs from sPCA on the simulated data.

2.3 Contrastive PCA

When the covariance structure on which eigenvalue decomposition is performed contains technical noise, traditional PCA may not be able to capture the important signals in the first few dimensions. Contrastive PCA (cPCA) helps to remove any noise in the covariance structure by contrasting the original data (target data) with background (control) data. The

background data must have the unwanted variation in the covariance structure that we want to remove. Noise can only be removed from the target data if it is also present in the background data. The aim of cPCA is to identify the important patterns in the data after removing the underlying noise information.

Let C_X and C_Y denote the empirical covariance matrices of the target and background data, respectively. cPCA is performed by computing the eigenvalue decomposition of C_γ where

$$C_\gamma = C_X - \gamma C_Y,$$

and γ is the contrastive parameter which quantifies the tradeoff between each variable's variances in the target and background datasets. When $\gamma = 0$, cPCA reduces to PCA.

The constrastive covariance matrix C_γ is generally not positive-semidefinite. Instead, an approximated version, $\tilde{C}_\gamma$ which is positive-semidefinite is used. To obtain $\tilde{C}_\gamma$, Abid et al. (2018) suggests that the negative eigenvalues in the diagonal matrix in the eigen decomposition of $\tilde{C}_\gamma$ are replaced by zeros.

$$C_\gamma = V \Lambda V^T$$

$$\tilde{C}_\gamma = V D V^T$$

$$\text{where } D_{ii} = \begin{cases} \Lambda_{ii}, & if\ \Lambda_{ii} > 0 \\ 0, & otherwise \end{cases} \text{,for } i = 1, \cdots, p$$

2.3.1 Contrastive PCA on Simulated Data

The cPCA algorithm requires two datasets: the target and background data. We use the data discussed in Section 2.1.1 as the target data and introduce the background data, which is also available in the scPCA package. The background data have $n = 400$ observations and $p = 30$ variables.

cPCA is applied to these simulated data. To fit contrastive PCA using the scPCA function, one sets the penalties argument to 0. Results for cPCA are shown in Table 5, Table 6, and

Figure 4.

	PC	Eigenvalue	Proportion_of_Variance
1	1	8.916	0.297
2	2	7.154	0.238

Table 5: Eigenvalues and proportion of variance explained by cPCA on the simulated data.

	V1	V2
1	-0.013	0.002
2	-0.011	-0.001
3	-0.037	-0.009
4	-0.024	-0.006
5	-0.002	-0.007
6	-0.003	-0.010
7	0.015	0.007
8	0.032	-0.008
9	-0.035	0.006
10	0.013	-0.000
11	-0.307	-0.016
12	-0.305	-0.000
13	-0.302	-0.006
14	-0.265	0.007
15	-0.270	0.007
16	-0.294	-0.014
17	-0.361	-0.008
18	-0.297	-0.015
19	-0.394	0.010
20	-0.325	-0.010
21	-0.001	-0.351
22	-0.052	-0.309
23	0.027	-0.317
24	0.033	-0.319
25	0.034	-0.302
26	0.006	-0.302
27	-0.031	-0.310
28	0.022	-0.339
29	0.003	-0.284
30	0.003	-0.322

Table 6: Loadings of the first two PCs from cPCA on the simulated data.

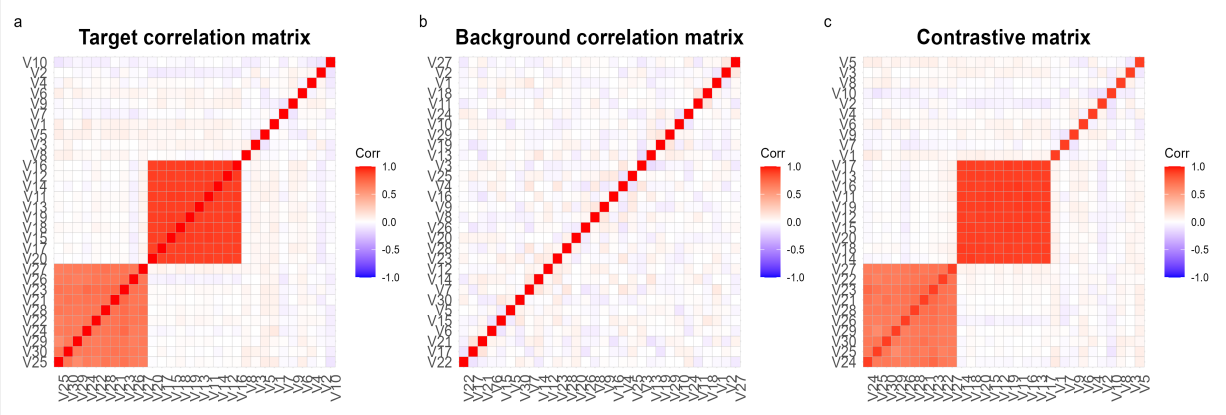

Figure 3: Plots of correlation matrices of target data, background data, and the contrastive matrix.

Figure 3 shows a comparison of the correlation matrices of the target and background data and the constrastive matrix. We are presenting a plot of the correlation matrices because they are easier to interpret than the covariance matrices, as the variables are on the same scale. The target correlation matrix contains both the variation of interest and unwanted variation in the original data, while the background correlation matrix captures the variation in the target data that we aim to remove. By taking the contrast between these matrices using a contrastive parameter of 5.54 and correcting any negative eigenvalues, we obtain the contrastive matrix. The contrastive matrix is the final matrix on which the eigenvalue decomposition is performed.

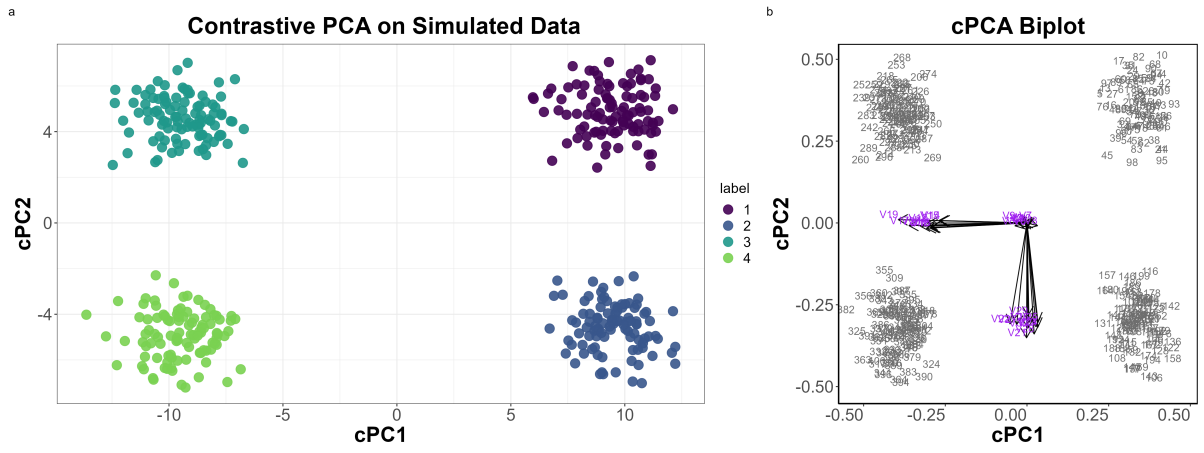

Figure 4: **a**) Plot of first two principal components (Adapted from BHDS, 2023). **b**) Biplot for contrastive principal components.

Figure 4a displays a plot of the first two principal components. After incorporating background data, cPCA can now distinguish the target data's signals of interest, leading to a clear separation of the four groups, which is a positive outcome. The first and second principal components explain about 29.7% and 23.8% (as shown in Table 5) of the variation, respectively. Together, these two PCs account for a total of 53.6% of the variation in the data, exceeding the variability explained by both PCA and sPCA. However, because no sparsity was imposed on the loadings, each principal component is a linear combination of all the original variables (as shown in Table 6), posing challenges for interpretation. The biplot in Figure 4b indicates that most variables contribute mainly to either the first or second PC, with a few contributing to the third and subsequent PCs. Additionally, the variables contributing to PC1 are close to orthogonal to those contributing to PC2.

3 Method

3.1 Sparse Contrastive PCA

Sparse contrastive PCA combines contrastive PCA to remove technical noise with sparse PCA to reduce the number of explicitly used variables in each PC, which makes the PCs more interpretable. The sparse contrastive PCA (scPCA) procedure applies sPCA with minimal modifications to a pair of target and background datasets' contrastive covariance matrix C_γ (Boileau et al., 2020a).

The elastic net solution for the j^{th} loading vector, $j = 1, \cdots, k$, is

$$\max_{\beta_j} \boldsymbol{\beta}_j^T C_\gamma \boldsymbol{\beta}_j - \lambda_0 \left\|\boldsymbol{\beta}_j\right\|_2^2 - \lambda_{1,j} \left\|\boldsymbol{\beta}_j\right\|_1,$$

where β_j is a vector of sparse loadings.

3.2 Tuning Parameters

The tuning parameters in the scPCA algorithm are the contrastive parameter γ and the l_1 penalty parameter λ_1. Boileau et al. (2020a) proposed to select γ and λ_1 by k-means clustering the observations of the target dataset based on their first k scPCs and choosing the combination of γ and λ_1 that produces the "strongest" or "best" cluster assignments. This selection is done by maximizing the average silhouette width over clustering in the reduced dimension of the data with k-means as the chosen clustering algoritm.

K-means, as discussed by Charrad et al. (2014), is an iterative clustering method that aims to minimize the sum of squares within each cluster. It starts by guessing initial cluster centers and then assigns each observation to the closest cluster center. The cluster centers are then updated, and this process repeats until convergence. Often, other clustering algorithms are used initially to determine the starting points for the cluster centers. It is important to note that the random initialization often impacts the final cluster allocations in k-means, so

results for the methods tuned in this fashion could change with different random number seeds. Running k-means multiple times and selecting the best solution for each number of clusters can be used to reduce the variability in the results, but it also then dramatically increases the computing effort to optimize these methods.

The silhouette, as described by Rousseeuw (1987), is a graphical tool used in cluster analysis to represent clusters based on the comparison of how close and distinct each cluster is from the others. It shows which objects are well within their cluster and which ones are in between clusters or might even be closer to other clusters than the cluster they are currently being incorporated into. The construction of silhouettes requires two components: the partition generated by applying a clustering technique and the collection of all pairwise distances between objects.

Specifically, take any object i in the dataset, and denote by A the cluster to which it has been assigned. Then compute

$$a(i) = \text{ average dissimilarity of } i \text{ to all other objects of } A.$$

Assuming that the number of clusters is more than one, consider any cluster C which is different from A, and compute $d(i, C)$ which is the average dissimilarity of i to all objects of C. Then let

$$b(i) = \min_{C \neq A} d(i, C).$$

With these, the $s(i)$, called the silhouette, is calculated as

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))},$$

where $-1 \leq s(i) \leq 1$ for each object i.

When $s(i)$ is close to 1, it means the object is "well-clustere" with $a(i)$ much smaller than $b(i)$. When $s(i)$ is around zero, it suggests i is equidistant from both its current cluster and the one it is closest to that isn't the current cluster. A value of $s(i)$ close to -1 indicates that

the object is likely misclassified as $a(i)$ is much larger than $b(i)$, and it is actually closer to a cluster that it was not placed in by the clustering algorithm.

The individual silhouette values are averaged together to create overall measures for the cluster quality for a given cluster (when averaged just over the members of a single cluster) or when averaged over all the observations, the average silhouette is a measure of the complete cluster solution. The original authors do suggest removing singletons and observations that always have large negative silhouette values from the calculation. Due to average silhouette being a mean, outliers have major impacts on it. In general, larger average silhouette values suggest better cluster solutions as the observations are fitting into their clusters well. But it is important to note that this measure is unable to check for a single cluster versus two or more clusters and in Tran (2019), the average silhouette width often optimized at two clusters when simulations were performed with only one group present.

3.3 Software Implementation

According to Boileau et al. (2020a), the number of observations in both the target and background data are not limiting factors in terms of computational complexity. However, scPCA's computational efficiency is most affected by the number of features and the size of the hyperparameter grid.

Implementation of scPCA is available in the scPCA package Boileau et al. (2020b) in R, and the package has been released as part of the Bioconductor project.

3.4 Sparse Contrastive PCA on Simulated Data

scPCA is applied to the covariance matrices in the simulated dataset discussed in Section 2.3.1. Results for scPCA are shown in Tables 7 and 8 and Figure 5.

