A REVIEW OF MULTI-BLOCK DIMENSIONALITY REDUCTION VIA MULTIPLE CO-INERTIA ANALYSIS

A thesis

submitted by

Maximilian C. Mattessich

in partial fulfillment of the requirements

for the degree of

Master of Science

in

Mathematics

TUFTS UNIVERSITY

May 2022

© Copyright 2022 by Maximilian C. Mattessich

Adviser: Dr. Misha Kilmer

Abstract

Traditional dimensionality reduction approaches often struggle to create simultaneous low-dimensional embeddings for data with multiple heterogeneous blocks, which poses problems in fields that combine different types of data. Multi-block methods remedy this by highlighting linear relationships across datasets while also illustrating structures on the per-dataset level. We focus on creating an integrated analytical and computational representation for one such multi-block method called Multiple Co-Inertia Analysis (MCIA). We illustrate how MCIA can be computed using a modification to the Nonlinear Iterative Partial Least Squares (NIPALS) method, and benchmark the clustering performance of different data pre-processing choices for MCIA using synthetic datasets. Finally, we propose a pathway to extend MCIA to apply to data tensors.

Acknowledgements

I am deeply grateful for the support and mentorship from Dr. Misha Kilmer (Tufts University) and Dr. Anna Konstorum (Yale School of Medicine) during the research that led to this thesis, particularly in helping me understand the existing literature on MCIA. I am also grateful for help from my wider research group at Tufts, including Dr. Shuchin Aeron (Tufts University) and Shoaib Bin Masud (Tufts University). Finally, I would like to thank my thesis defense committee for their help in polishing my work, including Dr. Abiy Tasissa (Tufts University) as well as Drs. Kilmer, Konstorum, and Aeron.

Contents

List of Tables **vi**

List of Figures **vii**

1 Principal Component Analysis **1**

1.1 Theory of Principal Component Analysis 2

1.2 Computing PCA 4

1.2.1 Significance Measure of Principal Components 5

1.2.2 Linking PCA with the Singular Value Decomposition 6

1.3 Data Pre-processing and Standardization 7

1.4 Limitations of PCA - Towards Multi-Block Methods 8

1.4.1 Worked Example of PCA on Multi-Block Data 8

2 Multiple Co-Inertial Analysis **11**

2.1 Multi-Block Data 11

2.2 Data Normalization 12

2.2.1 Variable-Level Normalization 12

2.2.2 Block-Level Normalization 13

2.3 Mathematical Background 14

2.3.1 Defining Block and Global Scores 15

2.3.2 Interpreting the Scores and Loadings 17

2.4 Computing First Order Vectors 17

2.5 Deflation and Higher Order Solutions 18

2.6 Interpreting the Results of MCIA 19

2.6.1 MCIA in Clustering 19

2.6.2 MCIA in Variable Selection 21
2.6.3 Importance of Scores and Loadings in MCIA 22

3 Computing MCIA Through NIPALS **24**
3.1 Iterative Stage of NIPALS 24
3.2 Choice of Deflation Step and Consensus PCA 26
3.3 Eigensystem vs. NIPALS Approach 27
3.3.1 Block Score Weights 27

4 Benchmarking MCIA on Synthetic Data **29**
4.1 Parameters Under Investigation 29
4.1.1 Data Normalization 29
4.1.2 Deflation Step 30
4.1.3 Labeling of Variants 30
4.2 Benchmarking Approach 31
4.2.1 Generating Data 31
4.2.2 Measuring Clustering Performance 32
4.3 Results of Benchmarking 33

5 Conclusions **36**
5.1 Further Evaluation of MCIA Variants 36
5.2 Adapting MCIA to Tensors 37

A Table of Symbols and Notation **39**

B MATLAB Code **40**
B.1 Implementation of the NIPALS in MATLAB 40

Bibliography **47**

List of Tables

3.1 Steps for the iteration stage of the modified NIPALS approach to MCIA. Adapted from [5]. 25

4.1 Labels for variants of MCIA and CPCA benchmarked. 30

4.2 Parameters that describe the eight benchmarks used to evaluate clustering with MCIA and its variants. Replicated from Table 2 in [14]. . 32

A.1 Table of symbols and notation. 39

List of Figures

1.1 1.1a Plot of a synthetic dataset of 200 observations of two correlated variables with Gaussian distributions, together with vectors representing the first and second PC loadings in $\mathbb{R}^2$. 1.1b Plot of the first two PC scores of the same dataset, together with the PC loadings represented as vectors illustrating how the data has been projected. Note that the PC loadings vectors have been lengthened for visibility. 4

1.2 Plot of the first two PC scores (**top**) and proportion of variance explained by each of the first 10 PCs (**bottom**) as a result of performing PCA on a multi-omics dataset containing $12,985$ mRNA variables and 537 miRNA variables. The graphs correspond to performing PCA on the entire dataset (**left**), only the mRNA variables (**middle**), and only the miRNA variables (**right**). The miRNA-only data has a very different low-dimensional structure compared to the mRNA and combined data, which illustrates how performing PCA on multi-block data drowns out data blocks with smaller numbers of variables. 10

2.1 Plot of the first two ($j = 1, 2$) global and block scores obtained from MCIA performed on a multi-omics cancer dataset. The central points represent the global scores, while each shape represents the first two block scores for the three blocks (mRNA, miRNA, and Proteomics) in the dataset. The data are colored according to the known cancer types associated with each cell line. This plot was computed using the NIPALS implementation of MCIA in Appendix B to replicate Figure 2A in [12]. 20

2.2 Plot of the first two ($j = 1, 2$) block loadings obtained from MCIA performed on a multi-omics dataset. Points are labeled and colored according to the block they belong to, in this case referencing the omics type (mRNA, proteins, miRNA).This plot was computed using the NIPALS implementation of MCIA in Appendix B to replicate Figure 2B in [12]. 22

2.3 Plot of the proportion of variance explained by each global score from performing MCIA on the same cut version of the NCI-60 cancer cell line dataset used in Figures 2.1 and 2.2. 23

3.1 Plot of the block weights from MCIA computed via the NIPALS approach. Blocks that contribute more to the first/second global scores are plotted further in the horizontal and vertical directions respectively. 28

4.1 Box plots of Adjusted Rand Index values for clustering simulated data for 4 variants of MCIA and CPCA across 8 different benchmarks. Each benchmark is repeated 50 times to generate a range of ARI values. Values closer to 1 indicate a better recovery of the true clusters in the data. 35

Chapter 1

Principal Component Analysis

To introduce dimensionality reduction as a topic, we may begin with arguably the most popular dimensionality-reduction method. Interpreting high-dimensional data has been an area of active research for at least the last century, and has become more important as datasets have grown ever larger. While it is relatively easy to visualize data in two or three-dimensional space, it is much harder to visualize data in four or higher dimensional spaces. This is a particular issue in exploratory data analysis, where visualizing data is useful to understand particular trends and spot outliers. Beyond visualization, many of our analytical tools tend to break down if we apply them in higher dimensional spaces. For instance, the famous 'curse of dimensionality' renders distance-based data clustering increasingly useless as the dimension of the space increases [9]. One class of methods to solve this problem focus on transforming the data into a lower-dimensional space, allowing for more traditional data analysis techniques to work.

