Capturing patterns of variation unique to a specific dataset

Robin Tu[1], Alexander H. Foss [2], and Sihai Dave Zhao[1]

[1]Department of Statistics, University of Illinois at Urbana-Champaign, Champaign, IL

[2]Statistical Sciences, Sandia National Laboratories, Albuquerque, NM

December 8, 2020

1 Introduction

Capturing patterns of variation present in a dataset is an important step in exploratory data analysis and unsupervised learning. Popular methods include principal component analysis (PCA) [10], nonnegative matrix factorization [13], projection pursuit [9], and independent component analysis [7]. Frequently, however, many of the identified patterns may actually arise from systematic or technical variation, for example batch effects, that are not of substantive interest.

New approaches are necessary for capturing meaningful patterns of variation. A popular recent approach is to contrast a target dataset of interest to a carefully chosen background dataset that represents unwanted or uninteresting variation. Patterns of variation unique to the target and not present in the background are more likely to be substantively meaningful. For example, in Section 2.2 we analyze images of the faces of actors expressing different emotions. Our goal is to identify features unique to the expression of disgust. However, the dominant modes of variation in the dataset correspond to general variation in facial features that are not meaningful for the emotion of interest. Instead, we contrast the data with a background dataset of images of other emotions that are not of interest. Patterns of variation unique to the target dataset reveal features specific to disgust.

This approach was introduced by Abid et al. [1], who proposed contrastive principal components analysis (cPCA). Whereas standard PCA identifies patterns that explain the most variation in the target dataset, cPCA seeks patterns that explain more variation in the target than in the background. The most important patterns are those corresponding to the largest gap between the two datasets. Salloum and Kuo [17] introduced cPCA++, which maximizes the ratio, rather than the difference, of the target and background explained variances. Boileau et al. [4] described sparse cPCA, which seeks maxmially contrastive patterns of variation that can be characterized using a parsimonious set of features. Other types of contrastive implementations include latent models [18] and autoencoders [2].

There are two main issues with existing contrastive methods. First, a major disadvantage is that they cannot accommodate multiple background datasets. Using multiple backgrounds allows researchers to better hone in on the unique variation of interest by removing multiple types or sources of unwanted variation. For example, in the emotion analysis in Section 2.2 where we uncover facial features characterizing the expression digust, the dataset contains background images from six other emotions. If we can simultaneously contrast the target data with multiple background emotions, such as anger and surprise, that are closely related to but still distinct from disgust, we will be able to identify more refined and distinctive patterns of variation. Naively applying existing methods by pooling the different emotions into a single background dataset is suboptimal, as variation in the pooled data may not be representative of variation in any of the individual datasets.

The second disadvantage of existing contrastive methods is that they typically require one or more tuning parameters. For example, cPCA requires the user to specify how much to penalize patterns that explain a large amount of variation in the background data. It is not clear in general how to choose these tuning parameters objectively.

We propose Unique Component Analysis (UCA), which addresses both of these issues. UCA can contrast a target dataset against multiple backgrounds and does not require any tuning parameters. It finds patterns that maximize the explained variation in the target under a constraint on the amount of variation they can explain in each of the backgrounds. With a single background, UCA is equivalent to cPCA but with an automatically selected tuning parameter. We show that UCA achieves similar results as cPCA and cPCA++ with a single background and that it can outperform them when using multiple backgrounds. We also develop computationally scalable algorithms for application to experiments with large numbers of measured features.

2 Results

2.1 Discovering subgroups in protein expression data

We first applied contrastive learning to mouse proteomics data, which were also used by Abid et al. [1] to illustrate the performace of cPCA. The study measured levels of 76 proteins in 1080 normal and trisomic (Down Syndrome) mice receiving various combinations of therapy (shock vs. no shock) and drugs (memantine vs. saline) [3, 12, 1]. The goal of the experiment was to assess whether memantine improved learning ability in trisomic mice and to identify subsets of protein that may be involved in this process.

We first replicate the analysis in Abid et al. [1]. Our goal is to extract patterns of variation in protein levels that can help discriminate normal from trisomic mice. Our target dataset consisted of protein data from unshocked mice given saline. However, natural variation, arising from factors such as age and gender, may dominant this dataset and obscure the variation of interest due to trisomy. To remove this natural variation, we contrasted the target data with a background dataset consisting of normal mice who had been given saline but had also been shocked. As natural variation is likely present in both the target and the background, patterns that explaining variation in the target but not the background may be more likely to related to trisomy.

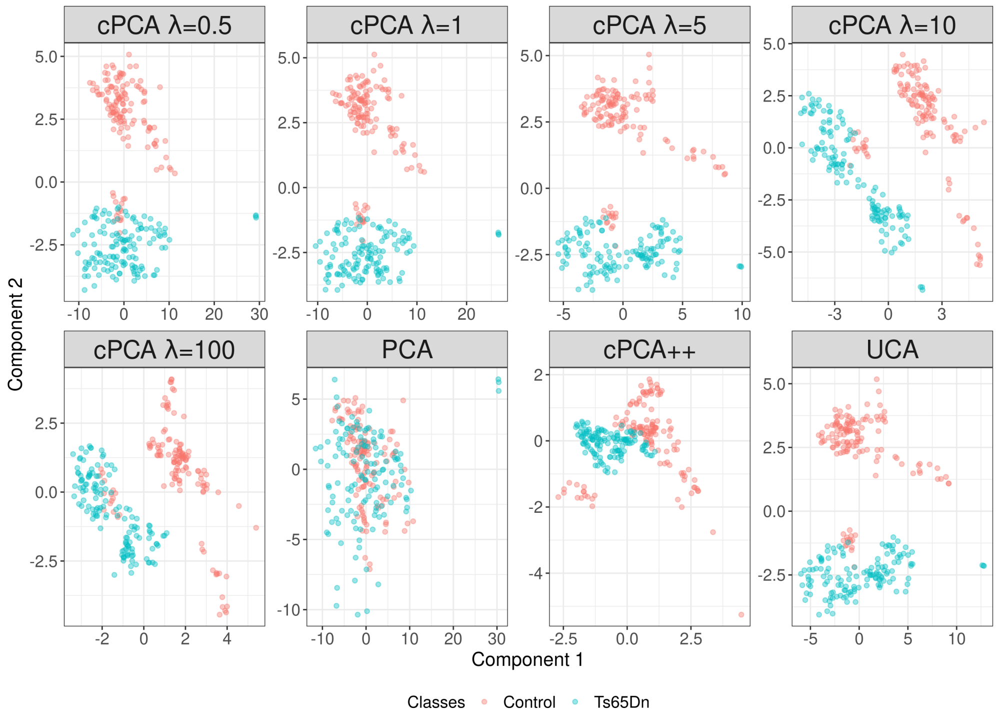

Figure 1: Mouse Protein Expression: Down Syndrome (Ts65Dn) Separability of Saline injected mice which were not subject to shock therapy.Confirming [1] results where Down Syndrome and control mice were unseparable using PCA alone but is easily separable by cPCA at all values of the contrastive parameterλ. The mice are also easily separable by cPCA++ and UCA.