	PC	Eigenvalue	Proportion_of_Variance
1	1	7.402	0.247
2	2	5.776	0.193

Table 7: Eigenvalues and proportion of variance explained by scPCA on the simulated data.

	V1	V2
1	0.000	0.000
2	0.000	0.000
3	0.000	0.000
4	0.000	0.000
5	0.000	0.000
6	0.000	0.000
7	0.000	0.000
8	0.000	0.000
9	0.000	0.000
10	0.000	0.000
11	-0.037	0.000
12	0.000	0.000
13	0.000	0.000
14	0.000	0.000
15	0.000	0.000
16	0.000	0.000
17	-0.964	0.000
18	0.000	0.000
19	-1.453	0.000
20	-0.353	0.000
21	0.000	1.027
22	0.000	0.076
23	0.000	0.339
24	0.000	0.292
25	0.000	0.000
26	0.000	0.000
27	0.000	0.000
28	0.000	0.824
29	0.000	0.000
30	0.000	0.147

Table 8: Loadings of the first two PCs from scPCA on the simulated data

After performing eigenvalue decomposition on the contrastive covariance matrix while

imposing sparsity constraints on the PC loadings, scPCA returned a value of 2.15 for the contrastive parameter, γ, and a value of 0.01 for the l_1 penalty, λ_1. The results in Table 7 indicate that the first two principal components explain approximately 26% and 19.2% of the variation, totaling 45.2% of the variation in the original data. This total variation explained by scPCA is higher than the total variation explained by both PCA and sPCA in the first two dimensions. However, compared to cPCA, the first two PCs of cPCA explain about 8% more of the total variation than scPCA. From Table 8, PC1 has seven non-zero loadings and PC2 has six non-zero loadings, making these loadings easily interpretable compared to cPCA. This trade-off between variability and interpretability suggests that scPCA sacrifices some variability for enhanced interpretability.

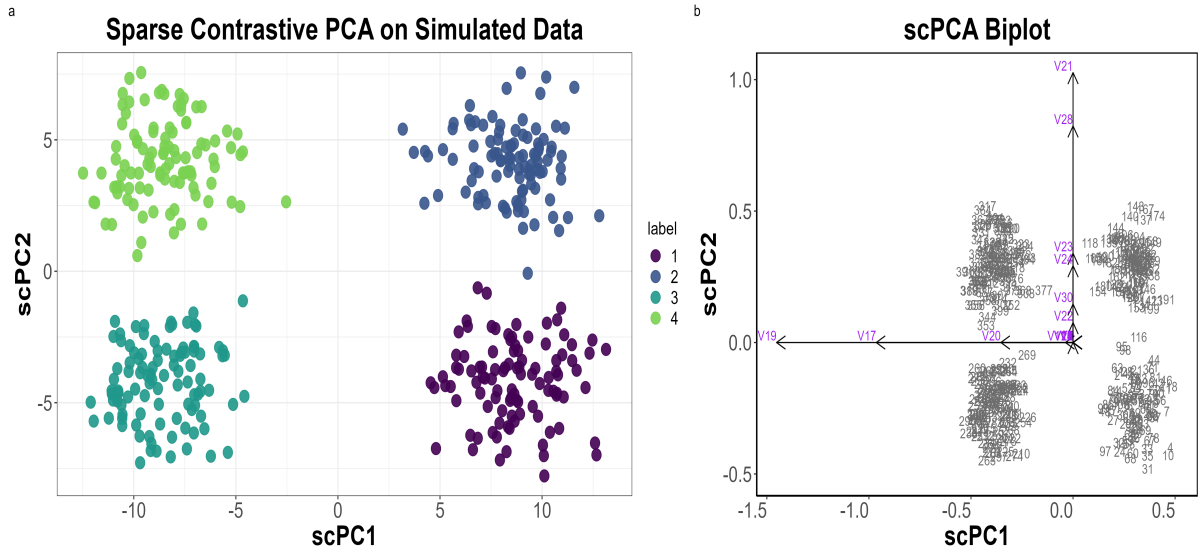

Figure 5: **a**) Plot of first two principal components of simulated data (adapted from BHDS, 2023). **b**) Biplot for sparse contrastive principal components.

scPCA is also able to separate the distinct groups in the data in the first two dimensions, as shown in Figure 5a, similar to cPCA. The biplot in Figure 5b indicates that only a few variables contribute to PCs 1 and 2, as discussed.

4 Results

4.1 Toy Example: Alternate Version

In this section, we discuss a second version of the results on the simulated dataset discussed in Section 2.1.1. After exploring the calculation of the contrastive covariance matrix and its scaling before eigenvalue decomposition, it became evident that the contrastive matrix is scaled in this process. Consequently, all results presented for cPCA and scPCA in the previous section are outcomes of eigenvalue decomposition performed on a scaled contrastive matrix. In contrast, both PCA and sPCA were conducted on a non-scaled covariance matrix, resulting in an unfair comparison. To address this, results for PCA and sPCA on a scaled covariance matrix are provided below.

	PC	Eigenvalue	Proportion_of_Variance
1	1	9.128	0.304
2	2	7.186	0.240

Table 9: Eigenvalues and proportion of variance explained by PCA and sPCA on the correlation matrix of the simulated data.

Tables 9, 10 and 11demonstrate that the proportion of variation explained by the first two principal components increased from 22% to 54.4% for both PCA and sPCA. While PCA continued to show no non-zero loadings in the first two components, sPCA exhibited a reduction in non-zero loadings from 20 to 10 for PC1 and from 18 to 10 for PC2. Additionally, Figure 6 illustrates that PCA and sPCA effectively capture the variation in the data in the first two dimensions, akin to cPCA and scPCA, with clear separation among the four groups.

	PC1	PC2
V1	-0.022	0.027
V2	0.026	-0.005
V3	-0.018	0.006
V4	-0.002	0.002
V5	-0.014	0.032
V6	-0.018	0.022
V7	0.005	-0.019
V8	0.003	0.019
V9	-0.017	-0.009
V10	0.008	-0.029
V11	-0.316	-0.009
V12	-0.315	-0.009
V13	-0.315	-0.009
V14	-0.316	-0.019
V15	-0.316	-0.016
V16	-0.317	-0.014
V17	-0.314	-0.011
V18	-0.315	-0.005
V19	-0.316	-0.014
V20	-0.314	-0.014
V21	-0.011	0.322
V22	-0.019	0.314
V23	-0.017	0.328
V24	-0.003	0.316
V25	-0.012	0.307
V26	0.009	0.314
V27	-0.028	0.313
V28	-0.013	0.320
V29	-0.012	0.306
V30	-0.009	0.313

Table 10: Loadings of the first two PCs from PCA on the correlation matrix of the simulated data.

	PC1	PC2
V1	-0.006	0.011
V2	0.009	0.000
V3	-0.002	0.000
V4	0.000	0.000
V5	0.000	0.014
V6	-0.001	0.004
V7	0.000	-0.001
V8	0.000	0.003
V9	-0.002	0.000
V10	0.000	-0.014
V11	-0.315	0.000
V12	-0.315	0.000
V13	-0.312	0.000
V14	-0.323	0.000
V15	-0.311	0.000
V16	-0.319	0.000
V17	-0.320	0.000
V18	-0.317	0.000
V19	-0.319	0.000
V20	-0.311	0.000
V21	0.000	0.324
V22	0.000	0.310
V23	0.000	0.329
V24	0.000	0.321
V25	0.000	0.312
V26	0.000	0.314
V27	0.000	0.312
V28	0.000	0.323
V29	0.000	0.304
V30	0.000	0.311

Table 11: Loadings of the first two PCs from sPCA on the correlation matrix of the simulated data

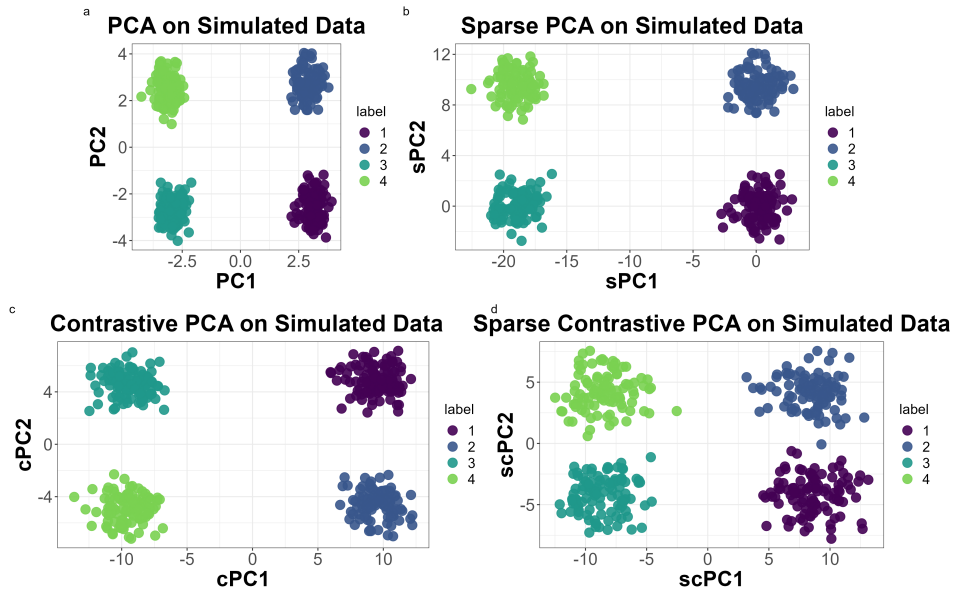

Figure 6: Plot of first two principal components for correlation matrix of simulated data (adapted from BHDS, 2023).

4.2 RNA Sequence Data

scPCA was tested on scRNA-seq data from cryopreserved bone marrow mononuclear cell (BMMC) samples as discussed by Boileau et al. (2020a). Samples were collected from an acute myeloid leukemia (AML) patient before and after undergoing stem cell transplant treatment. The target data consist of 4501 cells and 1000 genes (variables) from the AML patient. The background (control) data consist of $n = 4457$ cells and $p = 1000$ genes from two healthy patients.

Figure 7 illustrates that PCA fails to capture important biological signals in the first two dimensions, unable to differentiate between pre- and post-treatment statuses. In contrast, both cPCA and scPCA effectively capture these signals by displaying distinguishable clusters for pre- and post-transplant cells. scPCA shows a denser grouping of pre-transplant cells, while cPCA shows a denser grouping of post-transplant cells.

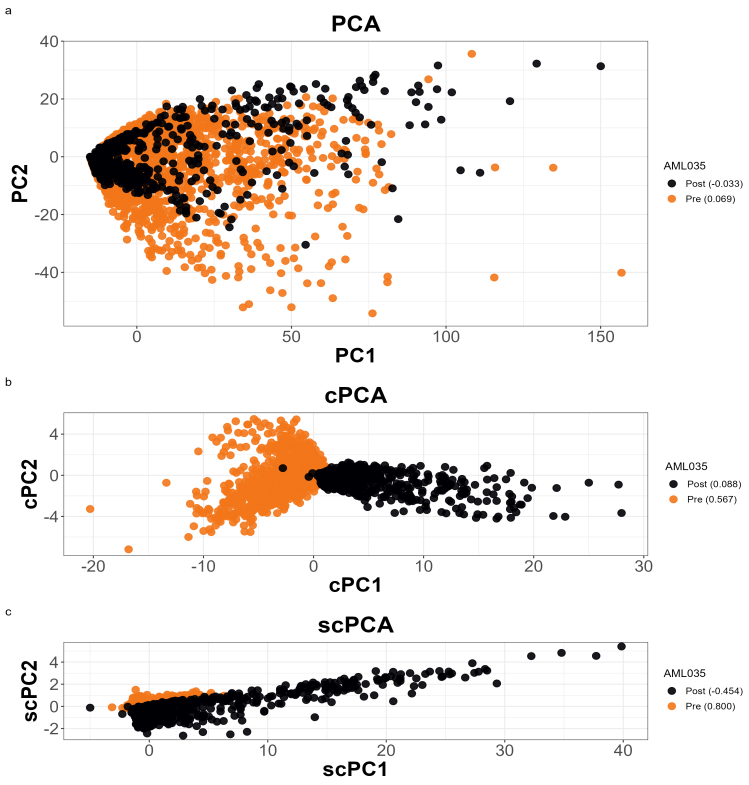

Figure 7: Two-dimensional embeddings of patient BMMCs generated by PCA, cPCA, and scPCA, with average silhouette widths indicating the strength of the biological signal (adapted from BHD, Fig. 3A).