Fundamentally, a dataset consists of a set of samples (or observations) of a set of variables (or features). For example, a dataset tracking the weather around Boston might have temperature, humidity, and pressure as variables, sampled across various locations around the city. Another example could be a genomics dataset, with variables being the specific genes that are measured and samples referring to the patients from which these measurements were taken. We can format a dataset of n observations of p variables as a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, where

$$\mathbf{X} = \begin{pmatrix} \text{——} \vec{r}_1^T \text{——} \\ \vdots \\ \text{——} \vec{r}_n^T \text{——} \end{pmatrix} = \begin{pmatrix} | & & | \\ \vec{x}_1 & \cdots & \vec{x}_p \\ | & & | \end{pmatrix}. \tag{1.1}$$

Note that each row vector $\vec{r}_i^T \in \mathbb{R}^p$ represents the measurements of all variables for

a single sample, while each column $\vec{x}_j \in \mathbb{R}^n$ represents the measurements of one variable across all samples. The dimension of the dataset is p, which is the number of variables measured for each sample. Thus, dimensionality-reduction methods are usually concerned when the number of variables exceeds the number of samples - i.e. when $\mathbf{X}$ is a very wide matrix.

1.1 Theory of Principal Component Analysis

There are a variety of approaches to dimensionality-reduction, but one of the most fundamental is called Principal Component Analysis (PCA). In brief, PCA creates a set of 'optimal axes' based on the data, then projects the data onto these axes to create a lower-dimensional representation. Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ as above, the 'optimal axes' are linear combinations of the variable vectors ($\vec{x}_i$) in $\mathbb{R}^n$ with maximal variance [8]. In other words, PCA seeks constants $a_1, ..., a_p$ satisfying

$$\underset{a_1,\ldots,a_p}{\text{argmax}} \quad \text{var}\left(\sum_{i=1}^{p} a_i \vec{x}_i\right) \qquad \text{subject to} \quad \sum_{i=1}^{p} (a_i)^2 = 1 \tag{1.2}$$

where var is the variance function for a vector of observations $\vec{y}^T = \begin{pmatrix} y_1, & \cdots & , y_p \end{pmatrix} \in \mathbb{R}^p$ defined by

$$\text{var}(\vec{y}) = \frac{1}{p} \sum_{i=1}^{p} (y_i - \mu_{\vec{y}})^2 \qquad \text{where} \quad \mu_{\vec{y}} = \frac{1}{p} \sum_{i=1}^{p} (y_i). \tag{1.3}$$

We can use matrices and vectors to express (1.2) equivalently as

$$\underset{\mathbf{a} \in \mathbb{R}^p}{\text{argmax}} \quad \text{var}\,(\mathbf{Xa}) \qquad \text{subject to} \quad \mathbf{a}^T\mathbf{a} = 1 \tag{1.4}$$

where $\mathbf{a}^T = \begin{pmatrix} a_1 & \cdots & a_p \end{pmatrix} \in \mathbb{R}^p$ is the vector of coefficients in the linear combination. Note that the restriction $\mathbf{a}^T\mathbf{a} = 1$ on the length of $\mathbf{a}$ is necessary for the problem to be well-defined, as otherwise var$(\mathbf{Xa})$ can be scaled arbitrarily high. However, the exact sign and magnitude of $\mathbf{a}$ is not important in the results of PCA.

The solution to (1.4) can be thought of as defining the direction in $\mathbb{R}^p$ along which

the data has most variance. The vector $\mathbf{a}$ is called the first Principal Component (PC) **loading**. Recalling that each row of $\mathbf{X}$ corresponds to a sample in the dataset, the corresponding linear combination vector $\mathbf{f} = \mathbf{Xa}$ is a vector of coefficients from projecting each sample in $\mathbb{R}^p$ onto vector $\mathbf{a}$. Thus, $\mathbf{f}$ is a representation in a 1-dimensional coordinate system defined by the unit vector $\mathbf{a}$, and is called the first principal component of $\mathbf{X}$ or the first PC **score**.

To increase the number of dimensions in the low-dimensional representation of $\mathbf{X}$, PCA repeats the optimization process of (1.4) requiring each successive solution vector to be orthogonal to all previous solutions. Thus, if a J-dimensional representation of $\mathbf{X}$ is required (for $J < p$), PCA finds vectors $\mathbf{a}^{(1)}, ..., \mathbf{a}^{(J)}$ in $\mathbb{R}^p$ satisfying

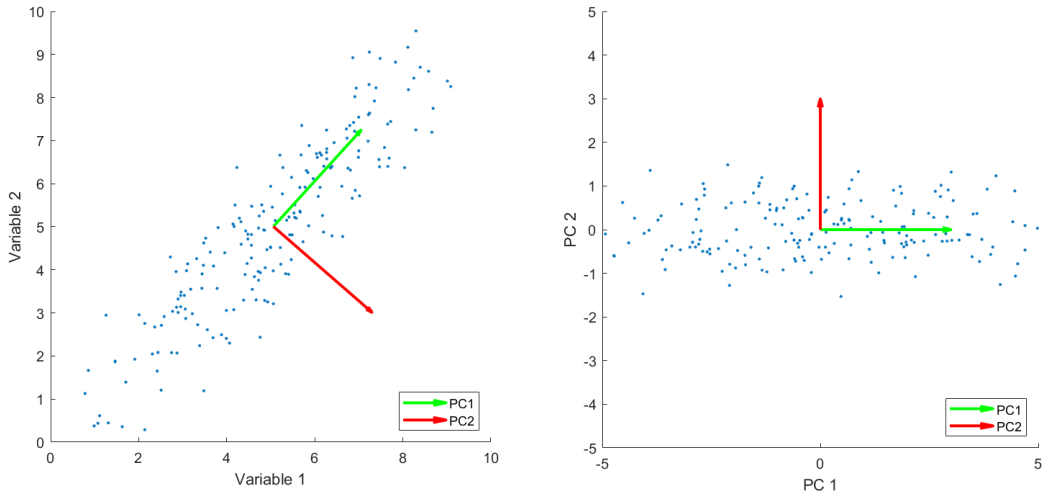
$$\underset{\mathbf{a}^{(j)} \in \mathbb{R}^p}{\operatorname{argmax}} \ \operatorname{var}\left(\mathbf{X}\mathbf{a}^{(j)}\right) \qquad \text{subject to} \quad \left(\mathbf{a}^{(j)}\right)^T \mathbf{a}^{(i)} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{1.5}$$

for $j = 1, ..., J$, $i = 1, ..., j$. The PC scores corresponding to this set of loadings are defined as before,

$$\mathbf{f}^{(j)} = \mathbf{X}\mathbf{a}^{(j)} \qquad\qquad j = 1, ..., J. \tag{1.6}$$

The orthogonality condition in (1.5) defines a natural ordering of the PC loadings and scores: $\mathbf{a}^{(1)}$ is the vector in $\mathbb{R}^p$ such that $\mathbf{f}^{(1)} = \mathbf{X}\mathbf{a}^{(1)}$ has maximal variance, while $\mathbf{f}^{(2)} = \mathbf{X}\mathbf{a}^{(2)}$ is the 'second best' maximal variance solution in the sense of respecting the orthogonality condition (and so on). Thus, we call $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \mathbf{a}^{(3)}$... and $\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \mathbf{f}^{(3)}$... the first, second, third (etc.) PC loadings and scores respectively. Given that $\{\mathbf{a}^{(1)}, ..., \mathbf{a}^{(J)}\}$ is an orthogonal set, their span defines a J-dimensional subspace of $\mathbb{R}^p$ where the score vectors $\mathbf{f}^{(1)}, ..., \mathbf{f}^{(J)}$ represent the projection coefficients of each row in $\mathbf{X}$ onto the basis vectors of this subspace [8]. Thus, PCA yields a J-dimensional representation of the data in $\mathbf{X}$.