Figure 1 shows the data projected to the first two components identified by PCA, cPCA, cPCA++, and UCA. Normal and trisomic mice are not well-separated in the PCA results, showing that dominant variation in the target data indeed does not stem from trisomy. In contrast, the two mouse groups are much more clearly separated in the cPCA results. While this separation is obvious for each of the tuning parameter values we tried, the actual projected data can vary considerably, and it remains unclear which tuning parameter to pick. cPCA++, which does not require specifying tuning parameter here because the number of samples exceeds the number of features, shows better separation than PCA but does not perform as well as cPCA. UCA performs as well as cPCA but without requiring a tuning parameter.

We next repeat the same analysis, but this time using shocked mice given saline as our target dataset. This was not considered by Abid et al. [1], and separating normal and trisomic shocked mice turns out to be a much more challenging problem. Figure 2 shows that normal and trisomic mice are again not well-separated by the first two components learned by PCA.

To remedy this, we contrast trisomic mice because of the additional gene expression variability due to the confounding effects of drug, trisomic gene, and shocking. We applied UCA using trisomic mice as background: unshocked trisomic mice injected with Memantine (Mem-S/C-Ts65Dn), shocked trisomic mice injected with Memantine (Mem-C/S-Ts65Dn),

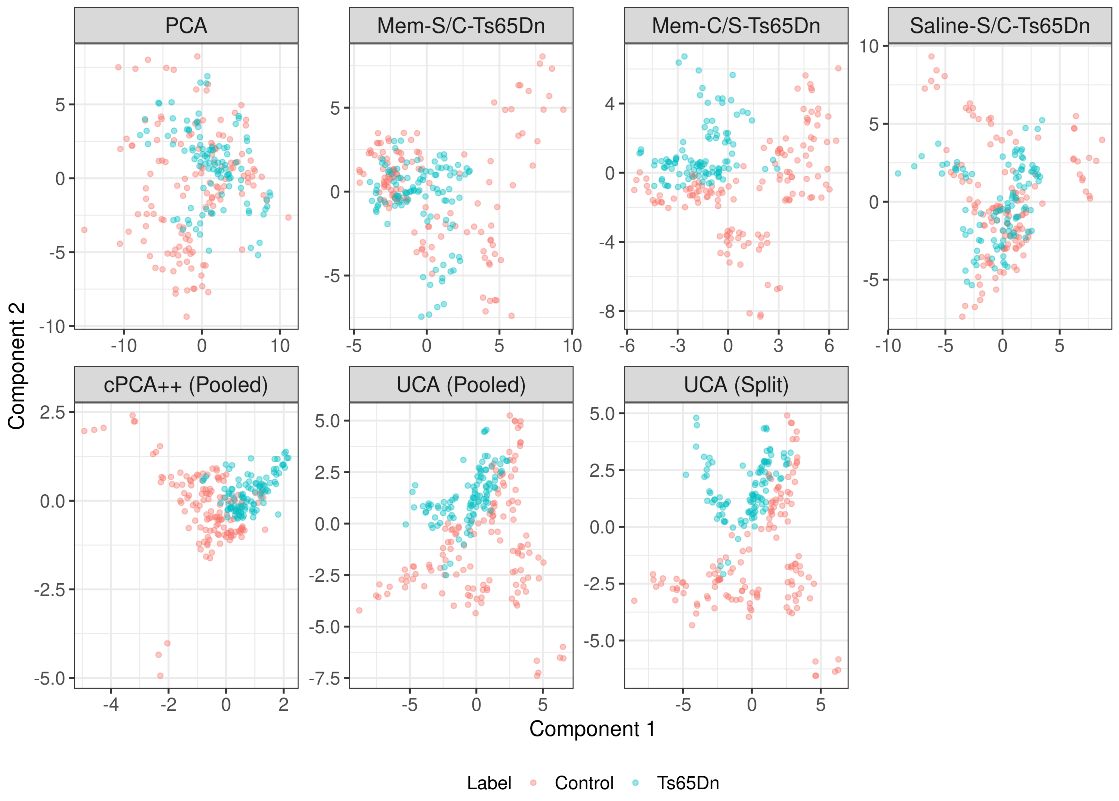

Figure 2: Mouse Protein Expression: Separability of normal and trisomic mice that were shocked injected with Saline. Separating normal and trisomic mice is more difficult in this case when only using a single background dataset. UCA with every combination of background (in parenthesis), which represents cPCA under the best case scenario, cannot separate between normal and trisomic mice. However, when combining information from all backgrounds via pooling (Pooled), UCA improves the separation. Note that cPCA++ has difficulty even when leveraging the same pooled background. When treating each individual background separately (Split), UCA improves on the pooled results.

and unshocked trisomic mice injected with Saline (Saline-S/C-Ts65Dn).

We see that UCA using unshocked trisomic mice as background does not separate normal and trisomic mice, likely due to overwhelming variability induced by shocking. When UCA uses shocked trisomic mice injected with Memantine, we better account for variability induced by shocking, but may not account for all the variability attributed to trisomic mutation and drug. By pooling the three individual backgrounds together, we use UCA to contrast the variability due to drug, shocking, and trisomic mutation, and achieve better separability between normal and trisomic mice compared to cPCA++, and to UCA using only a single background. However, pooling dilutes variability that is uniquely attirbuted to each background. Natively constrast multiple backgrounds simultaneously without pooling (Split) allows UCA to remove variability specific to unshocked trisomic mice treated with either Memantine or saline, and shocked trisomic mice treated with Memantine. This results in even better separability than UCA with pooled background.