Although cPCA and scPCA effectively capture the biological signal, scPCA produces sparse loadings. In Figures 8 and 9, cPCA shows numerous variables contributing to the first two PCs, whereas scPCA has fewer such variables. Specifically, cPCA, like traditional PCA, has 1000 non-zero loadings in each PC while scPCA has only 116 non-zero loadings in the first PC and 11 in the second PC. scPCA reduces some loadings to zero, making each principal component more interpretable.

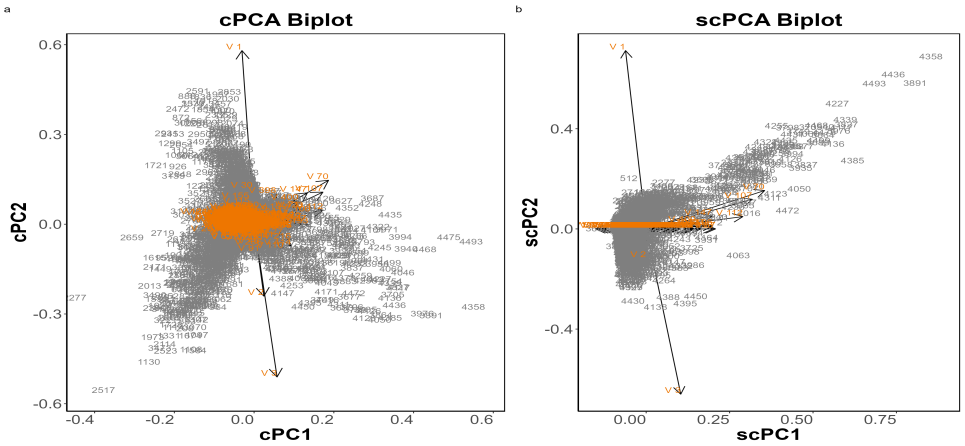

Figure 8: Biplot of cPCA and scPCA from the scRNA-seq data.

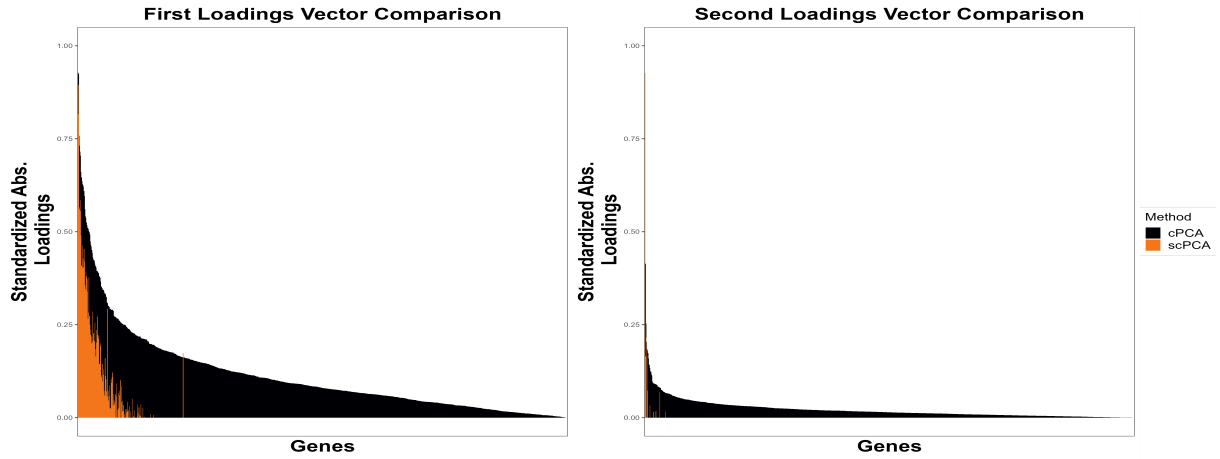

Figure 9: The standardized absolute loadings in the two leading loading vectors of cPCA and scPCA from the scRNA-seq data (adapted from BHD, Fig. 3B).

4.3 Dengue Microarray Data

As discussed by Boileau et al. (2020a), a study focusing on dengue patients used gene expression microarrays to examine the whole-blood transcriptome of 47 patients hospitalized at the Siriraj Hospital in Bangkok and 9 local health individuals (controls). Among the patients, 18 were identified as experiencing acute dengue fever (DF), 10 as experiencing acute dengue hemorrhagic fever (DHF), and 19 as convalescent at least 4 weeks after being discharged. In the data pre-processing, only $p = 500$ most variable genes were retained and the rest were filtered

out. The target dataset comprises log-transformed expression data from the microarrays of $n = 47$ dengue patients, while the background dataset includes log-transformed expression data from the control samples.

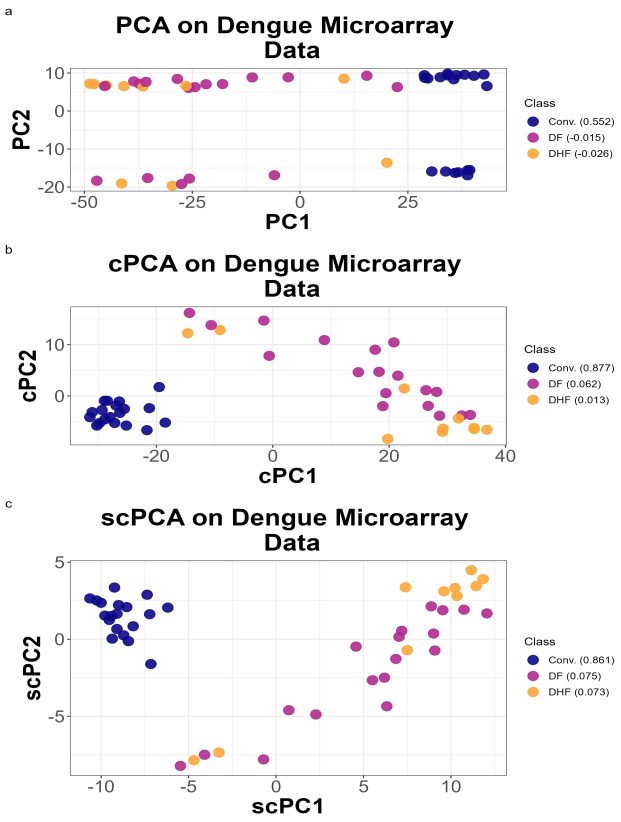

Figure 10: The two-dimensional representations of the dengue microarray dataset by PCA, cPCA and scPCA with average silhouette widths indicating the strength of the biological signal (adapted from BHD, Fig. 2A).

Figure 10 shows that none of the three dimensionality reduction techniques can distinguish between the three classes of dengue patients. Boileau et al. (2020a) discussed that, based on previous research, the transcriptomes of patients with DF and DHF are virtually indistinct. However, both cPCA and scPCA separate the convalescent patients from those with DF and DHF. The average silhouette widths for cPCA and scPCA are similar, with high values for the convalescent class and very low values for the DH and DHF classes.

Figures 11 and 12 display the variable contributions to the first and second principal components. While the results of cPCA may appear similar to that of scPCA in Figure 10, their differences are evident here. In cPCA, all 500 variables contribute to the first and second principal components, posing challenges for interpretation due to the large number of variables. Conversely, scPCA has only 34 and 18 variables contributing to the first and second principal components, respectively. The introduction of sparsity in scPCA reduces the contribution of most variables to zero, enhancing the interpretability of the principal components.

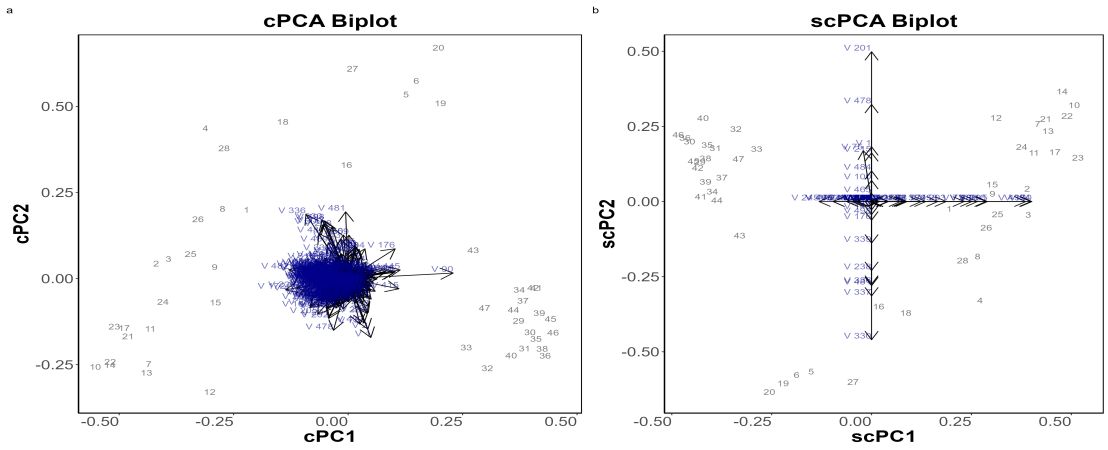

Figure 11: Biplot of cPCA and scPCA from the dengue microarray data.

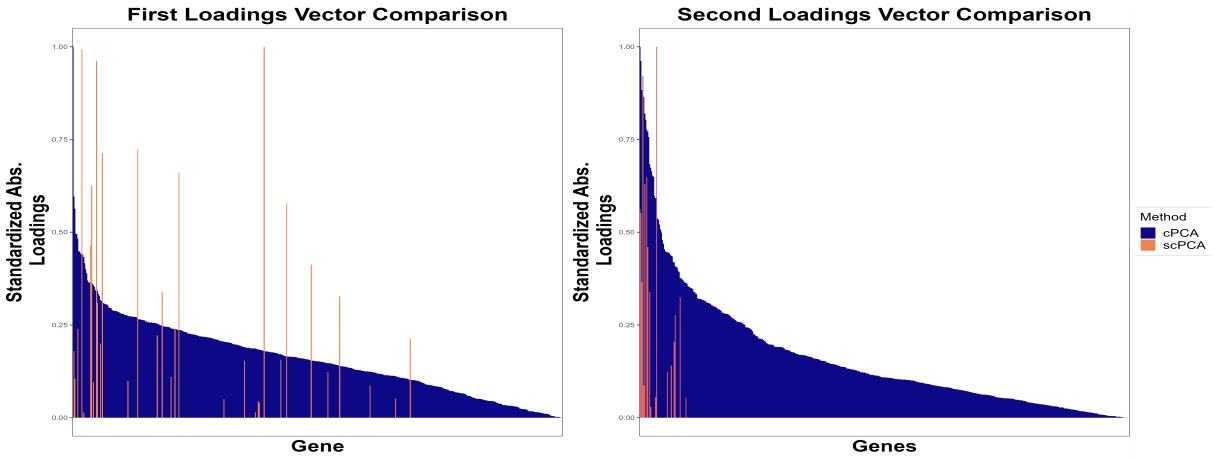

Figure 12: The standardized absolute loadings in the two leading loading vectors of cPCA and scPCA of the dengue microarray data (adapted from BHD, Fig.2B).

5 Discussion

The results from the dimensionality reduction methods were represented in a two-dimensional (2D) space, even when the explained variance was often low with just those two dimensions. This choice was made for ease of visualization and interpretation, as 2D plots are easier to interpret and understand than higher-dimensional displays. While this choice doesn't affect PCA, it does impact cPCA and scPCA in a possibly nonlinear fashion as it relates to the tuning of the relative weights chosen for the contrastive covariance matrix. The scPCA package defaults to computing two eigenvectors (so 2D) and this choice should be explored in more detail and how it might interact with other aspects of tuning the scPCA.

According to Boileau et al. (2020b), the contrastive parameter, γ, is a non-negative real number, which defaults to 40 logarithmically spaced values between 0.1 and 1000. This implies that the parameter can take on values as high as 1000. If γ is scaled to set the diagonal elements of the matrix to all 1's (correlation matrix), a contrastive parameter as high as 1000 would mean multiplying the background matrix by 1000 and subtracting from 1. While this range of contrastive parameters may work well with certain applications, different ranges of values may also work better in different situations. Thus, further investigation may be necessary for the best range of contrastive parameters in a given situation. By using correlation scale covariance matrices, some several rules might be possible.

The choice of the sparsity parameter, $\lambda_{1,j}$, also has an impact on the interpretability of the principal components, as it determines the number of non-zero loadings in each principal component. The default choice for the sparsity parameter is to use equidistant values between 0.05 and 1 as described by Boileau et al. (2020b). It is unclear if restricting this range might be warranted in certain situations or if there are possibly multiple values that produce nearly the same average silhouette values. In some sparse PCA implementations there is a tendency for the optimal penalization to make single PCs relate to only a single variable and provide little in the way of dimension reduction. This seems to be less common in this implementation of the method, but the performance of the sPCA tuning methods suggested in the context of

scPCA versus those used in Zou et al. (2006) should be explored in detail.