Figure 1.1 presents a very simplified visualization of PCA on a synthetic dataset of 200 observations of two variables. Fig. 1.1a illustrates the geometric significance of the first PC loadings vector $\mathbf{a}^{(1)}$ as pointing in the direction where the variance of



(a) (b)

Figure 1.1: 1.1a Plot of a synthetic dataset of 200 observations of two correlated variables with Gaussian distributions, together with vectors representing the first and second PC loadings in $\mathbb{R}^2$. 1.1b Plot of the first two PC scores of the same dataset, together with the PC loadings represented as vectors illustrating how the data has been projected. Note that the PC loadings vectors have been lengthened for visibility.

the data is largest. Fig. 1.1b) illustrates a plot in the two-dimensional space spanned by $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$, using the score vectors $\mathbf{f}^{(1)}$ and $\mathbf{f}^{(2)}$ as lists of horizontal and vertical coordinates respectively. Note that since the data is in $\mathbb{R}^2$ and is projected onto the span of two vectors, there is no dimensionality reduction in this example.

1.2 Computing PCA

The optimization criterion in (1.5) is the defining feature of PCA. However, it is not immediately obvious how to actually compute its solutions. Given a dataset $\mathbf{X} \in \mathbb{R}^{n \times p}$ as above, we first define a matrix $\mathbf{S}$ called the 'sample covariance matrix' of $\mathbf{X}$ by

$$\mathbf{S} = \begin{pmatrix} \operatorname{cov}(\vec{x}_1, \vec{x}_1) & \operatorname{cov}(\vec{x}_1, \vec{x}_2) & \cdots & \operatorname{cov}(\vec{x}_1, \vec{x}_p) \\ \operatorname{cov}(\vec{x}_2, \vec{x}_1) & & & \vdots \\ \vdots & & & \vdots \\ \operatorname{cov}(\vec{x}_p, \vec{x}_1) & \cdots & \cdots & \operatorname{cov}(\vec{x}_p, \vec{x}_p) \end{pmatrix} \tag{1.7}$$

where $\vec{x}_i$ represents the i^{th} column of $\mathbf{X}$ and cov is the sample covariance function between two vectors $\vec{y}, \vec{z} \in \mathbb{R}^n$:

$$\text{cov}(\vec{y}, \vec{z}) = \frac{1}{n-1} \sum_{i=1}^{p} (y_i - \mu_{\vec{y}})(z_i - \mu_{\vec{z}}). \tag{1.8}$$

It can be shown [8] that we can write the variance of any linear combination $\mathbf{Xa}$ as

$$\text{var}(\mathbf{Xa}) = \mathbf{a}^T \mathbf{S} \mathbf{a}. \tag{1.9}$$

Thus (1.5) reduces to maximizing this quadratic form with the constraint that $\mathbf{a}^T \mathbf{a} = 1$. Using Lagrange multipliers (see [8]) it can also be shown that solutions maximizing (1.9) must satisfy the eigenvalue equation $\mathbf{Sa} = \lambda \mathbf{a}$. Combining this with (1.9), we can also see that

$$\text{var}(\mathbf{Xa}) = \mathbf{a}^T \mathbf{S} \mathbf{a} = \mathbf{a}^T (\lambda \mathbf{a}) = \lambda \tag{1.10}$$

for any unit-length eigenvector $\mathbf{a}$ of matrix $\mathbf{S}$. Furthermore, the construction of $\mathbf{S}$ shows it is a real symmetric matrix in $\mathbb{R}^{p \times p}$. Hence, by the spectral theorem for Hermitian matrices, the eigenvectors $\mathbf{a}^{(1)}, ..., \mathbf{a}^{(p)}$ of $\mathbf{S}$ form an orthogonal basis for $\mathbb{R}^p$. Since the variance of each linear combination $\mathbf{Xa}^{(i)}$ is simply the associated eigenvalue λ_i, finding the first J PC loadings of $\mathbf{X}$ is equivalent to finding the J unit-length eigenvectors of $\mathbf{S}$ with the largest-magnitude eigenvalues.

1.2.1 Significance Measure of Principal Components

In addition to simply providing a way to compute the PC loadings, the relation in (1.10) gives insight into how important each PC loading/score is. The total variance of associated with dataset $\mathbf{X}$ is defined as the sum of variances of each variable, i.e.

$$\text{var}(\mathbf{X}) = \sum_{i=1}^{p} \text{var}(\vec{x}_i) = \text{Tr}(\mathbf{S}). \tag{1.11}$$

Given that the trace of a matrix is equal to the sum of its eigenvalues, this becomes

$$\text{var}(\mathbf{X}) = \text{Tr}(\mathbf{S}) = \sum_{i=1}^{p} \lambda_i, \tag{1.12}$$

where $\lambda_1, ..., \lambda_p$ are the eigenvalues of $\mathbf{S}$. Thus, the proportion of the total dataset variance captured by projection onto the PC loading $\mathbf{a}^{(i)}$ is

$$\pi_i = \frac{\lambda_i}{\text{Tr}(\mathbf{S})} = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}. \tag{1.13}$$

This is a standard measure for judging the importance of a given PC loading [8] , and is used to pick the number of PC loadings needed to capture a given percentage of the total dataset variance.

1.2.2 Linking PCA with the Singular Value Decomposition

While it is possible to compute the PC loadings through the covariance matrix $\mathbf{S}$ as above, the links between PCA and the Singular Value Decomposition (SVD) become apparent if we first transform the data so that each variable has mean zero (i.e. the data is centered). For any entry x_{ij} of $\mathbf{X}$, we define the centered dataset $\mathbf{X}^* = [x^*_{ij}]$ where

$$x^*_{ij} = x_{ij} - \mu_{\bar{x}_j}. \tag{1.14}$$

It can then be shown using the definition of covariance ((1.8)) that the covariance matrix of the centered dataset $\mathbf{X}^*$ is

$$\mathbf{S}_{\mathbf{X}^*} = \frac{1}{n-1}(\mathbf{X}^*)^T\mathbf{X}^* \tag{1.15}$$

where n is the number of rows of $\mathbf{X} \in \mathbb{R}^{n \times p}$. If $\text{rank}(\mathbf{X}^*) = r$, we can compute the compact SVD of $\mathbf{X}^*$ to find

$$\mathbf{X}^* = \mathbf{U}\Sigma\mathbf{V}^T, \tag{1.16}$$

where $\mathbf{U} \in \mathbb{R}^{n\times r}$ and $\mathbf{V} \in \mathbb{R}^{p\times r}$ are orthonormal matrices and where $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal matrix of the singular values of $\mathbf{X}^*$. Combining this with (1.15), we find

$$(n-1)\mathbf{S}_{\mathbf{X}^*} = (\mathbf{X}^*)^T\mathbf{X}^* = (\mathbf{U}\Sigma\mathbf{V}^T)^T(\mathbf{U}\Sigma\mathbf{V}^T) = \mathbf{V}\Sigma^2\mathbf{V}^T. \tag{1.17}$$

Since the PC loadings of the centered dataset $\mathbf{X}^*$ are the eigenvectors of $(n-1)\mathbf{S}_{\mathbf{X}^*}$, they are also simply the right singular vectors of $\mathbf{X}^*$ with eigenvalues corresponding to the squared singular values of $\mathbf{X}^*$. Thus, computing the PC loadings for a centered data matrix is equivalent to computing the SVD, and the numerical methods developed to compute the latter can also be used for PCA.