2.2 Discovering eigenfaces of emotion

We exemplify advantages of the multiple background capabilities of UCA using the Karolinska Directed Emotional Faces (KDEF) [6], which captured images of 7 emotions, 5 different perspectives, from 70 amateur actors in either a practice or final session. Here we construct an example demonstrating that using UCA with multiple backgrounds is superior to using a single background. This is similar to a real world use-case of UCA where researchers may have targeted variation of interest, but also various control groups which may be used to remove known variability. For illustrative purposes, We focus on uncover variation unique to the Disgust emotion after contaminating the data with Surprise and Angry emotions from the final session. In Figure 3, we look at the top five eigenfaces produced by PCA, UCA with a single practice session background emotion of either Angry, Surprise, or an amalgamation of both Angry and Suprise (Stack), compared to cPCA++ with stacked background and UCA with dual practice session backgrounds of Angry and Suprised (Split).

PCA eigenfaces generally represent faces of all the emotions and does not highlighting features specifically to any individual emotion. UCA with only Surprise (SUS) practice faces as the background is dominated by features unique to Angry faces, whereas UCA with only Angry (ANS) practice faces as the background contains a mixture of surprise and disgust emotional features. Using both Surprise and Angry practice faces in the background in either Split or Stack better highlights emotional features of Disgust better than using a single background emotion. When UCA has both Surprise and Angry in the background (Split), we see that eyebrows and nasolabial folds, features unique to digust, are more pronounced compared to using the joint covariance of Surprise and Angry as the background, where features of disgust are pushed to the third eigenface. The second eigenface of Stack closely resembles the third eigenface of Split, and the third eigenface closely resembles the 2nd eigenface from split.

To better see the differences between the eigenfaces, we project the faces onto the first two components found and color the points by emotional class in figure 4. We can see that under the Split framework unique to UCA, we can actually separate each emotional face in the final session using the first two component, which cannot be accomplished when only using a single background, or combining multiple background data to produce a single joint covariance matrix.

Other face combinations show either similar or better separation when treating backgrounds separately, rather than stacking them to form a joint covariance matrix.

3 Discussion

In many data analytics settings we are interested in removing uninteresting variation that contaminate the data of interest. We propose an update to the original cPCA framework, UCA, by extending contrastive abilities to multiple background data. To do this, we first solved the tuning parameter problem which plague current methods. UCA does not require user input in selecting the best contrastive parameter, rather it works by maximizing the target data variability subject to keeping the variability in each background small using weak duality of Lagrangians. Furthermore, we reformulate the contrastive problem to avoid

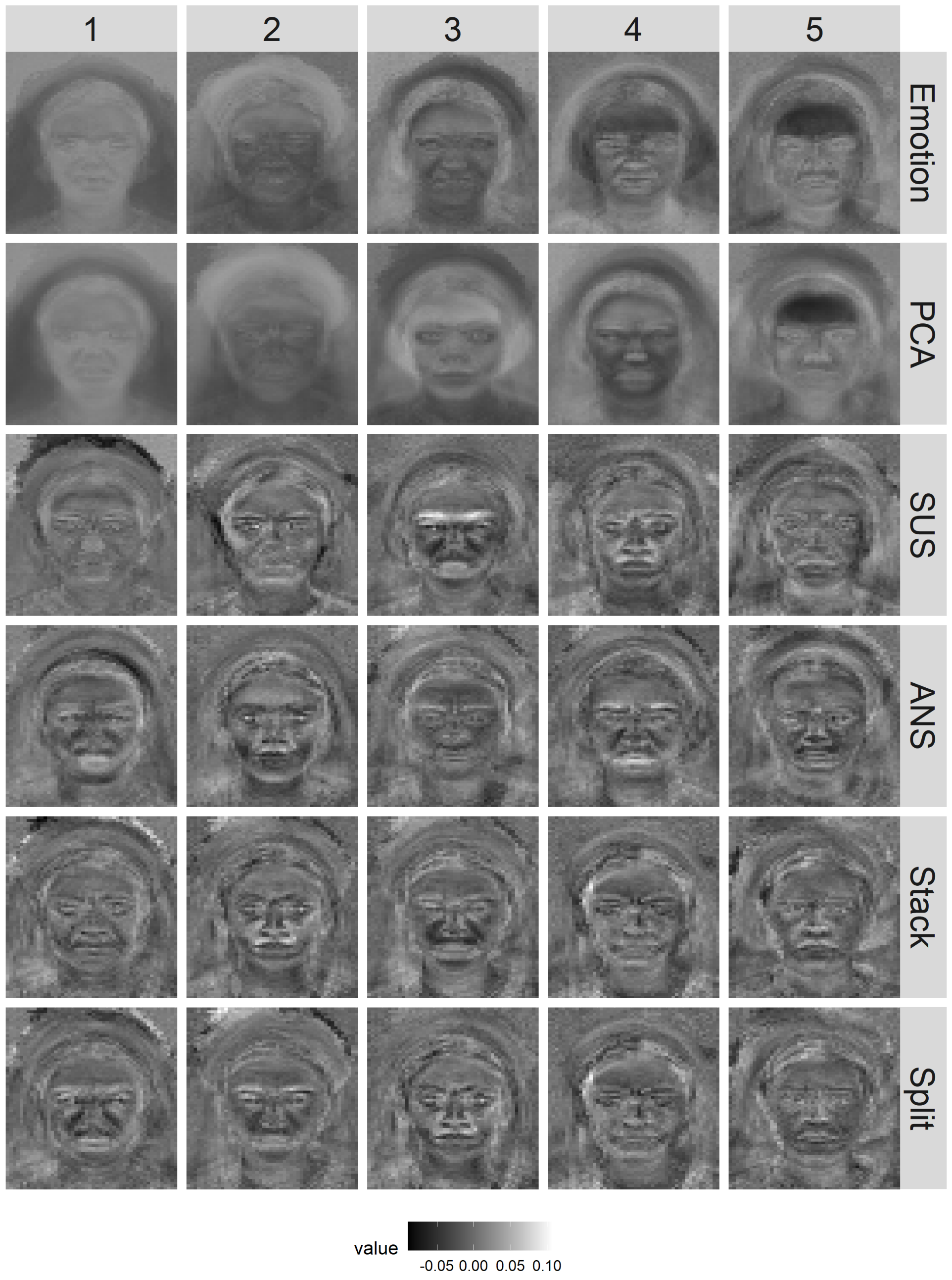

Figure 3: Eigenfaces of Surprise (SUS), Angry (ANS), and Disgust (DIS) emotions from the Female Actors from KDEF [6]. "Emotion" row contains final session DIS eigenfaces. PCA eigenfaces describe general emotional faces of SUS, ANS, and DIS, whereas UCA with only SUS background exhibits features of both ANS and DIS. UCA with only ANS as background exhibits features of both SUS and DIS. Combining backgrounds SUS and ANS in the Split manner better highlight the nasolabial fold and the eyebrows, features unique to DIS, but not as prominent in SUS or ANS. Combining SUS and ANS in the Stacking manner using UCA does not achieve similar results.