Instead of using k-means and maximizing the average silhouette width to select the tuning parameters, other options could be considered. For example, what about using agglomerative hierarchical clustering, which comes with choices of the agglomeration method. There are other methods of selecting the number of clusters that are competitors for using the average silhouette width, such as those discussed in Milligan (1981) or the Gap statistic Tibshirani et al. (2002). In particular, the Gap statistic might be a useful choice as it would allow the optimal choice to be a single cluster (all observations in the same group) to be the optimal version of the grouping structure. Further research is be needed to explore other alternatives, the potential interaction of these choices that could be used in different situations, and how those might work in sparse PCA, contrastive PCA, and sparse contrastive PCA with the variations in the proposed tuning methods.

Sparse contrastive PCA can be useful in various applications. For example, in Welhaven et al. (2024), data consisting of 35 femoral heads from end-stage osteoarthritis (OA) patients and 11 healthy subjects with 10,853 metabolite features were analyzed. The primary aim was to identify disease-associated OA metabolomic profiles, which would reveal OA's pathological mechanisms. Additionally, the study sought to examine and classify OA endotypes. While PCA showed near-perfect separation of healthy and OA cartilage, it could not clearly separate the four OA subgroups. scPCA could be useful in removing the healthy subjects' variation so we could see the differences in the OA groups more clearly.

References

Abid, A., Zhang, M. J., Bagaria, V. K., and Zou, J. (2018). Exploring patterns enriched in a dataset with contrastive principal component analysis. *Nature communications*, 9(1):2134.

Boileau, P., Hejazi, N. S., and Dudoit, S. (2020a). Exploring high-dimensional biological data with sparse contrastive principal component analysis. *Bioinformatics*, 36(11):3422–3430.

Boileau, P., Hejazi, N. S., and Dudoit, S. (2020b). scpca: A toolbox for sparse contrastive principal component analysis inr. *Journal of Open Source Software*, 5(46):2079.

Charrad, M., Ghazzali, N., Boiteau, V., and Niknafs, A. (2014). Nbclust: an r package for determining the relevant number of clusters in a data set. *Journal of statistical software*, 61:1–36.

Everitt, B. and Hothorn, T. (2011). *An introduction to applied multivariate analysis with R*. Springer Science & Business Media.

Hardle, W., Simar, L., et al. (2003). *Applied multivariate statistical analysis*.

Milligan, G. W. (1981). A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46:187–199.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Tibshirani, R., Walther, G., and Hastie, T. (2002). Estimating the Number of Clusters in a Data Set Via the Gap Statistic. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 63(2):411–423.

Tran, T. V. (2019). Monothetic cluster analysis with extensions to circular and functional data. *Montana State University - Bozeman, College of Letters Science*, Dissertation.

Welhaven, H. D., Welfley, A. H., Brahmachary, P., Bergstrom, A. R., Houske, E., Glimm, M., Bothner, B., Hahn, A. K., and June, R. K. (2024). Metabolomic profiles and pathways in osteoarthritic human cartilage: A comparative analysis with healthy cartilage. *Metabolites*, 14(4):183.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286.

Appendix

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
# if (!require("BiocManager", quietly = TRUE))
    #install.packages("BiocManager")
library(microbenchmark)
#BiocManager::install("scPCA")
library(scPCA)
library(tidyverse)
library(ggpubr)
#BiocManager::install("splatter")
library(splatter)
library(cluster)
library(corrplot)
#remotes::install_github("greenwood-stat/catstats2")
library(catstats2)
library(dplyr)
library(patchwork)
library(knitr)
library(xtable)
# BiocManager::install("genefilter")
library(genefilter)
library(GEOquery)

```

PCA
```

```
## PCA on covariance matrix of simulated data


```{r}
# load data from scPCA package
data(toy_df)
data(background_df)
# set seed for reproducibility
set.seed(1742)
# perform PCA
pca_sim <- prcomp(toy_df[, 1:30])
# plot the 2D rep using first 2 components
df <- as_tibble(list("PC1" = pca_sim$x[, 1],
                     "PC2" = pca_sim$x[, 2],
                     "label" = as.character(toy_df[, 31])))
p_pca <- ggplot(df, aes(x = PC1, y = PC2, colour = label)) +
  ggtitle("PCA on Simulated Data") +
  geom_point(alpha = 0.9, size = 4.5) +
  scale_color_viridis_d(end = 0.8) +
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 15),
        legend.title = element_text(size=15),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
p_pca
```
```

```
## Table of eigenvalues and eigenvectors

```{r}
pca_eigen_values <- (pca_sim$sdev)^2
pca_loadings <- data.frame(pca_sim$rotation)
# data frame for  first 2 eigenvalues
pca_data <- data.frame(
  PC = 1:2,
  Eigenvalue = pca_eigen_values[c(1:2)]
)
# Calculate the proportion of variance explained
pca_tot_var <- sum(pca_eigen_values)
pca_data$Proportion_of_Variance <- pca_data$Eigenvalue / pca_tot_var
# Select only the "PC", "Eigenvalue", and "Proportion_of_Variance" columns
pca_data <- pca_data[, c("PC", "Eigenvalue", "Proportion_of_Variance")]
# Convert the data frame to LaTeX code
table1 <- xtable(pca_data, caption = "Eigenvalues and proportion of variance explained by
# Write the LaTeX code to a .tex file
file <- file("table1.tex")
writeLines(print(table1, type = "latex"), con = file)
close(file)

loadings_pca <- data.frame(pca_loadings[,c(1,2)])
table2 <- xtable(loadings_pca, caption = "Loadings of the first two PCs from PCA on the s
file <- file("table2.tex")
writeLines(print(table2, type = "latex"), con = file)
```
```

```
close(file)
```

```

## Biplots for PCA (using covariance matrix)

```{r}
pca_scores <- pca_sim$x
pca_data <- cbind(label = rownames(toy_df),
                        as.data.frame(pca_scores[, c("PC1", "PC2")]))
#We want to use min-max feature scaling on the scores so they are between
#zero and one, the same as the loadings.
normalize <- function(x) return ((x - min(x)) / (max(x) - min(x)))
pca_data[, "PC1"] <- scale(normalize(pca_data[, "PC1"]),
                                  center = TRUE, scale = FALSE)
pca_data[, "PC2"] <- scale(normalize(pca_data[, "PC2"]),
                                  center = TRUE, scale = FALSE)

b_pca1 <- ggplot() +
  geom_point(data = pca_data, mapping = aes(x = PC1, y = PC2),
             alpha = 0.1, color = "white") +
  ggtitle("PCA Biplot")+
  coord_fixed()

b_pca2 <- b_pca1 +
  geom_segment(data = pca_loadings, aes(x = 0, y = 0,
                                        xend = PC1, yend = PC2),
               arrow = arrow(length = unit(0.03, "npc")),
```
```

```
              arrow.fill = "purple",  alpha = 0.95, ) +
  geom_text(data = pca_loadings, aes(x = PC1,
                                     y = PC2,
                                     label = rownames(pca_loadings)),
            hjust = 1, vjust = -0.2, colour = "purple",
            size = 4, check_overlap = FALSE) +
  geom_text(data = pca_data, aes(x = PC1, y = PC2, label = label),
            hjust = 1, vjust = -0.2, colour = "grey43",
            size = 4, check_overlap = FALSE)

b_pca3 <- b_pca2 +
  theme(axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill = NA, size = 1),
        axis.text=element_text(size=18),
        axis.text.x = element_text(angle = 360, hjust = 1),
        axis.title=element_text(size=23,face="bold"),
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "white", colour = "white"),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))

b_pca3
pca_toy <- (p_pca + b_pca3) + plot_annotation(tag_levels = "a")
ggsave("pca_toy.png", plot = pca_toy, width = 18, height = 9)
```

```
# Sparse PCA
## Sparse PCA on Covariance matrix of simulated data
```

```{r}
# perform sPCA on toy_df
penalties <- exp(seq(log(10), log(1000), length.out = 1))
df_ls <- lapply(penalties, function(penalty) {
  spca_sim <- elasticnet::spca(toy_df[, 1:30], K = 2, para = rep(penalty, 2),
                    type = "predictor", sparse = "penalty")
  spca_sim <- spca_sim$loadings
  spca_sim_p <- as.matrix(toy_df[, 1:30]) %*% spca_sim
  spca_out <- list("SPC1" = spca_sim_p[, 1],
                   "SPC2" = spca_sim_p[, 2],
                   "penalty" = round(rep(penalty, nrow(toy_df))),
                   "label"  = as.character(toy_df[, 31])) %>%
    as_tibble()
  return(spca_out)
})

df <- dplyr::bind_rows(df_ls)
# plot the results of sPCA
p_spca <- ggplot(df, aes(x = SPC1, y = SPC2, colour = label)) +
  geom_point(alpha = 0.9, size = 4.5) +
  ggtitle("Sparse PCA on Simulated Data") +
  xlab("sPC1") +
  ylab("sPC2") +
  scale_colour_viridis_d(end = 0.8)+
  theme_bw() +
  theme(axis.text=element_text(size=18),
```

```
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 15),
        legend.title = element_text(size=15),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
p_spca
```

```
## Biplot for sPCA (using covariance matrix)
```

```{r}
penalty <- exp(seq(log(10), log(1000), length.out = 30))
spca_siml <- elasticnet::spca(toy_df[,1:30], K=2, type="predictor",
                              sparse="penalty", para=penalty, use.corr = FALSE)
spca_var <- spca_siml$pev
spca_sim <- spca_siml$loadings
spca_sim_df <- as.matrix(toy_df[, 1:30]) %*% spca_sim
scpca_df <- data.frame("SPC1" = spca_sim_df[,1], "SPC2" = spca_sim_df[,2])
data <- cbind(label = rownames(toy_df), scpca_df)
normalize <- function(x) return ((x - min(x)) / (max(x) - min(x)))
data[, "sPC1"] <- scale(normalize(data[, "SPC1"]),
                        center = TRUE, scale = FALSE)
data[, "sPC2"] <- scale(normalize(data[, "SPC2"]),
                        center = TRUE, scale = FALSE)
loadings <- as.data.frame(spca_sim)
rownames(loadings) <- paste("V", 1:30)
colnames(loadings) <- c("sPC1", "sPC2")
```

```
b_spca1 <- ggplot() +
  geom_point(data = data, mapping = aes(x = sPC1, y = sPC2),
             alpha = 0.1, color = "white") +
  ggtitle("sPCA Biplot")+
  coord_fixed()


b_spca2 <- b_spca1 +
  geom_segment(data = loadings, aes(x = 0, y = 0,
                                          xend = sPC1, yend = sPC2),
               arrow = arrow(length = unit(0.03, "npc")),
               arrow.fill = "purple",  alpha = 0.95, ) +
  geom_text(data = loadings, aes(x = sPC1,
                                       y = sPC2,
                                       label = rownames(loadings)),
            hjust = 1, vjust = -0.2, colour = "purple",
            size = 4, check_overlap = FALSE) +
  geom_text(data = data, aes(x = sPC1, y = sPC2, label = label),
            hjust = 1, vjust = -0.2, colour = "grey43",
            size = 4, check_overlap = FALSE)


b_spca3 <- b_spca2 +
  theme(axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill = NA, size = 1),
        axis.text = element_text(size = 18),
        axis.text.x = element_text(angle = 360, hjust = 1),
        axis.title = element_text(size = 23, face = "bold"),
```

```
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "white", colour = "white"),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))

b_spca3

spca_toy <- (p_spca + b_spca3)  + plot_annotation(tag_levels = 'a')
ggsave("spca_toy.png", plot = spca_toy, width = 18, height = 9)
```

Table of eigenvalues and eigenvectors for sPCA

```{r}
# Sample data
spca_data <- data.frame(
  PC = 1:2,
  Proportion_of_Variance = spca_var
)
# Convert the data frame to LaTeX code
table3 <- xtable(spca_data, caption = "Proportion of variance explained by sPCA on simula
# Write the LaTeX code to a .tex file
file <- file("table3.tex")
writeLines(print(table3, type = "latex"), con = file)
close(file)

loadings_sPC <- data.frame(spca_sim[,c(1,2)])
table4 <- xtable(loadings_sPC, caption = "Loadings of the first two PCs from sPCA on the
```

```
file <- file("table4.tex")
writeLines(print(table4, type = "latex"), con = file)
close(file)
```