1.3 Data Pre-processing and Standardization

As presented above, PCA is mathematically valid for any dataset. However, the dependence on covariance — a quantity that depends explicitly on the units each variable is expressed in — can pose a problem when a dataset contains variables with different length scales. Given dataset $\mathbf{X} = [x_{ij}] \in \mathbb{R}^{n\times p}$, this is rectified by performing PCA on the standardized dataset $\mathbf{Z} = [z_{ij}]$ such that

$$z_{ij} = \frac{x_{ij} - \mu_{\vec{x}_j}}{\sqrt{\mathrm{var}(\vec{x}_j)}} \tag{1.18}$$

where $\vec{x}_j$ are the columns of $\mathbf{X}$ (corresponding to variables). PCA can then be performed on the standardized matrix $\mathbf{Z}$, in what is commonly called 'correlation matrix PCA' [8]. Note that since the variables z_{ij} are centered, the SVD approach always works for correlation matrix PCA. It is important to note that correlation matrix PCA defines its score vectors $\mathbf{f}^{(i)}$ as linear combinations of the standardized variables $\vec{z}_j$, and thus the guarantees of variance explained in (1.13) are with respect only to the variance of the standardized variables.

1.4 Limitations of PCA - Towards Multi-Block Methods

An issue that is somewhat apparent in Section 1.3 is that PCA is not particularly suited to datasets incorporating multiple different variable types. While the standardization in (1.18) ensures each variable contributes equally to the total variance of the dataset, it doesn't adjust for differing numbers of variables of each type. This becomes an issue with datasets where there are large discrepancies in the types of each data collected. For instance, the field of 'multi-omics' in biology involves analyzing large biological datasets incorporating many different types of data, such as genomics, proteomics, transcriptomics (etc.) — each corresponding to a different 'omics'-type. There is often huge variation in the number of variables measured for each 'omics' type. Datasets may measure tens of thousands of genes, but only a few hundred micro-RNA expression levels (e.g. [10]). Even if variables are standardized as in Section 1.3, the micro-RNA variables will contribute little to the total variance of the dataset purely because there are many more variables related to genetics.

A related issue is the fact that, after applying PCA to a dataset, there is no clear way to check which types of data as a whole are most significant in generating the resulting low-dimensional representation. This is useful especially for exploratory data analysis where it is not known which variables are most useful or which variables follow the same low-dimensional structure. This capability is also useful to detect batch effects that might occur in measuring certain types of data [12].

1.4.1 Worked Example of PCA on Multi-Block Data

These concerns can be illustrated more concretely by applying PCA to a relevant dataset. Following the multi-omics example, we consider a dataset consisting of measurements of messenger RNA (mRNA) and microRNA (miRNA) on 21 cell lines from the NCI-60 cancer cell line database (adapted from the 'omicade4' R package [11]). The key feature of this dataset is a large discrepancy in number of variables — there are roughly 13,000 mRNA variables but only 537 miRNA variables. Thus,

we can represent the data as a matrix $\mathbf{X}$ as a concatenation of two matrices

$$\mathbf{X} = [\mathbf{X}_{\text{mRNA}} | \mathbf{X}_{\text{miRNA}}] \tag{1.19}$$

where $\mathbf{X}_{\text{mRNA}} \in \mathbb{R}^{21 \times 12,985}$ is the matrix composed of columns corresponding to the mRNA variables and $\mathbf{X}_{\text{miRNA}} \in \mathbb{R}^{21 \times 537}$ is the matrix composed of columns corresponding to the miRNA variables. Each of these sub-matrices is called a 'block', and thus $\mathbf{X}$ is called a 'multi-block dataset'.

After centering and normalizing each column as in (1.18), we can perform PCA on this dataset as before. Figure 1.2 shows the results of this analysis — namely with plots of projections onto the first two PC loadings (top) and graphs of the proportion of variance explained by the first 10 PCs (bottom). The leftmost graph corresponds to performing PCA on the whole dataset $\mathbf{X}$, whereas the middle and rightmost graphs represent the results of PCA on just the blocks $\mathbf{X}_{\text{mRNA}}$ and $\mathbf{X}_{\text{miRNA}}$ respectively.

The projections and variance graphs of Fig. 1.2 both illustrate that the mRNA and miRNA data possess different low-dimensional embeddings. Since PCA only considers the variance of the entire dataset, the larger number of mRNA variables effectively drowns out the miRNA variables in the combined PCA. This is evidenced by the similarity of PCA results on the combined dataset $\mathbf{X}$ and the mRNA-only dataset $\mathbf{X}_{\text{mRNA}}$. This motivates the need for an analysis method that can incorporate datasets that have multiple blocks of different sizes.