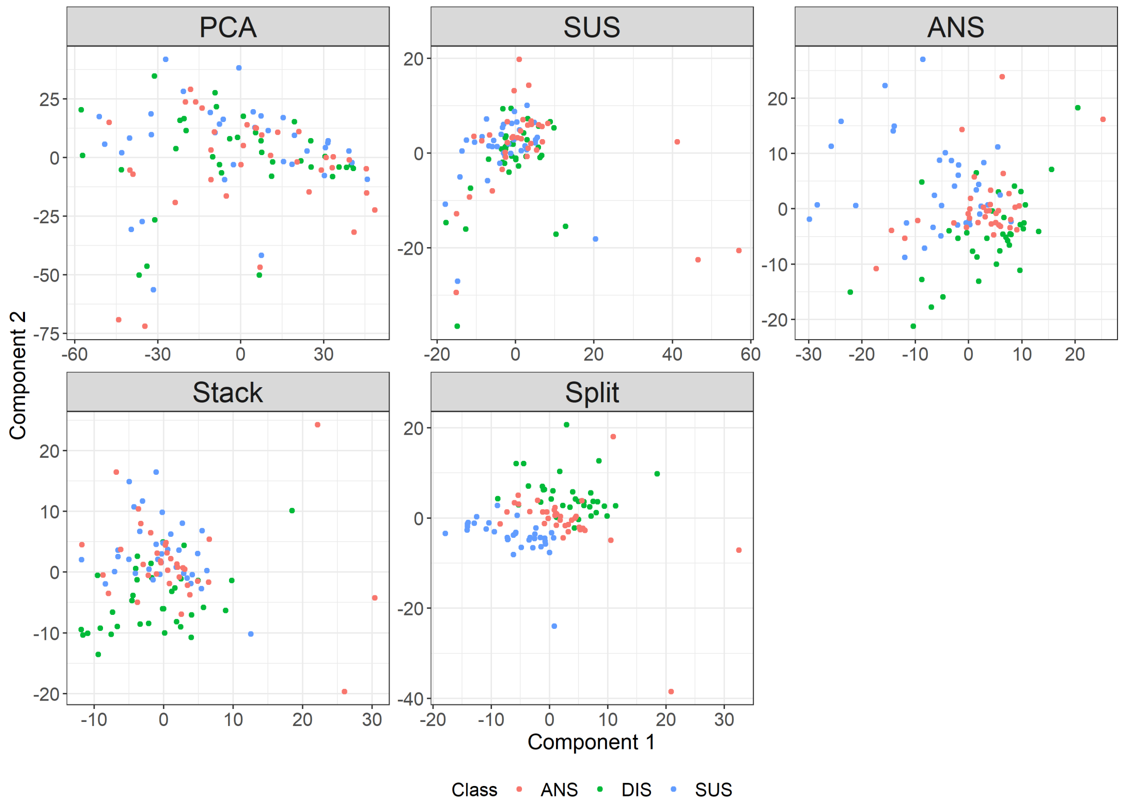

Figure 4: Eigenfaces projected onto the first two components found by PCA, cPCA++, UCA using Surprise (SUS) as the background, UCA using Angry (ANS) faces as the background, UCA using both Surprise and Angry as the background treated separately (Split), UCA using the joint covariance of Surprise and Split (Stack) as the background. UCA with Split background is capable of separating final session emotions whereas UCA with a single practice session background and both UCA and cPCA++ with stacked practice session backgrounds cannot produce the same separation.

creating several $p \times p$ covariance matrices, enabling us to perform contrastive analysis in high-dimensions.

We show that UCA is consistent with existing methods in the easily separable case within the mouse data. We then presented a less separable case with the mouse data, in which cPCA, cPCA++, and rPCA fail to do an adequate job in separating Down Syndrome and control mice for any choice of tuning parameter, where applicable. We showed that simply stacking individual background datasets can leverage multiple backgrounds to improve the single background separation between Down Syndrome and control mice. However, using the multiple background framework provided by UCA achieves even better results than stacking individual background datasets. Lastly, we demonstrate how UCA can be used to highlight emotions in eigenfaces by natively handling multiple backgrounds, which cannot be replicated by cPCA and UCA under the stacking framework.

Like cPCA, the choice of background still plays a pivotal role in the directions found by UCA. UCA only finds the optimal contrastive parameter using Lagrange Multipliers and does not remedy a poor background choice. We have released the code for UCA as an R package, along with documentation and examples.

4 Method

4.1 Contrastive principal components analysis

We first briefly review cPCA [1]. Let A denoted the $p \times p$ sample covariance matrix constucted from target data $X_{n_x \times p}$ and B denote the $p \times p$ sample background covariance constructed from background data $Y_{n_y \times p}$. The goal of cPCA is to find the directions of variability, $v \in \mathbb{R}^p$, which account for large variation in the target data Y and small variation in the background data X where $\|v\|_2 = 1$. Specifically, for some parameter λ , cPCA seeks the eigenvectors v solving

$$\arg\max_{v \in \mathbb{R}^p} \left(v^T A v - \lambda v^T B v\right) = \arg\max_{v \in \mathbb{R}^p} v^T \left(A - \lambda B\right) v.$$

For a given λ, the direction is simply the eigenvector of the weighted difference between covariance matrices where v maximize the variation in A while constraining variation in B.

The tuning parameter λ measures how much to penalize the background data covariance. When $\lambda = 0$, background variation isn't accounted for, thus the cPCA is reduced to PCA. As λ increases, the relative background variation becomes more dominant, causing v_λ to focus on directions which minimize background variation rather than maximizing the target. For very large values of λ, v_λ is equivalent to PCA after projecting the target data onto the nullspace of the background data. The authors of cPCA suggested that λ be chosen using spectral clustering via a parameter sweep of logarithmically spaced candidate values.