Contrastive PCA

```{r}
set.seed(1239)
cpca_sim <- scPCA(target = toy_df[, 1:30],
                  background = background_df,
                  penalties = 0,
                  n_centers = 4)

# create a dataframe to be plotted
cpca_df <- cpca_sim$x %>%
  as_tibble() %>%
  mutate(label = toy_df[, 31] %>% as.character)
colnames(cpca_df) <- c("cPC1", "cPC2", "label")
# plot the results
p_cpca <- ggplot(cpca_df, aes(x = cPC1, y = cPC2, colour = label)) +
  geom_point(alpha = 0.9, size = 4.5) +
  ggtitle(" Contrastive PCA on Simulated Data") +
  scale_colour_viridis_d(end = 0.8)+
  theme_bw() +
  theme(axis.text=element_text(size=18),


```

```
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 15),
        legend.title = element_text(size=15),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
p_cpca
```

```
## Biplot for cPCA
```

```{r}
cpca_loadings <- cpca_sim$rotation # Loadings Matrix
cpca_scores <- cpca_sim$x
colnames(cpca_scores) <- c("cPC1", "cPC2")
data <- cbind(label = rownames(toy_df), as.data.frame(cpca_scores[, c("cPC1", "cPC2")]))
normalize <- function(x) return ((x - min(x)) / (max(x) - min(x)))
data[, "cPC1"] <- scale(normalize(data[, "cPC1"]),
                        center = TRUE, scale = FALSE)
data[, "cPC2"] <- scale(normalize(data[, "cPC2"]),
                        center = TRUE, scale = FALSE)
loadings <- as.data.frame(cpca_loadings)
rownames(loadings) <- paste("V", 1:30)
colnames(loadings) <- c("cPC1", "cPC2")
b_cpca1 <- ggplot() +
  geom_point(data = data, mapping = aes(x = cPC1, y = cPC2),
             alpha = 0.1, color = "white") +
  ggtitle("cPCA Biplot")+
```

```
  coord_fixed()


b_cpca2 <- b_cpca1 +
  geom_segment(data = loadings, aes(x = 0, y = 0, xend = cPC1, yend = cPC2),
               arrow = arrow(length = unit(0.03, "npc")),
               arrow.fill = "purple",  alpha = 0.95, ) +
  geom_text(data = loadings, aes(x = cPC1, y = cPC2,
                                 label = rownames(loadings)),
            hjust = 1, vjust = -0.2, colour = "purple",
            size = 4, check_overlap = FALSE) +
  geom_text(data = data, aes(x = cPC1, y = cPC2, label = label),
            hjust = 1, vjust = -0.2, colour = "grey43",
            size = 4, check_overlap = FALSE)


b_cpca3 <- b_cpca2 +
  theme(axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill = NA, size = 1),
        axis.text = element_text(size = 18),
        axis.text.x = element_text(angle = 360, hjust = 1),
        axis.title = element_text(size = 23, face = "bold"),
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "white", colour = "white"),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))


b_cpca3
cpca_toy <- (p_cpca + b_cpca3)  + plot_annotation(tag_levels = 'a')
ggsave("cpca_toy.png", plot = cpca_toy, width = 18, height = 9)
```

```

## The contrastive covariance matrix

```{r}
safeColScale <- function(X,
                         center = TRUE,
                         scale = TRUE,
                         tol = .Machine$double.eps,
                         eps = 0.01,
                         scaled_matrix = FALSE) {

  # check argument types
  assertthat::assert_that(is.logical(center))
  assertthat::assert_that(is.logical(scale))
  assertthat::assert_that(is.numeric(tol))
  assertthat::assert_that(is.numeric(eps))
  assertthat::assert_that(is.logical(scaled_matrix))

  # input X must be a matrix for matrixStats
  if (!is.matrix(X) && class(X)[1] != "dgCMatrix" &&
      class(X)[1] != "DelayedMatrix") {
    X <- as.matrix(X)
  }
  # center if required
  if (center) {
```
```

```
    colMeansX <- Matrix::colMeans(X, na.rm = TRUE)
  } else {
    # just subtract off zero if not centering
    colMeansX <- rep(0, ncol(X))
  }
  # scale if required
  if (scale) {
    if (is.matrix(X) || class(X)[1] %in% c("dgCMatrix", "DelayedMatrix")) {
      colSdsX <- MatrixGenerics::colSds(X, na.rm = TRUE, useNames = FALSE)
    }
    colSdsX[colSdsX < tol] <- eps
  } else {
    colSdsX <- rep(1, length(colMeansX))
  }
  # compute re-centered and re-scaled output
  if (scaled_matrix) {
    stdX <- ScaledMatrix::ScaledMatrix(X, center = colMeansX, scale = colSdsX)
  } else {
    stdX <- t((t(X) - colMeansX) / colSdsX)
  }
  # return output
  return(stdX)
}
################################
#covariance matrix of target and background data
covMat <- function(data, center = TRUE, scale = TRUE, scaled_matrix = FALSE) {
  # center and scale the data matrix if required
```

```
  if (scale) {
    data <- safeColScale(data, center = TRUE, scale = TRUE,
                         scaled_matrix = scaled_matrix)
  } else {
    data <- safeColScale(data, center = TRUE, scale = FALSE,
                         scaled_matrix = scaled_matrix)
  }
  # compute the covariance matrix of the data
  if (is.matrix(data) || is.data.frame(data) || is_tibble(data)) {
    cov_mat <- coop::covar(data)
  } else {
    cov_mat <- 1 / (nrow(data) - 1) * Matrix::crossprod(data)
    cov_mat <- as.matrix(cov_mat)
  }
  return(cov_mat)
}



t_covM <- covMat(toy_df[, 1:30], center = F, scale = T)
b_covM <-  covMat(background_df, center = F, scale = T)
############################################################
#contrastive covariance matrix
contrastiveCov <- function(
  target, background, contrasts, center, scale, scaled_matrix = FALSE
) {
  # get the covariance matrices of the target and background
  c_target <- covMat(
```

```
    target, center = center, scale = scale, scaled_matrix = scaled_matrix
  )
  c_background <- covMat(
    background, center = center, scale = scale, scaled_matrix = scaled_matrix
  )

  # get the list of contrastive covariance matrices
  c_contrasts <- lapply(contrasts, function(x) {
    c_target - x * c_background
  })

  # output
  return(c_contrasts)
}
cl_covM <- contrastiveCov(toy_df[, 1:30], background_df,
                          contrasts = exp(seq(log(0.1), log(100),
                                              length.out = 5)), center = F, scale = T)
c_covM <- cl_covM[[1]]
```

```
## Comparing the covariance matrices (target, background and contrast)
```

```{r fig.width = 12, fig.height = 6}
p1 <- catstats2::ggcorrplot(corr = t_covM, hc.order = T) +
theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Target correlation matrix") +
```

```
  theme(plot.title = element_text(size = 17, hjust = 0.5, face = "bold"))+
  coord_fixed()
p2 <- catstats2::ggcorrplot(corr = b_covM, hc.order = T) +
  theme(axis.text.x = element_text(angle = 90))+
  ggtitle("Background correlation matrix")+
  theme(plot.title = element_text(size = 17, hjust = 0.5, face = "bold"))+
  coord_fixed()
p3 <- catstats2::ggcorrplot(corr = c_covM, hc.order = T) +
 theme(axis.text.x = element_text(angle = 90))+
  ggtitle("Contrastive matrix")+
  theme(plot.title = element_text(size = 17, hjust = 0.5, face = "bold"))+
  coord_fixed()
covM <- (p1 + p2 + p3) + plot_annotation(tag_levels = 'a')
ggsave("covM.png", plot = covM, width = 18, height = 4.5)
```
```

## Table of eigenvalues and eigenvectors for cPCA

```{r}
eigenvec_cpca <- cpca_sim$rotation
S <- t_covM
eigenvalues_cpca <- 0
total_variance_cpca <- sum(diag(S))
for (i in 1:ncol(eigenvec_cpca)) {
  eigenvalues_cpca[i] <- t(eigenvec_cpca[, i]) %*% S %*% eigenvec_cpca[, i]
}
prop_var_cpca <- (eigenvalues_cpca / total_variance_cpca)
```
```

```
eigenvalues_cpca
prop_var_cpca
cpca_data <- data.frame(
  PC = 1:2,
  Eigenvalue = eigenvalues_cpca,
  Proportion_of_Variance = prop_var_cpca
)
cpca_data <- cpca_data[, c("PC", "Eigenvalue", "Proportion_of_Variance")]
# Convert the data frame to LaTeX code
table5 <- xtable(cpca_data, caption = "Eigenvalues and proportion of variance explained b
# Write the LaTeX code to a .tex file
file <- file("table5.tex")
writeLines(print(table5, type = "latex"), con = file)
close(file)
loadings_cpca <- data.frame(eigenvec_cpca[,c(1,2)])
table6 <- xtable(loadings_cpca, caption = "Loadings of the first two PCs from cPCA on the
file <- file("table6.tex")
writeLines(print(table6, type = "latex"), con = file)
close(file)
```

Sparse Contrastive PCA

```{r}
scpca_sim <- scPCA(target = toy_df[, 1:30],
                   background = background_df,
                   n_centers = 4,
```

```
                    penalties = exp(seq(log(0.01), log(0.5), length.out = 10)),
                    alg = "var_proj")

scpca_df <- scpca_sim$x %>%
  as_tibble() %>%
  mutate(label = toy_df[, 31] %>% as.character)
colnames(scpca_df) <- c("scPC1", "scPC2", "label")
# plot the results
p_scpca <- ggplot(scpca_df, aes(x = scPC1, y = scPC2, colour = label)) +
  geom_point(alpha = 0.9, size = 4.5) +
  ggtitle("Sparse Contrastive PCA on Simulated Data") +
  scale_colour_viridis_d(end = 0.8) +
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 15),
        legend.title = element_text(size=15),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
p_scpca
```

```

## Biplot for scPCA

```{r}
scpca_loadings <- scpca_sim$rotation # Loadings Matrix
scpca_scores <- scpca_sim$x
```
```

```
colnames(scpca_scores) <- c("scPC1", "scPC2")
data <- cbind(label = rownames(toy_df),
              as.data.frame(scpca_scores[, c("scPC1", "scPC2")]))
normalize <- function(x) return ((x - min(x)) / (max(x) - min(x)))
data[, "scPC1"] <- scale(normalize(data[, "scPC1"]), center = TRUE, scale = FALSE)
data[, "scPC2"] <- scale(normalize(data[, "scPC2"]), center = TRUE, scale = FALSE)
loadings <- as.data.frame(scpca_loadings)
rownames(loadings) <- paste("V", 1:30)
colnames(loadings) <- c("scPC1", "scPC2")

b_scpca1 <- ggplot() +
  geom_point(data = data, mapping = aes(x = scPC1, y = scPC2),
             alpha = 0.1, color = "white") +
  ggtitle("scPCA Biplot")+
  coord_fixed()

b_scpca2 <- b_scpca1 +
  geom_segment(data = loadings, aes(x = 0, y = 0, xend = scPC1, yend = scPC2),
               arrow = arrow(length = unit(0.03, "npc")),
               arrow.fill = "purple",  alpha = 0.95, ) +
  geom_text(data = loadings, aes(x = scPC1, y = scPC2,
                                 label = rownames(loadings)),
            hjust = 1, vjust = -0.2, colour = "purple",
            size = 4, check_overlap = FALSE) +
  geom_text(data = data, aes(x = scPC1, y = scPC2, label = label),
            hjust = 1, vjust = -0.2, colour = "grey43",
            size = 4, check_overlap = FALSE)
```

```
b_scpca3 <- b_scpca2 +
  theme(axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill = NA, size = 1),
        axis.text = element_text(size = 18),
        axis.text.x = element_text(angle = 360, hjust = 1),
        axis.title = element_text(size = 23, face = "bold"),
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "white", colour = "white"),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))
b_scpca3
scpca_toy <- (p_scpca + b_scpca3)  + plot_annotation(tag_levels = 'a')
ggsave("scpca_toy.png", plot = scpca_toy, width = 18, height = 7)
```

```
## Table of eigenvalues and eigenvectors for scPCA