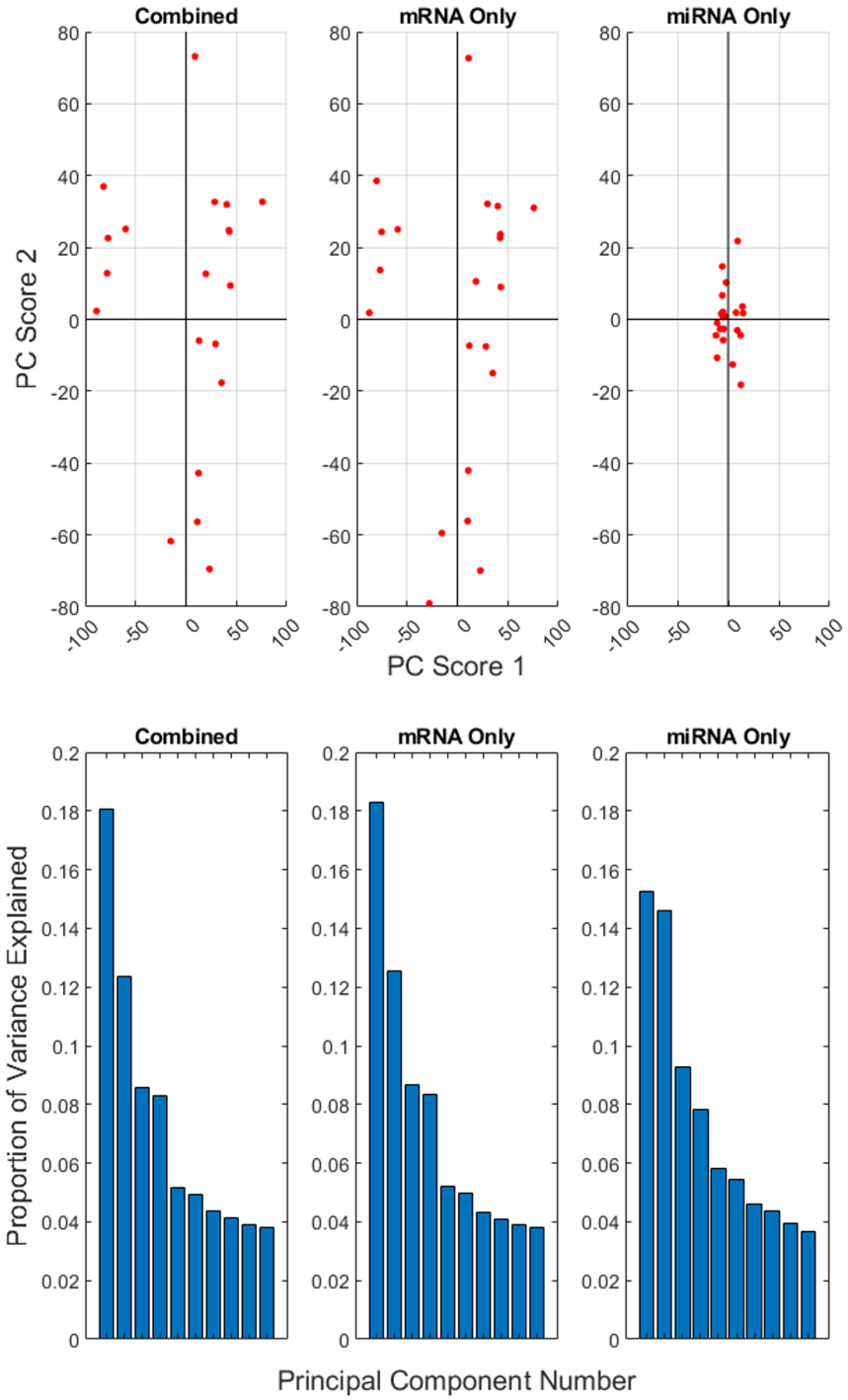


Figure 1.2: Plot of the first two PC scores (**top**) and proportion of variance explained by each of the first 10 PCs (**bottom**) as a result of performing PCA on a multi-omics dataset containing 12,985 mRNA variables and 537 miRNA variables. The graphs correspond to performing PCA on the entire dataset (**left**), only the mRNA variables (**middle**), and only the miRNA variables (**right**). The miRNA-only data has a very different low-dimensional structure compared to the mRNA and combined data, which illustrates how performing PCA on multi-block data drowns out data blocks with smaller numbers of variables.

Chapter 2

Multiple Co-Inertial Analysis

Multiple Co-Inertia Analysis (MCIA) is a dimensionality-reduction method designed for datasets with multiple blocks of different variables. It offers two key improvements over principal component analysis on these datasets. Firstly, through normalizing data on a block level, MCIA ensures each data block accounts for the same proportion of the total variance. This avoids the issue where smaller blocks of variables are drowned out by larger blocks in the low-dimensional representation (see Section 1.4.1).

The second improvement is, unlike PCA, MCIA generates two different types of score and loadings vectors. The first is a set of 'global' scores and loadings that describe patterns across the whole dataset, while the second are a set of 'local' or 'block' scores and loadings that describe patterns within each block of variables. Thus, MCIA is useful in illustrating block-level patterns and simultaneously analyzing multiple different types of data. It has been successfully applied in areas such as bioinformatics, particularly in analyzing multi-omics data [12].

2.1 Multi-Block Data

As with PCA, we need a more rigorous definition of 'multi-block data'. Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a dataset matrix of n observations of p variables. Furthermore, suppose the p variables of $\mathbf{X}$ can be grouped into N different blocks, so that X can be written as a concatenation of N matrices of dimension $n \times p_k$ each,

$$\mathbf{X} = \Big[\mathbf{X}_1 | \mathbf{X}_2 | ... | \mathbf{X}_N \Big].$$

Each matrix $\mathbf{X}_k$ is called a 'block' matrix in the multi-block structure of the 'super-matrix' $\mathbf{X}$. One example of such a structure can be seen in 'multi-omics' studies, where each block corresponds to variables of a different biological category (e.g. genes, proteins, RNA methylation). Another example could be a wine tasting dataset, where variables can be grouped according to whether they involve the taste, smell, or visuals of a given sample of wine (see Section 6 in [16], or [13]).

2.2 Data Normalization

Data pre-processing is essential in MCIA to ensure comparability among blocks, both due to different units across the variables as well as differing block sizes. Unlike for standard PCA, the normalization process for MCIA has two steps — variable-level normalization that is computed for each entry in the data matrix and block-level weights that are applied to each data block as a whole. As with other methods, there are many potential pre-processing choices to make.

2.2.1 Variable-Level Normalization

Variable-level normalization is applied to make sure each variable contributes equally to the low-dimensional representation in MCIA. It is therefore usual to divide each column by the square root of its variance in order to remove the effect of units of measurement. Much of the theory surrounding MCIA also relies on expressing the covariance matrix as $\mathbf{X}^T\mathbf{X}$. Given the discussion in Section 1.2, it is therefore necessary for the columns of the data matrix $\mathbf{X}$ to be centered.

Thus, one form of variable-level pre-processing is simply performing the column centering and scaling in (1.18), i.e.

$$x_{ij} \longmapsto \frac{x_{ij} - \mu_{\mathbf{x}_{:,j}}}{\sqrt{\text{var}(\mu_{\mathbf{x}_{:,j}})}} \tag{2.1}$$

where x_{ij} is the $(i,j)^{\text{th}}$ entry of $\mathbf{X}_k$ and $\mathbf{x}_{:,j}$ is the j^{th} column of $\mathbf{X}_k$. This ensures

that each column is centered with unit variance.

Another variation that occurs in literature and in popular implementations of MCIA (e.g. [11]) is to transform each block $\mathbf{X}_k = [x_{ij}]$ into a 'centered column profile' matrix. This is done by first offsetting the data so it is non-negative, then dividing any entry by the sum of its corresponding column before subtracting the row contribution to the total sum of the matrix, i.e.

$$x_{ij} \longmapsto \frac{x_{ij}}{\sum \mathbf{x}_{:,j}} - \frac{\sum \mathbf{x}_{i,:}}{\sum \mathbf{x}_{:,:}}, \tag{2.2}$$

where $\sum \mathbf{x}_{:,j}$ is the sum of the j^{th} column of $\mathbf{X}_k$, $\sum \mathbf{x}_{i,:}$ is the sum of the i^{th} row of $\mathbf{X}_k$, and $\sum \mathbf{x}_{:,:}$ is the sum of all entries in $\mathbf{X}_k$. Each column is then further scaled by multiplying each block $\mathbf{X}_k \in \mathbb{R}^{n \times p_k}$ by a matrix $\mathbf{Q}_k \in \mathbb{R}^{p_k \times p_k}$, i.e.

$$\mathbf{X}_k \longmapsto \mathbf{X}_k \mathbf{Q}_k \qquad \text{where} \qquad [\mathbf{Q}_k]_{j,j} = \sqrt{\frac{\sum \mathbf{x}_{:,j}}{\sum \mathbf{x}_{:,:}}}, \qquad j = 1, ..., p_k. \tag{2.3}$$

Note that $\mathbf{Q}_k$ is a diagonal matrix, hence the operation $\mathbf{X}_k \mathbf{Q}_k$ simply weights each column by the square root of its contribution to the total sum of $\mathbf{X}_k$. It can be verified computationally that this initialization leads to a column-centered matrix as with the 'standard' initialization'.