4.2 Unique Component Analysis

We introduce the Unique Component Analysis (UCA) framework to identify the optimal contrastive parameter, λ, and the resulting directions of variation using the duality of lagrangians. By adding the additional constraints of $v^T v = 1$ and $v^T B v = 1$ we arrive at an objective method to choose λ. These constraints are motivated by similar constraints in generalized eigenvalue problems, e.g. PCA where $Av = \lambda I v$ enforces $v^T v = 1$, cPCA++ where $Av = \lambda B v$ which enforces $v^T B v = 1$. Constraining $v^T v = 1$ while $v^T B v$ is unconstrained may result in eigenvectors that explain large amounts of variability in the target but also explain large amounts of variability in the background. However, constraining $v^T B v = 1$ while leaving $v^T v$ unconstrained results in eigenvectors that are not norm 1 and explain near zero variability in the target. UCA combines both of these constraints together and can be expressed as a primal optimization problem:

$$\max_{v \in \mathbb{R}^p} v^T A v \tag{1}$$
$$\text{subject to } \; v^T v = 1, \;\; v^T B v \leq 1$$

We can solve 1 using weak duality of Lagrangians, $\mathcal{L}$:

$$\begin{aligned}\max_{\lambda\geq 0}\max_{v\in\mathbb{R}^P}\mathcal{L}\left(v,\lambda\right) &= \max_{\lambda\geq 0}\max_{v\in\mathbb{R}^P}\left(v^TAv-\lambda\left(v^TBv-1\right)\right)\\ &= \max_{\lambda\geq 0}\max_{v\in\mathbb{R}^P}\left(v^T\left(A-\lambda B\right)v+\lambda\right)\\ &= \max_{\lambda\geq 0}\left(\lambda_{\max}\left(A-\lambda B\right)+\lambda\right) \qquad (2)\end{aligned}$$

where λ is the Lagrange Multiplier associated with the constraint $v^TBv \leq 1$ and $\lambda_{max}\left(\cdot\right)$ is the largest eigenvalue associated with $A-\lambda B$. Although we have two constraints, we get $v^Tv = I$ through eigendecomposition. This is slightly different from cPCA++, where v^Tv is not necessarily orthonormal.

To solve for the largest Lagrange Multiplier, λ we follow the standard method of taking the derivative of the Lagrangian $\mathcal{L}$ with respect to the Lagrange Multiplier λ and set the derivative equal to zero:

$$\begin{aligned}\frac{\partial}{\partial\lambda}\max_{v\in\mathbb{R}^P}\mathcal{L}\left(v,\lambda\right) &= \frac{\partial}{\partial\lambda}\left(\lambda_{max}\left(A-\lambda B\right)+\lambda\right)\\ &= \frac{\partial}{\partial\lambda}\left(v_\lambda^T\left(A-\lambda B\right)v_\lambda+\lambda\right)\\ &\Leftrightarrow \frac{\partial}{\partial\lambda}\left(-\lambda v_\lambda^TBv_\lambda+\lambda\right)\\ &= 1-v_\lambda^TBv_\lambda = 0 \qquad (3)\end{aligned}$$

where v_λ is the eigenvector associated with the largest eigenvalue for particular λ.

We use an iterative algorithm (L-BFGS-B) [5] to optimize between finding the Lagrange Multiplier λ, and finding the associated eigenvectors v_λ, stopping when equation 3 is satisfied.

With the optimal Lagrange Multiplier, eigenvector v_λ maximizes equation 1. Our constraints allows for a natural way to pick the contrastive parameter λ in cPCA. Similar to the constrastive parameter, the Lagrange Multiplier can't be less than zero, and allows us to use box constrained optimization methods.

Similarly, for multiple backgrounds, again let the A be the target $p \times p$ covariance matrix and $B_1, \ldots, B_m$ be the m background $p \times p$ covariance matrices constructed from a $n_y \times p$ dimensional Y data matrix and corresponding $(n_{x_1} \times p), \ldots, (n_{x_m} \times p)$ dimensional $X_1, \ldots, X_m$ background data matrices.

One way to incorporate m background would be to use the framework in the single background setting and stack multiple background datasets before calculate the background covariance matrix, creating a weighted average of the covariance matrices. This is suboptimal since the background covariance matrix with the most samples may not accounts for the all the background noise, therefore potentially washing out background noise from the other smaller background datasets.

A more natural way to incorporate m background datasets in a more generalized framework would be to append the background matrices like so:

$$\max_{v\in\mathbb{R}^p} v^T\left(A-\sum_{j=1}^{m}\lambda_jB_j\right)v$$

However, a parameter sweep for each λ_j corresponding to background data B_j would be computationally expensive. We can easily extend our single background covariance method to address the m background covariance scenario by adding additional constraints to Lagrangian, enforcing $v^T B_j v \leq 1$ for each m background covariance matrix. Specifically, the primal problem of our multiple background UCA is:

$$\begin{aligned} &\max_{v \in \mathbb{R}^p} v^T A v \\ \text{subject to } \; &v^T v = 1, \;\; v^T B_1 v \leq 1, \ldots, v^T B_m v \leq 1 \end{aligned} \tag{4}$$

Again, using weak duality of Lagrangians, our objective function now can be written as:

$$\begin{aligned} \max_{\lambda_1, \ldots, \lambda_m \geq 0} \max_{v \in \mathbb{R}^P} \mathcal{L}\left(v, \lambda_1, \ldots, \lambda_m\right) &= \max_{\lambda_1, \ldots, \lambda_m \geq 0} \max_{v \in \mathbb{R}^P} \left(v^T \left(A - \sum_{j=1}^{m} \lambda_j B_j \right) v + \sum_{j=1}^{m} \lambda_j \right) \\ &= \max_{\lambda_1, \ldots, \lambda_m \geq 0} \left(\lambda_{\max} \left(A - \sum_{j=1}^{m} \lambda_j B_j \right) + \sum_{j=1}^{m} \lambda_j \right) \end{aligned} \tag{5}$$