```{r}
eigenvec_scpca <- scpca_sim$rotation
S <- t_covM
eigenvalues_scpca <- 0
total_variance_scpca <- sum(diag(S))

for (i in 1:ncol(eigenvec_scpca)) {
  eigenvalues_scpca[i] <- t(eigenvec_scpca[, i]) %*% S %*% eigenvec_scpca[, i]
}
```
```

```
prop_var_scpca <- eigenvalues_scpca / total_variance_scpca
eigenvalues_scpca
prop_var_scpca

scpca_data <- data.frame(
  PC = 1:2,
  Eigenvalue = eigenvalues_scpca,
  Proportion_of_Variance = prop_var_scpca
)
scpca_data <- scpca_data[, c("PC", "Eigenvalue", "Proportion_of_Variance")]
# Convert the data frame to LaTeX code
table7 <- xtable(scpca_data, caption = "Eigenvalues and proportion of variance explained
# Write the LaTeX code to a .tex file
file <- file("table7.tex")
writeLines(print(table7, type = "latex"), con = file)
close(file)
loadings_scpca <- data.frame(eigenvec_scpca[,c(1,2)])

table8 <- xtable(loadings_scpca, caption = "Loadings of the first two PCs from scPCA on t
file <- file("table8.tex")
writeLines(print(table8, type = "latex"), con = file)
close(file)
```

```
# Toy Example 2
### PCA on correlation matrix
```

```{r}
data(toy_df)
data(background_df)

# set seed for reproducibility
set.seed(1742)

# perform PCA
pca_sim2 <- prcomp(toy_df[, 1:30], scale = TRUE)
# pca_sim <- princomp(toy_df[, 1:30], cor = T)

eigen_values_pca2 <- (pca_sim2$sdev)^2
loadings_pca2 <- data.frame(pca_sim2$rotation)

# plot the 2D rep using first 2 components
pca2_df <- as_tibble(list("PC1" = pca_sim2$x[, 1],
                     "PC2" = pca_sim2$x[, 2],
                     "label" = as.character(toy_df[, 31])))
p_pca2 <- ggplot(pca2_df, aes(x = PC1, y = PC2, colour = label)) +
  ggtitle("PCA on Simulated Data") +
  geom_point(alpha = 0.9, size = 4.5) +
  scale_color_viridis_d(end = 0.8) +
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 15),
        legend.title = element_text(size=15),
```

```
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
p_pca2

```

Table of eigenvalues and eigenvectors for PCA on correlation matrix

```{r}
# Sample data
pca2_data <- data.frame(
  PC = 1:2,
  Eigenvalue = eigen_values_pca2[c(1,2)]
)
# Calculate the proportion of variance explained
total_variance <- sum(eigen_values_pca2)
pca2_data$Proportion_of_Variance <- pca2_data$Eigenvalue / total_variance
# Select only the "PC", "Eigenvalue", and "Proportion_of_Variance" columns
# pca2_data <- pc_data[, c("PC", "Eigenvalue", "Proportion_of_Variance")]
# Convert the data frame to LaTeX code
table9 <- xtable(pca2_data, caption = "Eigenvalues and proportion of variance explained b
# Write the LaTeX code to a .tex file
file <- file("table9.tex")
writeLines(print(table9, type = "latex"), con = file)
close(file)
loadings_pca2 <- data.frame(loadings_pca2[,c(1,2)])
table10 <- xtable(loadings_pca2, caption = "Loadings of the first two PCs from PCA on the
```

```
file <- file("table10.tex")
writeLines(print(table10, type = "latex"), con = file)
close(file)
```

Sparse PCA on Correlation Matrix

```{r}
penalties <- exp(seq(log(10), log(1000), length.out = 1))
df_ls <- lapply(penalties, function(penalty) {
  spca_sim <- elasticnet::spca(toy_df[, 1:30], K = 2, para = rep(penalty, 2),
                    type = "predictor", sparse = "penalty", use.corr = T)$loadings
  spca_sim_p <- as.matrix(toy_df[, 1:30]) %*% spca_sim
  spca_out <- list("SPC1" = spca_sim_p[, 1],
                   "SPC2" = spca_sim_p[, 2],
                   "penalty" = round(rep(penalty, nrow(toy_df))),
                   "label"  = as.character(toy_df[, 31])) %>%
    as_tibble()
  return(spca_out)
})
df <- dplyr::bind_rows(df_ls)
##############################
# plot the results of sPCA
p_spca2 <- ggplot(df, aes(x = SPC1, y = SPC2, colour = label)) +
  geom_point(alpha = 0.9, size = 4.5) +
  ggtitle("Sparse PCA on Simulated Data") +
  xlab("sPC1")+
```

```
  ylab("sPC2")+
  scale_colour_viridis_d(end = 0.8)+
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 15),
        legend.title = element_text(size=15),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
p_spca2
toy2 <-  (p_pca2 + p_spca2) / (p_cpca + p_scpca) + plot_annotation(tag_levels = 'a')
ggsave("toy2.png", plot = toy2, width = 18, height = 9)
```

Table of eigenvalues and eigenvectors for sPCA on correlation matrix

```{r}

penalty <- exp(seq(log(10), log(1000), length.out = 30))
spca_siml <- elasticnet::spca(toy_df[,1:30], K=2, type="predictor",
spca_var2 <- spca_siml$pev
spca_sim2 <- spca_siml$loadings
spca_sim2_df <- as.matrix(toy_df[, 1:30]) %*% spca_sim2
spca2_df <- data.frame("SPC1" = spca_sim2_df[,1], "SPC2" = spca_sim2_df[,2])
eigenvec_spca2 <- spca_sim2
S <- t_covM
eigenvalues_spca <- 0
```

```
total_variance_spca <- sum(diag(S))
for (i in 1:ncol(eigenvec_spca2)) {
  eigenvalues_spca[i] <- t(eigenvec_spca2[, i]) %*% S %*% eigenvec_spca2[, i]
}
prop_var_spca2 <- (eigenvalues_spca / total_variance_spca) * 100
eigenvalues_spca
prop_var_spca2
# Sample data
spca_data <- data.frame(
  PC = 1:2,
  Proportion_of_Variance = spca_var2
)
# Convert the data frame to LaTeX code
table12 <- xtable(spca_data, caption = "Proportion of Variance explained by the first two
# Write the LaTeX code to a .tex file
file <- file("table12.tex")
writeLines(print(table12, type = "latex"), con = file)
close(file)
loadings_spca2 <- data.frame(spca_sim2[,c(1,2)])
table11 <- xtable(loadings_spca2, caption = "Loadings of the first two PCs from sPCA on t
file <- file("table11.tex")
writeLines(print(table11, type = "latex"), con = file)
close(file)
```

```
# RNA Seq Data


```{r  message=FALSE, warning=FALSE}
# load the data
library(here)
source(file = here("analyses/bmmc_data/helpers/load_count_data.R"))
# retain genes that contain 5 or more non-zero counts accross all cells
sce <- sce[(Matrix::rowSums(counts(sce) != 0) > 4), ]
# retain 1000 most variable genes
vars <- rowVars(as.matrix(log1p(counts(sce))))
names(vars) <- rownames(sce)
vars <- sort(vars, decreasing = TRUE)
core <- sce[names(vars)[1:1000], ]
# split into target and background datasets
background_sce <- core[, which(core$cell_prov %in% c("healthy_1", "healthy_2"))]
target_sce <- core[, which(!(core$cell_prov %in% c("healthy_1", "healthy_2")))]
patient_027_sce <- core[, which(core$cell_prov %in%
                                  c("pre_trans_027", "post_trans_027"))]
patient_035_sce <- core[, which(core$cell_prov %in%
                                  c("pre_trans_035", "post_trans_035"))]
background <- t(counts(background_sce))
target <- t(counts(target_sce))
patient_027 <- t(counts(patient_027_sce))
patient_035 <- t(counts(patient_035_sce))
# get the classes
bmmc_class <- target_sce$cell_prov %>% factor
pat027_class <- patient_027_sce$cell_prov %>% factor
```
```

```
pat035_class <- patient_035_sce$cell_prov %>% factor
```

```
# AML035
## PCA

```{r pca_pat035}
# perform PCA
bmmc_pca <- prcomp(patient_035, center = TRUE, scale. = TRUE)
# compute the average silhouette widths
group_mem <- if_else(pat035_class == "pre_trans_035", 1, 2)
pca_group_sil <- silhouette(group_mem, dist(bmmc_pca$x))
# plot the 2D representation
pca_df <- data.frame(
  PC1 = bmmc_pca$x[, 1],
  PC2 = bmmc_pca$x[, 2],
  class = factor(pat035_class, levels = c("pre_trans_035", "post_trans_035"))
) %>%
  dplyr::mutate(
    class = if_else(
      class == "pre_trans_035",
      paste0(
        "Pre (",
        sprintf("%.3f", round(summary(pca_group_sil)$clus.avg.widths[1], 3)),
        ")"),
      paste0(
        "Post (",
```
```

```
        sprintf("%.3f", round(summary(pca_group_sil)$clus.avg.widths[2], 3)),
        ")"),
    )
  )
pca_pb <- pca_df %>%
  ggplot(aes(x = PC1, y = PC2, colour = class)) +
  geom_point(alpha = 0.9, size = 3.5) +
  ggtitle("PCA") +
  scale_colour_viridis_d(name = "AML035",
                         end = 0.7, option = "inferno") +
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 10.5),
        legend.title = element_text(size=12),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
pca_pb
```

```

## cPCA on RNA Seq Data

```{r cpca_pat035}
set.seed(1436262)
# perform cpca
bmmc_cpca <- scPCA(as.matrix(patient_035), as.matrix(background),
```
```

```
                    center = TRUE, scale = TRUE, penalties = 0, n_centers = 2,
                    max_iter = 1000)
# compute the average silhouette widths
cpca_group_sil <- silhouette(group_mem, dist(bmmc_cpca$x))
# plot the 2D representation
cpca_df <- data.frame(
  cPC1 = bmmc_cpca$x[, 1],
  cPC2 = bmmc_cpca$x[, 2],
  class = factor(pat035_class, levels = c("pre_trans_035", "post_trans_035"))
) %>%
  dplyr::mutate(
    class = if_else(
      class == "pre_trans_035",
      paste0(
        "Pre (",
        sprintf("%.3f", round(summary(cpca_group_sil)$clus.avg.widths[1], 3)),
        ")"),
      paste0(
        "Post (",
        sprintf("%.3f", round(summary(cpca_group_sil)$clus.avg.widths[2], 3)),
        ")"),
    )
  )
cpca_pb <- cpca_df %>%
  ggplot(aes(x = cPC1, y = cPC2, colour = class)) +
  geom_point(alpha = 0.9, size = 3.5) +
  ggtitle("cPCA") +
```

```
  scale_colour_viridis_d(name = "AML035",
                         end = 0.7, option = "inferno") +
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 10.5),
        legend.title = element_text(size=12),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
cpca_pb
```

Biplot for cPCA on RNA Seq Data

```{r}
bmmc_cpca_loadings <- bmmc_cpca$rotation # Loadings Matrix
bmmc_cpca_scores <- bmmc_cpca$x
colnames(bmmc_cpca_scores) <- c("cPC1", "cPC2")
bmmc_data <- cbind(label= paste(1:4501),
              as.data.frame(bmmc_cpca_scores[, c("cPC1", "cPC2")]))
normalize <- function(x) return ((x - min(x)) / (max(x) - min(x)))
bmmc_data[, "cPC1"] <- scale(normalize(bmmc_data[, "cPC1"]),
                             center = TRUE, scale = FALSE)
bmmc_data[, "cPC2"] <- scale(normalize(bmmc_data[, "cPC2"]),
                             center = TRUE, scale = FALSE)
loadings <- as.data.frame(bmmc_cpca_loadings)
```

```
rownames(loadings) <- paste("V", 1:1000)
colnames(loadings) <- c("cPC1", "cPC2")

bc_p1 <- ggplot() +
  geom_point(data = bmmc_data, mapping = aes(x = cPC1, y = cPC2),
             alpha = 0.1, color = "white") +
  ggtitle("cPCA Biplot")+
  coord_fixed()
bc_p2 <- bc_p1 +
  geom_segment(data = loadings, aes(x = 0, y = 0, xend = cPC1, yend = cPC2),
               arrow = arrow(length = unit(0.03, "npc")),
               arrow.fill = "darkorange2",  alpha = 0.95, ) +
  geom_text(data = bmmc_data, aes(x = cPC1, y = cPC2, label = label),
            hjust = 1, vjust = -0.2, colour = "grey50",
            size = 4, check_overlap = FALSE)+
  geom_text(data = loadings, aes(x = cPC1, y = cPC2,
                                 label = rownames(loadings)),
            hjust = 1, vjust = -0.2, colour = "darkorange2",
            size = 4, check_overlap = FALSE)
bc_p3 <- bc_p2 +
  theme(axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill = NA, size = 1),
        axis.text = element_text(size = 18),
        axis.text.x = element_text(angle = 360, hjust = 1),
        axis.title = element_text(size = 23, face = "bold"),
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "white", colour = "white"),
```

```
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))
bc_p3
```