2.2.2 Block-Level Normalization

We now assume that we are given a dataset with multi-block structure $\mathbf{X} = [\mathbf{X}_1|...|\mathbf{X}_N]$, where $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{X}_k \in \mathbb{R}^{n \times p_k}$ for $k = 1, ..., N$. Block-level normalization is used to ensure each block contributes equally to the results of MCIA. As in section 1.4.1, variance-based criteria tend to favor data blocks with larger numbers of variables in a multi-block setting. Thus, most block-level pre-processing steps ensure that the total variance of each block is equal.

Given variable-level standardized as in (2.1), one strategy is to divide each block by the square root of its variance (e.g. in [16]). If $\mathbf{X}_k \in \mathbb{R}^{n \times p_k}$ is any block in the

multi-block structure, then

$$\begin{aligned} \mathrm{var}\left(\frac{1}{\sqrt{p_k}}\mathbf{X}_k\right) &= \frac{1}{p_k}\mathrm{var}(\mathbf{X}_k) \\ &= \frac{1}{p_k}\left(\sum_{j=1}^{p_k}\mathrm{var}(\vec{x}_j)\right) = \frac{1}{p_k}\cdot p_k = 1 \end{aligned} \tag{2.4}$$

where $\vec{x}_j$ is the j^{th} column of $\mathbf{X}_k$ and where the last line follows from the fact that $\mathrm{var}(\vec{x}_j) = 1$ for any variable normalized according to (2.1).

If each variable does not have unit variance (such as with (2.2)), we may instead simply divide by the square root of its total variance, since clearly

$$\mathrm{var}\left(\frac{1}{\sqrt{\mathrm{var}(\mathbf{X}_k)}}\mathbf{X}_k\right) = \frac{1}{\mathrm{var}(\mathbf{X}_k)}\mathrm{var}(\mathbf{X}_k) = 1. \tag{2.5}$$

The total variance of $\mathbf{X}_k$ can be computed from direct computation as above, or by the sum of the squared singular values of $\mathbf{X}_k$ as in (1.17).

Note that in either case, each block contributes one unit of variance to the total dataset. These block-level normalization factors are sometimes called 'block weightings', and are integrated into the global data matrix as

$$\mathbf{X} = [\omega_1^{1/2}\mathbf{X}_1|...|\omega_N^{1/2}\mathbf{X}_N] \tag{2.6}$$

where ω_i is the i^{th} block weighting.

2.3 Mathematical Background

The mathematical foundation for MCIA was originally developed using an object called a 'statistical triplet' [3]. This object is a 3-tuple of matrices

$$(\mathbf{X}_k, \mathbf{Q}_k, \mathbf{D}) \qquad \mathbf{X}_k \in \mathbb{R}^{n\times p_k}, \mathbf{Q}_k \in \mathbb{R}^{p_k\times p_k}, \mathbf{D} \in \mathbb{R}^{n\times n} \qquad k = 1, ..., N$$

where $\mathbf{X}_k$ is a data block matrix (post-normalization), $\mathbf{Q}_k$ is a positive symmetric matrix that defines weights in variable space ($\mathbb{R}^{p_k}$), and $\mathbf{D}$ is a positive symmetric

matrix that defines weightings in sample space ($\mathbb{R}^n$). The choice of $\mathbf{Q}_k$ and $\mathbf{D}$ allow for differing definitions of distance between points in variable space and individual/sample space respectively.

While this definition of MCIA is more flexible, it makes the parallels with PCA more cloudy. In most implementations of MCIA, it is typical to assume $\mathbf{D}$ is the $n \times n$ identity indicating equal weights across all samples. We have already seen a possible choice for $\mathbf{Q}_k$ in (2.3), which weights columns with higher variation more heavily. However, as the figures in the following sections will show, it is possible to replicate the results of MCIA while only using the matrices $\mathbf{Q}_k$ for data pre-processing. Thus, this illustration of MCIA will assume $\mathbf{D}$ and $\mathbf{Q}_k$ are identity matrices for all $k = 1, ..., N$, under the assumption that the choice for $\mathbf{Q}_k$ has already been applied during pre-processing. See [11] or [2] (in French) for a treatment of MCIA with statistical triplets.

2.3.1 Defining Block and Global Scores

As with PCA, given multi-block dataset $\mathbf{X} = [\mathbf{X}_1|...|\mathbf{X}_N]$, where $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{X}_k \in \mathbb{R}^{n \times p_k}$ for $k = 1, ..., N$, MCIA requires a target dimension $J \ll p$ for dimensionality-reduction. It then generates two sets of vectors analogous to the PC loadings and PC scores discussed previously. The first is a set of 'global' loadings and scores that describe trends across the whole dataset, represented by a set of vectors

$$\mathbf{f}^{(j)} = \mathbf{X}\mathbf{a}^{(j)} \qquad j = 1, 2, ..., J. \tag{2.7}$$

The vectors $\mathbf{a}^{(j)} \in \mathbb{R}^p$ represent a set of vectors in variable space, thus are similar to the PC loadings. The vectors $\mathbf{f}^{(j)}$ contain the projection coefficient of each row of $\mathbf{X}$ (i.e. each individual) onto the corresponding vector $\mathbf{a}^{(j)}$, and thus are the analog of the PC scores.

The second set of vectors are 'local' or 'block' scores and loadings specific to each

data block $\mathbf{X}_k$. These are represented by

$$\mathbf{f}_k^{(j)} = \mathbf{X}_k \mathbf{a}_k^{(j)} \qquad j = 1, 2, ..., J \qquad k = 1, 2, ..., N. \tag{2.8}$$

Again, the vectors $\mathbf{a}_k^{(j)}$ define a set of optimal axes in the variable space of each block ($\mathbb{R}^{p_k}$). The scores $\mathbf{f}_k^{(j)}$ again denote the projections of each individual onto the corresponding optimal axis as with the global scores.

The defining feature of MCIA is the optimization criterion it uses to create these sets of vectors. Similarly to PCA, MCIA finds global scores $\mathbf{f}$ and block loadings $\mathbf{a}_k$ that satisfy the covariance criterion

$$\underset{\mathbf{a}_1,...,\mathbf{a}_k,\mathbf{f}}{\operatorname{argmax}} \sum_{k=1}^{N} \operatorname{cov}^2(\mathbf{X}_k \mathbf{a}_k, \mathbf{f}) \tag{2.9}$$

Note that (2.9) is equivalent to finding axes such that the sum of squared covariances of the block scores ($\mathbf{f}_k = \mathbf{X}_k \mathbf{a}_k$) with the global score ($\mathbf{f}$) is maximized.

Given that (2.9) only involves a single set (i.e. first order) of global scores and block loadings, we introduce orthogonality constraints to generate higher orders. Specifically the s^{th}-order scores/loadings satisfy the orthogonality constraints

$$\begin{aligned} (\mathbf{f}^{(j)})^T \mathbf{f}^{(s)} &= 0 \qquad j = 1, ..., s-1 \\ (\mathbf{a}_k^{(j)})^T \mathbf{a}_k^{(s)} &= 0 \qquad j = 1, ..., s-1, \qquad k = 1, ..., N. \end{aligned} \tag{2.10}$$

Note that these constraints mean the global *scores* $\mathbf{f}^{(j)}$ are mutually orthogonal, while for the blocks it is the *loadings* $\mathbf{a}_k^{(j)}$ that are mutually orthogonal. In addition to (2.10), the results of MCIA are normalized to prevent arbitrary scaling

$$\operatorname{var}(\mathbf{f}) = 1, \qquad (\mathbf{a}_k)^T \mathbf{a}_k = 1, \qquad k = 1, ..., N \tag{2.11}$$

for any order of global score and block loadings.