For background $j \in 1, ..., m$, we can solve for the jth Lagrange Multiplier in equation 5 similarly by taking the partial derivative and setting to zero:

$$\begin{aligned} \frac{\partial}{\partial \lambda_j} \left(\lambda_{\max} \left(A - \sum_{j=1}^{m} \lambda_j B_j \right) + \sum_{j=1}^{m} \lambda_j \right) &= \frac{\partial}{\partial \lambda_j} v_\lambda^T \left(A - \sum_{j=1}^{m} \lambda_j B_j \right) v_\lambda + \sum_{j=1}^{m} \lambda_j \\ &\Leftrightarrow \frac{\partial}{\partial \lambda_j} \left(v_\lambda^T \left(A - \lambda_j B_j \right) v_\lambda + \lambda_j \right) \\ &\Leftrightarrow \frac{\partial}{\partial \lambda_j} \left(-\lambda_j v_\lambda^T B_j v_\lambda + \lambda_j \right) \\ &= 1 - v_{\lambda_j}^T B_j v_{\lambda_j} = 0 \end{aligned} \tag{6}$$

To solve this multiple background dataset problem we propose a coordinate descent algorithm that utilizes the single background dataset L-BFGS-B [5] algorithm iteratively. For m background datasets, on the q^{th} iteration, let $A^*_{(j)}$ be the variation in A after excluding background variation that is not B_j:

$$A^*_{(j)} = A - \sum_{i=1}^{j-1} \lambda_i^{(q+1)} B_i - \sum_{i=j+1}^{m} \lambda_i^{(q)} B_i$$

to solve for j^{th} of the m Lagrange Multipliers, we use algorithm 1.

The advantage to the Lagrangian method in the multiple background setting is that rather than constructing an aggregated covariance matrix by stacking data, which arbitrarily weighs covariance matrices by their sample size, our method objectively constructs weights which satisfy the constraint $v^T B_j v \leq 1$ for each m background covariance matrix. If multiple backgrounds contain the same information, constructing the covariance matrix by stacking data will change the covariance by a factor less than 1, e.g. for two backgrounds, the covariance changes by $\frac{2}{2n-1}$. If only one covariance matrix will constrain the Lagrangian, the corresponding Lagrange Multiplier will be non-zero and Lagrange Multipliers corresponding to all other redundant constraints on the Lagrangian are set to zero.

Algorithm 1: coordinate descent algorithm

```
Inputs: $m$ centered background data $X_1, \dots, X_m$, centered Target data $Y$, $k$ number of components;
Initialize $\lambda_1^{(0)}, \dots, \lambda_m^{(0)}$, $\epsilon$;
Compute the emperical covariance matrices:
    $$A = \frac{1}{n_y} Y^T Y, \;\; B_1 = \frac{1}{n_{x_1}} X_1^T X_1, \; \dots, B_m = \frac{1}{n_{x_m}} X_m^T X_m$$
while $l^{(q+1)} - l^{(q)} > \epsilon$ do
    for $j = 1 : m$ do
        Solve for $\lambda_j^{(q+1)}$ via L-BFGS-B, corresponding to the largest eigenvalue of
            $$C_{\lambda_j} = A^*_{(j)} - \lambda_j^{(q+1)} B_j$$
    After iterating through $m$ Lagrange Multipliers, calculate value of Lagrangian
        $$l^{(q+1)} = \lambda_{max}\left(A - \sum_{j=1}^{m} \lambda_j^{(q+1)} B_j\right) + \sum_{j=1}^{m} \lambda_j^{(q+1)}$$
Compute
    $$C_{\lambda_1,\dots,\lambda_m} = A - \sum_{j=1}^{m} \lambda_j B_j$$
Find $V_k \in \mathbb{R}^k$, spanned by the top $k$ eigenvectors of $C_{\lambda_1,\dots,\lambda_k}$;
Return: $V_k$
```

4.3 Extension to High Dimensional Data:

To extend our method to high-dimensional data, we avoid the traditional eigendecomposition method of finding eigenvalue/eigenvectors of the covariance matrix. Just like how singular-value decomposition (SVD) of the data matrix can be used to find the top eigenvalue/eigenvectors of it's covariance, we introduce the Product SVD method to exploit the structure of how the contrastive covariance matrices are formed and employ SVD of a product of matrices to quickly find our top eigenvalue. Breaking the problem down in this way considerably speeds up our L-BFGS-B algorithm, and allows us to run UCA even in the high-dimensional setting.

For a single background, let the $p \times p$ covariance matrices A be the target covariance matrix and B be the background covariance matrix. A and B are formed from corresponding centered $n_y \times p$ target data matrix Y, and centered $n_x \times p$ background data matrix X. We can write the difference of the covariance matrices $A - \lambda B$ as a product of a $p \times (n_y + n_x)$ dimensional left matrix , L, and a $(n_y + n_x) \times p$ dimensional right matrix, R as seen in