scPCA on RNA Seq Data

```{r scpca_pat035}
# load results from cluster files
bmmc_scpca <- readRDS(here("analyses/bmmc_data/data/bmmc_scpca_035.rds"))
# compute the average silhouette widths
scpca_group_sil <- silhouette(group_mem, dist(bmmc_scpca$x))
# plot the 2D representation
scpca_df <- data.frame(
  scPC1 = bmmc_scpca$x[, 1],
  scPC2 = bmmc_scpca$x[, 2],
  class = factor(pat035_class, levels = c("pre_trans_035", "post_trans_035"))
) %>%
  dplyr::mutate(
    class = if_else(
      class == "pre_trans_035",
      paste0(
        "Pre (",
        sprintf("%.3f", round(summary(scpca_group_sil)$clus.avg.widths[1], 3)),
        ")"),
      paste0(
        "Post (",
        sprintf("%.3f", round(summary(scpca_group_sil)$clus.avg.widths[2], 3)),
```

```
        ")"),
    )
  )
scpca_pb <- scpca_df %>%
  ggplot(aes(x = scPC1, y = scPC2, colour = class)) +
  geom_point(alpha = 0.9, size = 3.5) +
  ggtitle("scPCA") +
  scale_colour_viridis_d(name = "AML035",
                         end = 0.7, option = "inferno") +
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 10.5),
        legend.title = element_text(size=12),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
scpca_pb
RNA_plot <-  (pca_pb + cpca_pb + scpca_pb) + plot_annotation(tag_levels = 'a')
ggsave("RNA_plot.png", plot = RNA_plot, width = 18, height = 3.5)
```

```
## Biplot for scPCA on RNA Seq Data
```

```{r}
bmmc_scpca_loadings <- bmmc_scpca$rotation # Loadings Matrix
bmmc_scpca_scores <- bmmc_scpca$x
```

```
colnames(bmmc_scpca_scores) <- c("scPC1", "scPC2")
bmmc_data <- cbind(label= paste(1:4501),
             as.data.frame(bmmc_scpca_scores[, c("scPC1", "scPC2")]))
normalize <- function(x) return ((x - min(x)) / (max(x) - min(x)))
bmmc_data[, "scPC1"] <- scale(normalize(bmmc_data[, "scPC1"]),
                              center = TRUE, scale = FALSE)
bmmc_data[, "scPC2"] <- scale(normalize(bmmc_data[, "scPC2"]),
                              center = TRUE, scale = FALSE)
loadings <- as.data.frame(bmmc_scpca_loadings)
rownames(loadings) <- paste("V", 1:1000)
colnames(loadings) <- c("scPC1", "scPC2")
bsc_p1 <- ggplot() +
  geom_point(data = bmmc_data, mapping = aes(x = scPC1, y = scPC2),
             alpha = 0.1, color = "white") +
  ggtitle("scPCA Biplot")+
  coord_fixed()
bsc_p2 <- bsc_p1 +
  geom_segment(data = loadings, aes(x = 0, y = 0, xend = scPC1, yend = scPC2),
               arrow = arrow(length = unit(0.03, "npc")),
               arrow.fill = "darkorange2",  alpha = 0.95, ) +
  geom_text(data = bmmc_data, aes(x = scPC1, y = scPC2, label = label),
            hjust = 1, vjust = -0.2, colour = "grey50",
            size = 4, check_overlap = FALSE)+
  geom_text(data = loadings, aes(x = scPC1, y = scPC2,
                                 label = rownames(loadings)),
            hjust = 1, vjust = -0.2, colour = "darkorange2",
            size = 4, check_overlap = FALSE)
```

```
bsc_p3 <- bsc_p2 +
  theme(axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill = NA, size = 1),
        axis.text = element_text(size = 18),
        axis.text.x = element_text(angle = 360, hjust = 1),
        axis.title = element_text(size = 23, face = "bold"),
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "white", colour = "white"),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))
bsc_p3
biplot_bmmc <- (bc_p3 + bsc_p3)  + plot_annotation(tag_levels = 'a')
ggsave("biplot_bmmc.png", plot = biplot_bmmc, width = 18, height = 9)
```

Analysis of Genes with Non-Zero Entries in Loadings Matrix

The following table provides the gene symbol and loadings of the genes with non-zero entries in one of the loadings vectors of the first two scPCs.

```{r non_zer_bmmc_pat035}
rot_035_df <- bmmc_scpca$rotation %>%
  as_tibble %>%
  dplyr::mutate(
    gene_sym = rowData(patient_035_sce)$X2,
    scPC1 = V1,
```

```
    scPC2 = V2
  ) %>%
  select(-V1, -V2) %>%
  dplyr::filter(scPC1 != 0 | scPC2 != 0)
rot_035_df
```

```{r}
# number of non-zero loadings for the first two scPCs - scRNA-seq data
nonzero_bmmc <- data.frame(bmmc_scpca$rotation)
colSums(nonzero_bmmc != 0)
```

Comparison of Loadings: cPCA and scPCA

```{r pat035_comp, message = FALSE, warning = FALSE}
# compute the relative absolute loading ratios
cpca_loads_df <- bmmc_cpca$rotation %>%
  as.data.frame() %>%
  dplyr::mutate(
    V1 = abs(V1),
    V2 = abs(V2),
    V1 = (V1-min(V1))/(max(V1)-min(V1)),
    V2 = (V2-min(V2))/(max(V2)-min(V2))
  )
scpca_loads_df <- bmmc_scpca$rotation %>%
  as.data.frame() %>%
```

```
  dplyr::mutate(
    V1 = abs(V1),
    V2 = abs(V2),
    V1 = (V1-min(V1))/(max(V1)-min(V1)),
    V2 = (V2-min(V2))/(max(V2)-min(V2))
  )

# create the loadings comparison plot
load_diff_df <- bind_rows(
  cpca_loads_df,
  scpca_loads_df
) %>%
  dplyr::mutate(
    sparse = c(rep("0", ncol(patient_035)),
               rep("1", ncol(patient_035))),
    sparse = factor(sparse, labels = c("cPCA", "scPCA")),
    gene = rep(1:ncol(patient_035), 2)
  )

colnames(load_diff_df) <- c("comp1", "comp2", "sparse", "gene")

# order the genes based on decreasing rel abs cpca loadings
ord_1_gene <- factor(load_diff_df$gene,
                     levels = sort(load_diff_df$comp1[1:ncol(patient_035)],
                                   decreasing = TRUE, index.return = TRUE)$ix)
ord_2_gene <- factor(load_diff_df$gene,
                     levels = sort(load_diff_df$comp2[1:ncol(patient_035)],
```

```
                                    decreasing = TRUE, index.return = TRUE)$ix)
load_diff_df <- load_diff_df %>%
  dplyr::mutate(
    gene1 = ord_1_gene,
    gene2 = ord_2_gene
  )

p1 <- load_diff_df %>%
  ggplot(aes(y = abs(comp1), x = gene1)) +
  geom_bar(stat = "identity", position = position_identity(), width = 1) +
  xlab("Genes") +
  ylab("Standardized Abs.\n Loadings") +
  ggtitle("First Loadings Vector Comparison") +
  scale_fill_viridis_d(name = "Method",
                       labels = c("cPCA", "scPCA"),
                       end = 0.7, option = "inferno") +
  scale_alpha_discrete(range = c(0.5, 1),
                       guide = FALSE) +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
       plot.title = element_text(size = 23.5, hjust = 0.5, face = "bold"),
       legend.text = element_text(size = 15),
       legend.title = element_text(size=15),
       axis.title=element_text(size=23,face="bold"))
```

```
p2 <- load_diff_df %>%
  ggplot(aes(y = abs(comp2), x = gene2)) +
  geom_bar(stat = "identity", position = position_identity(), width = 1) +
  xlab("Genes") +
  ylab("Standardized Abs.\n Loadings") +
  ggtitle("Second Loadings Vector Comparison") +
  scale_fill_viridis_d(name = "Method",
                       labels = c("cPCA", "scPCA"),
                       end = 0.7, option = "inferno") +
  scale_alpha_discrete(range = c(0.5, 1),
                       guide = FALSE) +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
       plot.title = element_text(size = 23.5, hjust = 0.5, face = "bold"),
       legend.text = element_text(size = 15),
       legend.title = element_text(size=15),
       axis.title=element_text(size=23,face="bold"))


loadings_compb <- annotate_figure(
  ggarrange(p1, p2, nrow = 1, ncol = 2,
            common.legend = TRUE, legend = "right")
)
loadings_compb
```

```
ggsave("loadings_compb.png", plot = loadings_compb, width = 18, height = 9)
```

Dengue Microarray Data.

The microarray data is fetched from the GEO (accession number GSE51808). The data consists of the transcriptome of blood samples from 56 Thai males, 28 of which were hospitalized with severe dengue, 19 patients which were convalescent 4 weeks after contracting dengue and 9 healthy controls. Among the patients with dengue, 18 had a fever and 10 had a hemorrhagic fever.

The data was filtered; only the 500 most variable genes were retained. The transcriptome data of the 9 healthy controls were removed and used as a background dataset for cPCA and scPCA. The remaining samples consist of the target data.

```{r  message=FALSE, warning=FALSE}
# load the data, already log2 transformed
ges <- getGEO("GSE51808")$GSE51808_series_matrix.txt.gz

# load the gene names
# download table from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL13158
gene_names <- read_tsv(here("analyses/dengue_data/GPL13158-5065.txt"),
                       skip = 16)

# keep the 500 most variable genes
var_filt_ges <- varFilter(ges, var.cutoff = 1-500/nrow(exprs(ges)))
```

```
# extract the target and background datasets
control_label <- which(var_filt_ges$`status:ch1` == "control")
target <- t(exprs(var_filt_ges)[, -control_label])
background <- t(exprs(var_filt_ges)[, control_label])

# get the target data labels
dengue_class <- var_filt_ges$`status:ch1`[-control_label]
```

PCA for Dengue Microarray Data

```{r pca}
# perform PCA
dengue_pca <- prcomp(target, center = TRUE, scale. = FALSE)

# compute the average silhouette widths
group_mem <- if_else(dengue_class == "convalescent", 1,
                     if_else(dengue_class == "DF", 2, 3))
pca_group_sil <- silhouette(group_mem, dist(dengue_pca$x))

# plot the 2D representation
pca_df <- data.frame(
  PC1 = dengue_pca$x[, 1],
  PC2 = dengue_pca$x[, 2],
  class = dengue_class
) %>%
  dplyr::mutate(
```

```
    class = if_else(
      class == "convalescent",
      paste0(
        "Conv. (",
        sprintf("%.3f", round(summary(pca_group_sil)$clus.avg.widths[1], 3)),
        ")"),
      if_else(
        class == "DF",
        paste0(
        "DF (",
        sprintf("%.3f", round(summary(pca_group_sil)$clus.avg.widths[2], 3)),
        ")"),
        paste0("DHF (",
        sprintf("%.3f", round(summary(pca_group_sil)$clus.avg.widths[3], 3)),
        ")")
      )
    )
  )

pca_pd <- pca_df %>%
  ggplot(aes(x = PC1, y = PC2, colour = class)) +
  geom_point(alpha = 0.9, size = 4.5) +
  ggtitle("PCA on Dengue Microarray \n Data") +
  scale_colour_viridis_d(name = "Class", end=0.8,
                         option = "plasma") +
  theme_bw() +
  theme(axis.text=element_text(size=18),
```

```
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 10.5),
        legend.title = element_text(size= 12),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
pca_pd
```

cPCA for Dengue Microarray Data

```{r cpca}
# perform cpca
dengue_cpca <- scPCA(target, background, center = TRUE,
                     penalties = 0, n_centers = 3, max_iter = 1000)
# compute the average silhouette widths
cpca_group_sil <- silhouette(group_mem, dist(dengue_cpca$x))
# plot the 2D representation
cpca_df <- data.frame(
  cPC1 = -dengue_cpca$x[, 1],
  cPC2 = dengue_cpca$x[, 2],
  class = dengue_class
) %>%
  dplyr::mutate(
    class = if_else(
      class == "convalescent",
      paste0(
        "Conv. (",