2.3.2 Interpreting the Scores and Loadings

It is clear from the orthogonality criteria that for any given block $\mathbf{X}_k \in \mathbb{R}^{n \times p_k}$, the set of block loadings $\mathbf{a}_k^{(1)}, ..., \mathbf{a}_k^{(J)}$ form an ortho-normal basis for a J-dimensional subspace of the block's variable space $\mathbb{R}^{p_k}$. In effect, the block scores and loadings provide a local low-dimensional representation for the data in a block.

Although it is not clear from these condition, it is also true that the global loadings $\mathbf{a}^{(1)}, ..., \mathbf{a}^{(J)}$ are orthogonal (Table 2 in [6]). Thus the global loadings again represent coordinates vectors of a J-dimensional subspace of the global variable space $\mathbb{R}^p$. Hence, we may interpret (2.9) as generating a global low-dimensional representation along with block-level low-dimensional representations such that the block-level variation patterns match the corresponding global variation pattern.

2.4 Computing First Order Vectors

In order to satisfy both (2.9) and (2.10), MCIA computes block and global loadings iteratively. For any fixed vector $\mathbf{f} = \mathbf{X}\mathbf{a} \in \mathbb{R}^n$, it is a standard result from partial least squares (e.g. in [15]) that unit-normalized solutions to

$$\underset{\mathbf{a}_1,...,\mathbf{a}_k}{\operatorname{argmax}} \sum_{k=1}^{N} \operatorname{cov}^2(\mathbf{X}_k\mathbf{a}_k, \mathbf{f}) \tag{2.12}$$

are given by the vectors

$$\mathbf{a}_k = \frac{\mathbf{X}_k^T\mathbf{X}\mathbf{a}}{||\mathbf{X}_k^T\mathbf{X}\mathbf{a}||}. \tag{2.13}$$

Thus, substituting this into the covariance criterion,

$$\begin{aligned} \sum_{k=1}^{N} \operatorname{cov}^2(\mathbf{X}_k\mathbf{a}_k, \mathbf{X}\mathbf{a}) &= \frac{1}{n-1} \sum_{k=1}^{N} \left((\mathbf{X}_k\mathbf{a}_k)^T(\mathbf{X}\mathbf{a})\right)^2 \\ &= \frac{1}{n-1} \sum_{k=1}^{N} \left(\frac{(\mathbf{a}^T\mathbf{X}^T\mathbf{X}_k\mathbf{X}_k^T\mathbf{X}\mathbf{a})}{||\mathbf{X}_k^T\mathbf{X}\mathbf{a}||} \right)^2 \\ &= \mathbf{a}^T(\mathbf{X}^T\mathbf{X})^2\mathbf{a}. \end{aligned} \tag{2.15}$$

This is a quadratic form similar to (1.9) subject to the constraint $\mathbf{a}^T\mathbf{a} = 1$, and the same Lagrange multiplier argument as before shows $\mathbf{a}$ must satisfy the eigensystem

$$(\mathbf{X}^T\mathbf{X})^2\mathbf{a} = \lambda\mathbf{a}. \tag{2.16}$$

As before, the maximizer of (2.15) is the eigenvector corresponding to the largest eigenvalue of matrix $\mathbf{X}^T\mathbf{X}$. Thus, $\mathbf{a}$ is exactly the first PC loading of global data matrix $\mathbf{X}$ and $\mathbf{f} = \mathbf{X}\mathbf{a}$ is the corresponding first PC score.

Once $\mathbf{f}$ is computed, the block loadings can thus be found using the expression above,

$$\mathbf{a}_k = \frac{\mathbf{X}_k^T\mathbf{X}\mathbf{a}}{\|\mathbf{X}_k^T\mathbf{X}\mathbf{a}\|}, \qquad k = 1, ..., N. \tag{2.17}$$

Note that this definition automatically ensures the vectors $\mathbf{a}_k$ satisfy the length constraints in (2.11).

2.5 Deflation and Higher Order Solutions

Having computed the first order scores and loadings, subsequent steps of MCIA compute further solutions that also satisfy the orthogonality constraints in (2.10). This is achieved by removing the components of each data block $\mathbf{X}_k$ that are not orthogonal to the prior block loadings. Thus, to compute the j^{th} order loadings, we replace $\mathbf{X}_k$ with $\widetilde{\mathbf{X}}_k$ defined by

$$\widetilde{\mathbf{X}}_k = \mathbf{X}_k - \mathbf{X}_k\mathbf{a}_k^{(j-1)}\left(\mathbf{a}_k^{(j-1)}\right)^T \qquad k = 1, .., N. \tag{2.18}$$

Note that this step subtracts the data in block $\mathbf{X}_k$ in the direction of the prior-order block loading $\mathbf{a}_k^{(j-1)}$. This process is typically referred to as the 'deflation step'. We thus form the deflated global data matrix from the deflated blocks via $\widetilde{\mathbf{X}} = [\widetilde{\mathbf{X}}_1|...|\widetilde{\mathbf{X}}_N]$, and compute subsequent scores and loadings using the same process in section 2.4.

2.6 Interpreting the Results of MCIA

2.6.1 MCIA in Clustering

One of the primary uses of the results of MCIA is clustering on the low-dimensional representations it generates. Returning to the data matrix $\mathbf{X} = [\mathbf{X}_1|\mathbf{X}_2|...|\mathbf{X}_N] \in \mathbb{R}^{n \times p}$, recall that each row vector in $\mathbb{R}^p$ corresponds to an individual/sample in the dataset:

$$\mathbf{X} = \begin{pmatrix} — & \vec{r}_1^T & — \\ — & \vec{r}_2^T & — \\ & \vdots & \\ — & \vec{r}_n^T & — \end{pmatrix} \in \mathbb{R}^{n \times p}. \tag{2.19}$$

Thus, we can project each individual onto the coordinate system defined by the global axes $\mathbf{a}^{(j)}$ to find a low-dimensional representation of the data. Given (2.7), these projections are exactly the entries of the corresponding global scores, i.e.

$$\mathbf{f}^{(j)} = \mathbf{X}\mathbf{a}^{(j)} \qquad \longleftrightarrow \qquad \mathbf{f}^{(j)} = \begin{pmatrix} \vec{r}_1^T\mathbf{a}^{(j)} \\ \vec{r}_2^T\mathbf{a}^{(j)} \\ \vdots \\ \vec{r}_n^T\mathbf{a}^{(j)} \end{pmatrix} \qquad j = 1, 2, ..., J << p \tag{2.20}$$

where j is the order of the axis projected onto. Thus, the i^{th} entries of $\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, ..., \mathbf{f}^{(J)}$ defines a J dimensional representation of the i^{th} individual.

This low-dimensional representation of the data has similar uses to other dimensionality-reduction strategies. A key example of this is to aid in clustering individuals. The central points of each 'star' in Fig. 2.1 illustrate an example of plotting individuals by their components in the first two ($j = 1, 2$) global scores, thus highlighting three different clusters of individuals in the dataset.