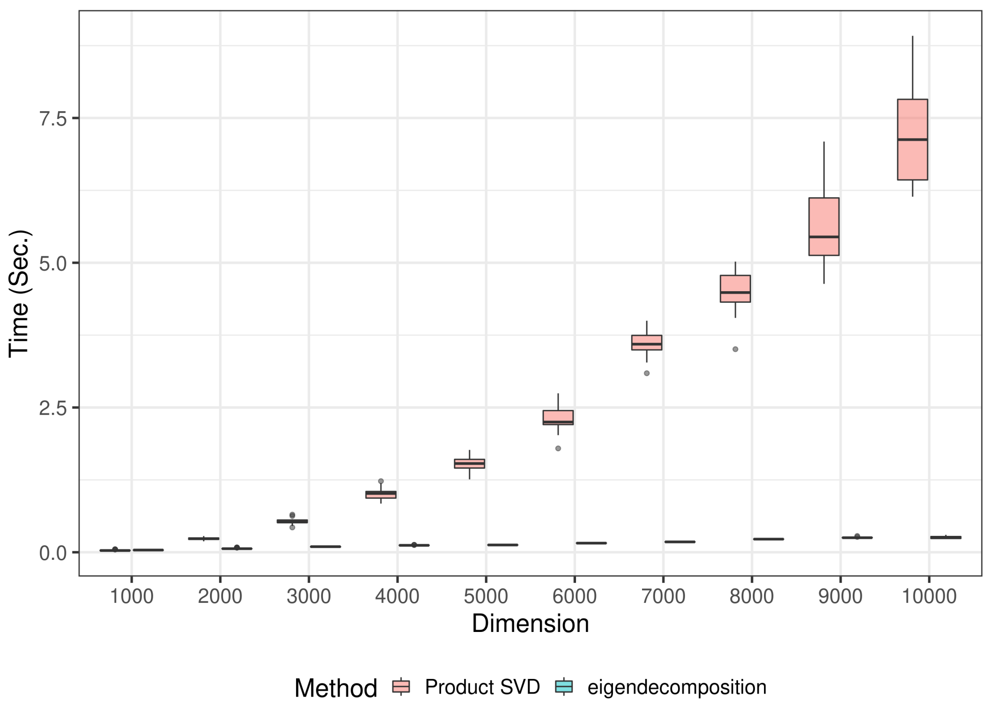

Figure 5: UCA and cPCA implementation using Product SVD vs. eigendecomposition for high-dimensional data: 25 random $100 \times p$ target and background matrices are generated from a standard normal distribution and where p, the dimension varied from 1,000 to 10,000 in steps of 1,000. Box plots of time (in seconds) is plotted for both eigendecomposition and the product SVD method. For small p, there is a negligible difference between the methods. However, as dimension p increases, the Product SVD is significantly faster.

equation 7:

$$\begin{aligned} C_\lambda &= A - \lambda B \\ &= \frac{1}{n_y} Y^T Y - \frac{\lambda}{n_x} X^T X \\ &= \underbrace{\left[\frac{1}{\sqrt{n_y}} Y^T, -\frac{\lambda}{\sqrt{n_x}} X^T \right]}_{L} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{n_y}} Y \\ \frac{1}{\sqrt{n_x}} X \end{bmatrix}}_{R} \end{aligned} \tag{7}$$

With this formulation, we can follow the steps in Golub et al. [11], and find the singular values and vectors of C_λ using a sequence of SVD and QR decompositions operating on the left and right matrices. Since singular values and eigenvalues coincide in square matrices, we use the singular vectors, U, to find the largest eigenvalues by sorting the diagonal of $ULRU^T$. We describe the Product SVD Method in algorithm 2, which can directly replace the more computationally expensive eigendecomposition in high dimensions.

Similarly, for multiple backgrounds, again let the A be the target $p \times p$ covariance matrix and $B_1, \ldots, B_m$ be the m background $p \times p$ covariance matrices constructed from a $n_y \times p$

```
Algorithm 2: Product SVD Method to calculate the largest Eigenvalue of $C_\lambda$
  Input: Centered background matrix $X$, centered target matrix $Y$, and $\lambda$;
1 Construct $L = \left[\frac{1}{\sqrt{n_y}}Y^T, -\frac{\lambda}{\sqrt{n_x}}X^T\right]$, $R = \begin{bmatrix}\frac{1}{\sqrt{n_y}}Y \\ \frac{1}{\sqrt{n_x}}X\end{bmatrix}$;
2 SVD the right matrix, $R = U_R S_R V_R^T$ ;
3 QR decompose $LU_R$ into $Q_{LU_R} R_{LU_R}$ ;
4 SVD the product of $R_{LU_R}$ (step 3) and $S_R$ (step 2), $R_{LU_R} S_R = EDF^T$ ;
5 The singular vectors of $C_\lambda$ is the product of $Q$ (step 3) and $E$ (step 4),
   $U_{C_\lambda} = Q_{LU_R} E$ ;
6 Find the largest eigenvalues of $C_\lambda$ by sorting the diagonal of $D_{C_\lambda}$, where
   $D_{C_\lambda} = U_{C_\lambda} C_\lambda U_{C_\lambda}^T$ ;
  Output: $\lambda_{\max}(C_\lambda)$
```

dimensional Y data matrix and corresponding $(n_{x_1} \times p), \ldots, (n_{x_m} \times p)$ dimensional $X_1, \ldots X_m$ background data matrices.

We can construct $C_{\lambda_1,\ldots,\lambda_m} = A - \sum_{j=1}^{m} \lambda_j B_j$ analogously by appending the additional datasets to the left and right matrices:

$$\begin{aligned} C_{\lambda_1,\ldots,\lambda_m} &= A - \sum_{j=1}^{m} \lambda_j B_j \\ &= \frac{1}{n_y} Y^T Y - \sum_{j=1}^{m} \frac{\lambda_j}{n_{x_j}} X_j^T X_j \\ &= \underbrace{\left[\frac{1}{\sqrt{n_y}} Y^T, -\frac{\lambda_1}{\sqrt{n_{x_1}}} X_1^T, \ldots, -\frac{\lambda_m}{\sqrt{n_{x_m}}} X_m^T\right]}_{L} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{n_y}} Y \\ \frac{1}{\sqrt{n_{x_1}}} X_1 \\ \vdots \\ \frac{1}{\sqrt{n_{x_m}}} X_m \end{bmatrix}}_{R} \end{aligned} \tag{8}$$

Using the coordinate descent (algorithm 1) to solve for each λ_j prescribed above, we can substitute the eigendecomposition of the covariance matrix, C_{λ_j} with the Product SVD method (algorithm 2) using L and R defined in equation 8.

The Product SVD method is advantageous in high-dimensions because it never explicitly operates on the entire $p \times p$ covariance matrix. Not only is our method more memory efficient, scaling with $n \times p$ bytes rather than p^2 bytes, but our method is also computationally more efficient at finding the largest eigenvalue/eigenvector as operating SVD and QR decomposition on either the left or right matrices is faster than directly operating eigendecomposition on the covariance matrix. In the single background scenario, λ only appears in L. Thus since, R doesn't change, we can pre-compute the SVD of R, and only update L when solving for $\lambda_{\max}$. Similarly, in the multi-background scenario, rather than modifying A^* in the original coordinate descent algorithm, this method allows us to only modify the $j+1$ element of the left data matrix, L. The SVD of the right matrix, R, only needs to be computed once in our coordinate descent. Furthermore, at each step within the coordinate descent only step 3 and 4, the SVD and QR, are computed, and are done on matrices with dimensions much smaller than $p \times p$.