```

```
        sprintf("%.3f", round(summary(cpca_group_sil)$clus.avg.widths[1], 3)),
        ")"),
      if_else(
        class == "DF",
        paste0(
        "DF (",
        sprintf("%.3f", round(summary(cpca_group_sil)$clus.avg.widths[2], 3)),
        ")"),
        paste0("DHF (",
        sprintf("%.3f", round(summary(cpca_group_sil)$clus.avg.widths[3], 3)),
        ")")
      )
    )
  )
cpca_pd <- cpca_df %>%
  ggplot(aes(x = cPC1, y = cPC2, colour = class)) +
  geom_point(alpha = 0.9, size = 4.5) +
  ggtitle("cPCA on Dengue Microarray \n Data") +
  scale_colour_viridis_d(name = "Class", end=0.8,
                         option = "plasma") +
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 10.5),
        legend.title = element_text(size= 12),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
```

```
cpca_pd
```

```
## bilpot for cPCA of microarray data
```

```{r}
dengue_cpca_loadings <- dengue_cpca$rotation # Loadings Matrix
dengue_cpca_scores <- dengue_cpca$x
colnames(dengue_cpca_scores) <- c("cPC1", "cPC2")

dengue_data <- cbind(label= paste(1:47),
              as.data.frame(dengue_cpca_scores[, c("cPC1", "cPC2")]))
normalize <- function(x) return ((x - min(x)) / (max(x) - min(x)))
dengue_data[, "cPC1"] <- scale(normalize(dengue_data[, "cPC1"]),
                               center = TRUE, scale = FALSE)
dengue_data[, "cPC2"] <- scale(normalize(dengue_data[, "cPC2"]),
                               center = TRUE, scale = FALSE)
loadings <- as.data.frame(dengue_cpca_loadings)
rownames(loadings) <- paste("V", 1:500)
colnames(loadings) <- c("cPC1", "cPC2")

dc_p1 <- ggplot() +
  geom_point(data = dengue_data, mapping = aes(x = cPC1, y = cPC2),
             alpha = 0.1, color = "white") +
  ggtitle("cPCA Biplot")+
  coord_fixed()
```

```
dc_p2 <- dc_p1 +
  geom_segment(data = loadings, aes(x = 0, y = 0, xend = cPC1, yend = cPC2),
               arrow = arrow(length = unit(0.03, "npc")),
               arrow.fill = "darkblue",  alpha = 0.95, ) +
  geom_text(data = dengue_data, aes(x = cPC1, y = cPC2, label = label),
            hjust = 1, vjust = -0.2, colour = "grey50",
            size = 4, check_overlap = FALSE)+
  geom_text(data = loadings, aes(x = cPC1, y = cPC2,
                                 label = rownames(loadings)),
            hjust = 1, vjust = -0.2, colour = "darkblue",
            size = 4, alpha = 0.55, check_overlap = FALSE)
dc_p3 <- dc_p2 +
  theme(axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill = NA, size = 1),
        axis.text = element_text(size = 18),
        axis.text.x = element_text(angle = 360, hjust = 1),
        axis.title = element_text(size = 23, face = "bold"),
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "white", colour = "white"),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))
dc_p3
```
```

## scPCA for Dengue Microarray Data

```{r scpca}
# perform scpca (takes a while to run, so load from saved file)
```
```

```
# see cluster_files folder
load(file = here("analyses/dengue_data/cluster_files/scpca.Rdata"))

# compute the average silhouette widths
scpca_group_sil <- silhouette(group_mem, dist(dengue_scpca$x))

# plot the 2D representation
scpca_df <- data.frame(
  scPC1 = dengue_scpca$x[, 1],
  scPC2 = dengue_scpca$x[, 2],
  class = dengue_class
) %>%
  dplyr::mutate(
    class = if_else(
      class == "convalescent",
      paste0(
        "Conv. (",
        sprintf("%.3f", round(summary(scpca_group_sil)$clus.avg.widths[1], 3)),
        ")"),
      if_else(
        class == "DF",
        paste0(
        "DF (",
        sprintf("%.3f", round(summary(scpca_group_sil)$clus.avg.widths[2], 3)),
        ")"),
        paste0("DHF (",
        sprintf("%.3f", round(summary(scpca_group_sil)$clus.avg.widths[3], 3)),
```

```
        ")")
      )
    )
  )

scpca_pd <- scpca_df %>%
  ggplot(aes(x = scPC1, y = scPC2, colour = class)) +
  geom_point(alpha = 0.9, size = 4.5) +
  ggtitle("scPCA on Dengue Microarray \n Data") +
  scale_colour_viridis_d(name = "Class",
                         end = 0.8, option = "plasma") +
  theme_bw() +
  theme(axis.text=element_text(size=18),
        axis.title=element_text(size=23,face="bold"),
        legend.text = element_text(size = 10.5),
        legend.title = element_text(size= 12),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))+
  coord_fixed()
scpca_pd

dengue_plot <-  (pca_pd + cpca_pd + scpca_pd) + plot_annotation(tag_levels = 'a')
ggsave("dengue_plot.png", plot = dengue_plot, width = 18, height = 3.5)
```

```

# bilpot for scPCA of microarray data

```{r}
```
```

```
dengue_scpca_loadings <- dengue_scpca$rotation # Loadings Matrix
dengue_scpca_scores <- dengue_scpca$x
colnames(dengue_scpca_scores) <- c("scPC1", "scPC2")
dengue_data <- cbind(label= paste(1:47),
              as.data.frame(dengue_scpca_scores[, c("scPC1", "scPC2")]))
normalize <- function(x) return ((x - min(x)) / (max(x) - min(x)))
dengue_data[, "scPC1"] <- scale(normalize(dengue_data[, "scPC1"]),
                                center = TRUE, scale = FALSE)
dengue_data[, "scPC2"] <- scale(normalize(dengue_data[, "scPC2"]),
                                center = TRUE, scale = FALSE)
loadings <- as.data.frame(dengue_scpca_loadings)
rownames(loadings) <- paste("V", 1:500)
colnames(loadings) <- c("scPC1", "scPC2")
dsc_p1 <- ggplot() +
  geom_point(data = dengue_data, mapping = aes(x = scPC1, y = scPC2),
             alpha = 0.1, color = "white") +
  ggtitle("scPCA Biplot")+
  coord_fixed()

dsc_p2 <- dsc_p1 +
  geom_segment(data = loadings, aes(x = 0, y = 0, xend = scPC1, yend = scPC2),
               arrow = arrow(length = unit(0.03, "npc")),
               arrow.fill = "darkblue",  alpha = 0.95, ) +
  geom_text(data = dengue_data, aes(x = scPC1, y = scPC2, label = label),
            hjust = 1, vjust = -0.2, colour = "grey50",
            size = 4, check_overlap = FALSE)+
  geom_text(data = loadings, aes(x = scPC1, y = scPC2,
```

```
                                label = rownames(loadings)),
              hjust = 1, vjust = -0.2, colour = "darkblue",
              size = 4, alpha = 0.55, check_overlap = FALSE)

dsc_p3 <- dsc_p2 +
  theme(axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill = NA, size = 1),
        axis.text = element_text(size = 18),
        axis.text.x = element_text(angle = 360, hjust = 1),
        axis.title = element_text(size = 23, face = "bold"),
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "white", colour = "white"),
        plot.title = element_text(size = 25, hjust = 0.5, face = "bold"))
dsc_p3
biplot_dengue <-  (dc_p3 + dsc_p3) + plot_annotation(tag_levels = 'a')
ggsave("biplot_dengue.png", plot = biplot_dengue, width = 18, height = 9)
```

Genes With Non-Zero Loadings

```{r map_genes}
# match the affymetrix code to the gene symbol
gene_df <- target %>% colnames %>% tibble()
colnames(gene_df) <- "affy_code"
gene_df <- gene_df %>%
  left_join(gene_names, by = c("affy_code" = "ID")) %>%
  select(affy_code, `Gene Symbol`, `Gene Title`)
```

```
# extract the gene symbols that have non-zero loadings in the scPC
rotation_df <- dengue_scpca$rotation %>%
  as_tibble %>%
  dplyr::mutate(
    affy_code = rownames(dengue_scpca$rotation)
  ) %>%
  dplyr::left_join(gene_df, by = "affy_code")
colnames(rotation_df) <- c("scPCA1", "scPCA2", "affy_code", "gene_sym",
                           "gene_title")
# print the table for the supp info
gene_scpca1 <- rotation_df %>%
  dplyr::filter(scPCA1 != 0) %>%
  select(gene_sym, gene_title, scPCA1) %>%
  xtable(
    caption = "Genes with non-zero loadings in first scPCA loadings vector.",
    digits = 4
  )
gene_scpca2 <- rotation_df %>%
  dplyr::filter(scPCA2 != 0) %>%
  select(gene_sym, gene_title, scPCA2) %>%
  xtable(
    caption = "Genes with non-zero loadings in second scPCA loadings vector.",
    digits = 4
  )
```

```{r}
```

```
# number of non-zero loadings for the first two cPCs - dengue data
nonzero_dengue <- data.frame(dengue_scpca$rotation)
colSums(nonzero_dengue != 0)
```

Comparison of cPCA and scPCA Loadings for Dengue Data

```{r comp_loadings, warning=FALSE, message=FALSE}
# compute the relative absolute loading ratios
cpca_loads_df <- dengue_cpca$rotation %>%
  as.data.frame() %>%
  dplyr::mutate(
    V1 = abs(V1),
    V2 = abs(V2),
    V1 = (V1-min(V1))/(max(V1)-min(V1)),
    V2 = (V2-min(V2))/(max(V2)-min(V2))
  )
scpca_loads_df <- dengue_scpca$rotation %>%
  as.data.frame() %>%
  dplyr::mutate(
    V1 = abs(V1),
    V2 = abs(V2),
    V1 = (V1-min(V1))/(max(V1)-min(V1)),
    V2 = (V2-min(V2))/(max(V2)-min(V2))
  )
# create the loadings comparison plot
load_diff_df <- bind_rows(
```

```
  cpca_loads_df,
  scpca_loads_df
) %>%
  dplyr::mutate(
    sparse = c(rep("0", ncol(target)), rep("1", ncol(target))),
    gene = rep(1:ncol(target), 2)
  )
# order the genes based on decreasing rel abs cpca loadings
ord_1_genes <- factor(load_diff_df$gene,
                      levels = sort(load_diff_df$V1[1:500], decreasing = TRUE,
                                    index.return = TRUE)$ix)
ord_2_genes <- factor(load_diff_df$gene,
                      levels = sort(load_diff_df$V2[1:500], decreasing = TRUE,
                                    index.return = TRUE)$ix)
load_diff_df <- load_diff_df %>%
  dplyr::mutate(
    gene1 = ord_1_genes,
    gene2 = ord_2_genes
  )
colnames(load_diff_df) <- c("comp1", "comp2", "sparse", "gene",
                            "gene1", "gene2")
p1 <- load_diff_df %>%
  ggplot(aes(y = comp1, x = gene1, fill = sparse)) +
    geom_bar(stat = "identity", position = position_identity(), width = 1) +
    xlab("Gene") +
    ylab("Standardized Abs.\n Loadings") +
    ggtitle("First Loadings Vector Comparison") +
```

```
    scale_fill_viridis_d(name = "Method",
                         labels = c("cPCA", "scPCA"),
                         end = 0.7,
                         option = "plasma") +
    scale_alpha_discrete(range = c(0.5, 1),
                         guide = FALSE) +
    theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
       plot.title = element_text(size = 25, hjust = 0.5, face = "bold"),
       legend.text = element_text(size = 15),
       legend.title = element_text(size=15),
       axis.title=element_text(size=23,face="bold"))


p2 <- load_diff_df %>%
  ggplot(aes(y = comp2, x = gene2, fill = sparse)) +
    geom_bar(stat = "identity", position = position_identity(), width = 1) +
    xlab("Genes") +
    ylab("Standardized Abs.\n Loadings") +
    ggtitle("Second Loadings Vector Comparison") +
    scale_fill_viridis_d(name = "Method",
                         labels = c("cPCA", "scPCA"),
                         end = 0.6,
                         option = "plasma") +
    scale_alpha_discrete(range = c(0.5, 1),
```

```
                          guide = TRUE) +
    theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
       plot.title = element_text(size = 25, hjust = 0.5, face = "bold"),
       legend.text = element_text(size = 15),
       legend.title = element_text(size=15),
       axis.title=element_text(size=23,face="bold"))


loadings_compd <- annotate_figure(
  ggarrange(p1, p2, nrow = 1, ncol = 2,
            common.legend = TRUE, legend = "right")
)
loadings_compd
ggsave("loadings_compd.png", plot = loadings_compd, width = 18, height = 9)
```