To demonstrate the speed improvements of the Product SVD method compared to the current fastest implementations of eigendecomposition in high-dimensions, we conduct a simulation study with 25 sample $100 \times p$ target and background data matrices generated from a standard Normal distribution with p varying from 1.000 to 10,000 in steps of 1,000. To ensure a fair comparison, we leverage C++ in both implementations using a custom RcppArmadillo [8] function for the Product SVD method and the RSpectra package (0.16-0) [15] for the eigendecomposition method; RSpectra being a package designed for large-scale eigendecompositions based off the C++ Spectra library. We use the microbenchmark package [14] to ensure accurate timings. Our benchmark does not take into account the additional cost of forming the $p \times p$ covariance matrices, which would only exacerbate the difference between the two methods in real world applications.

Figure 4.3 show box plots of time (in seconds) versus the dimension, p, colored by method, summarizes the results of the simulation study. As dimension p increases, the computational time of our Product SVD method increases much slower than the current eigendecomposition implementations. It should be mentioned that for $p < 1000$ the Product SVD method is slower due to overhead of additional computation on small matrices. In general, for low-dimensional settings where $n \geq p$, the Product SVD will be negligibly slower because of the additional QR, SVD, matrix products, and sort computations.

All computations in this paper were done with R 3.6.3 [16] on an AMD Ryzen 1700X 3.7 gHz processor and 64GB 3000 mhz DDR4 RAM, utilizing the Intel MKL libraries.

4.4 Code Availability

We have released a R implementation of UCA on GitHub. We have implemented both Product SVD on the data matrix and eigendecomposition on the contrastive covariance matrix and allow the background to take either a single background or a list of backgrounds. The GitHub repository also includes R markdown and datasets that reproduce most of the figures in this paper and in the Supplementary.

4.5 Data availability

Datasets that have been used to evaluate UCA in this paper are publicly available from the authors of the original studies. You can download the mouse protein expression dataset from the UCI Machine Learning Repository [12] and the Karolinska Directed Emotional Faces (KDEF) from kdef website [6].

References

[1] Abubakar Abid, Martin J. Zhang, Vivek K. Bagaria, and James Zou. Exploring patterns enriched in a dataset with contrastive principal component analysis. *Nature Communications*, 9:2134, 05 2018.

[2] Abubakar Abid and James Zou. Contrastive variational autoencoder enhances salient features, 2019.

[3] Md. Mahiuddin Ahmed, A. Ranjitha Dhanasekaran, Aaron Block, Suhong Tong, Alberto C. S. Costa, Melissa Stasko, and Katheleen J. Gardiner. Protein dynamics associated with failed and rescued learning in the ts65dn mouse model of down syndrome. *PLOS ONE*, 10(3):1–25, 03 2015.

[4] Philippe Boileau, Nima S Hejazi, and Sandrine Dudoit. Exploring high-dimensional biological data with sparse contrastive principal component analysis. *Bioinformatics*, 36(11):3422–3430, 03 2020.

[5] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[6] Manuel G. Calvo and Daniel Lundqvist. Facial expressions of emotion (kdef): Identification under different display-duration conditions. *Behavior Research Methods*, 40(1):109–115, Feb 2008.

[7] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287 – 314, 1994. Higher Order Statistics.

[8] Dirk Eddelbuettel and Conrad Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014.

[9] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–890, 1974.

[10] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[11] Gene Golub, Knut Solna, and Paul Van Dooren. Computing the svd of a general matrix product/quotient. *SIAM Journal on Matrix Analysis and Applications*, 22(1):1–19, 2000.

[12] Clara Higuera, Katheleen J. Gardiner, and Krzysztof J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS ONE*, 10(6):1–28, 06 2015.

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

[14] Olaf Mersmann. *microbenchmark: Accurate Timing Functions*, 2019. R package version 1.4-7.

[15] Yixuan Qiu and Jiali Mei. *RSpectra: Solvers for Large-Scale Eigenvalue and SVD Problems*, 2019. R package version 0.16-0.

[16] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[17] Ronald Salloum and C.-C. Jay Kuo. Efficient image splicing localization via contrastive feature extraction. *CoRR*, abs/1901.07172, 2019.

[18] Kristen A Severson, Soumya Ghosh, and Kenney Ng. Unsupervised learning with contrastive latent variable models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4862–4869, 2019.

5 Appendix

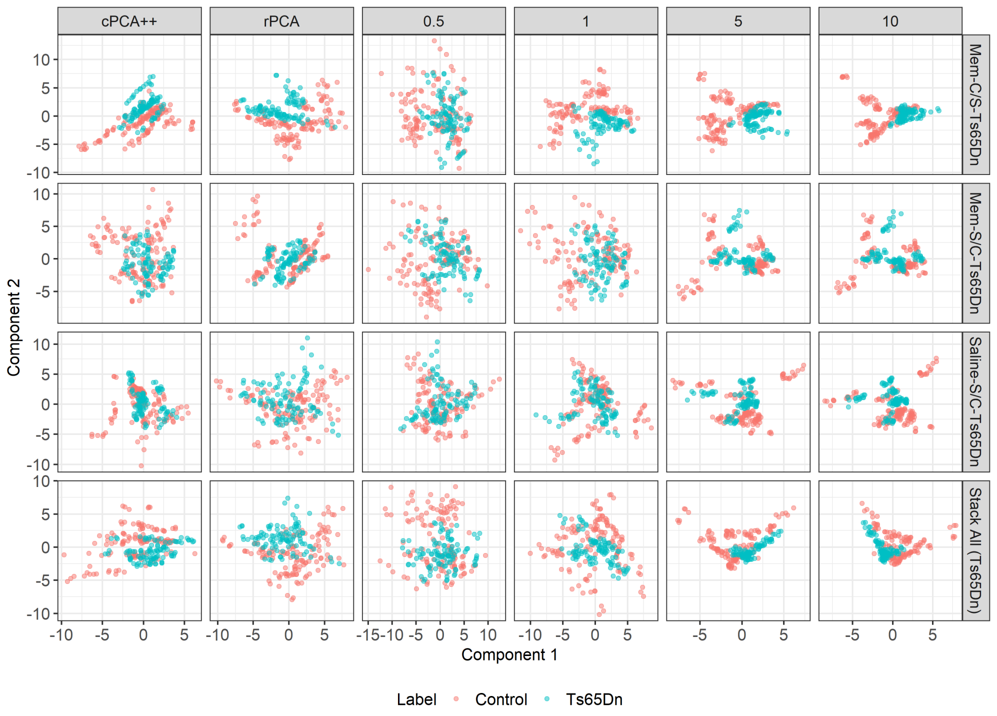

Figure 6: Mouse Protein Expression Dataset