MCIA's block scores can also be plotted as coordinates exactly like the global

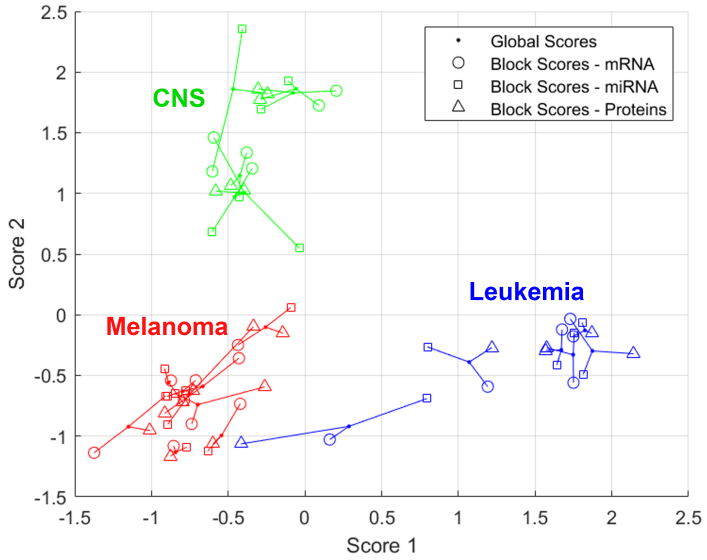


Figure 2.1: Plot of the first two ($j = 1, 2$) global and block scores obtained from MCIA performed on a multi-omics cancer dataset. The central points represent the global scores, while each shape represents the first two block scores for the three blocks (mRNA, miRNA, and Proteomics) in the dataset. The data are colored according to the known cancer types associated with each cell line. This plot was computed using the NIPALS implementation of MCIA in Appendix B to replicate Figure 2A in [12].

scores, since

$$\mathbf{f}_k^{(j)} = \mathbf{X}_k \mathbf{a}_k^{(j)} \qquad \longleftrightarrow \qquad \mathbf{f}_k^{(j)} = \begin{pmatrix} \vec{r}_1^T \mathbf{a}_k^{(j)} \\ \vec{r}_2^T \mathbf{a}_k^{(j)} \\ \vdots \\ \vec{r}_n^T \mathbf{a}_k^{(j)} \end{pmatrix} \qquad j = 1, ..., J << p. \tag{2.21}$$

Hence each entry of $\mathbf{f}_k^{(j)}$ defines a j^{th}-order coordinate for each sample in the dataset unique to data block k. Figure 2.1 illustrates plotting the first two ($j = 1, 2$) block scores as coordinates in individual space along with the global scores. Each of the three different block scores in this dataset is represented by a different shape (circle, square, triangle) linked to the central point representing the global score. This analysis shows how close the projections of each individual block are to the projections of the entire data matrix $\mathbf{X}$. If a block is projected far from the global coordinate or other block coordinates, it suggests the low-dimensional structure of that block doesn't follow the same trend as the rest of the dataset.

2.6.2 MCIA in Variable Selection

Another use for MCIA is in identifying the variables that contribute most to a given score. For any given block $\mathbf{X}_k \in \mathbb{R}^{n \times p_k}$, the j^{th}-order block loadings $\mathbf{a}_k^{(j)} \in \mathbb{R}^{p_k}$ define the corresponding block score $\mathbf{f}^{(j)} \in \mathbb{R}^n$ as a linear combination of variables (columns) of the block, i.e.

$$\mathbf{f}_k^{(j)} = \mathbf{X}_k \mathbf{a}_k^{(j)} \qquad <=> \qquad \mathbf{f}^{(j)} = \begin{pmatrix} | & & | \\ \vec{x}_{k1} & \cdots & \vec{x}_{kp_k} \\ | & & | \end{pmatrix} \cdot \begin{bmatrix} [\mathbf{a}_k^{(j)}]_1 \\ \vdots \\ [\mathbf{a}_k^{(j)}]_{p_k} \end{bmatrix} \tag{2.22}$$

$$<=> \qquad \mathbf{f}^{(j)} = \sum_{l=1}^{p_k} [\mathbf{a}_k^{(j)}]_l \cdot [\vec{x}_k]_l \tag{2.23}$$

where $\vec{x}_{ki}$ are the columns of block $\mathbf{X}_k$ and $[\mathbf{a}_k^{(j)}]_l$ represents the l^{th} entry of column vector $\mathbf{a}_k^{(j)}$. Hence, we can interpret the block loadings vectors $\mathbf{a}_k^{(j)}$ as lists of coordinates in variable space, and can thus plot the each variable in low-dimensional

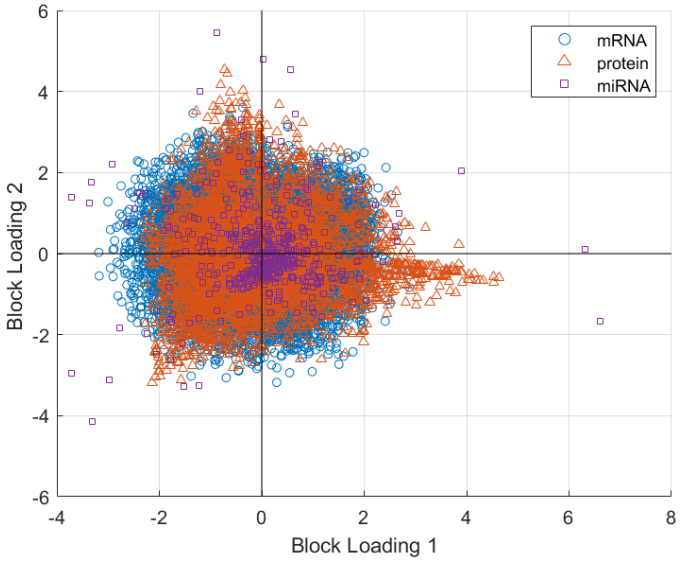


Figure 2.2: Plot of the first two ($j = 1, 2$) block loadings obtained from MCIA performed on a multi-omics dataset. Points are labeled and colored according to the block they belong to, in this case referencing the omics type (mRNA, proteins, miRNA).This plot was computed using the NIPALS implementation of MCIA in Appendix B to replicate Figure 2B in [12].

space. Figure 2.2 illustrates plotting the coordinates $([\mathbf{a}_k^{(1)}]_i, [\mathbf{a}_k^{(2)}]_i)$ for each variable indexed by i. Variables that strongly influence the first or second global scores are far from the origin on the horizontal and vertical axes respectively.

2.6.3 Importance of Scores and Loadings in MCIA

While the eigenvalues associated with each MCIA global score are not simultaneously computable like with PCA, the relation suggested by (2.16) gives us a measure of the variance explained by each component. If we let $\mathbf{X}^{(j)}$ be the deflated global data matrix at the $j^{(th)}$ order, then (2.16) tells us that the $j^{(th)}$ global score $\mathbf{f}^{(j)}$ is the first PC score of $\mathbf{X}^{(j)}$. Thus, its associated squared singular value $\lambda^{(j)}2$ represents the variance explained by this global score. Hence, given a set of global scores $\mathbf{f}^{(1)}, ..., \mathbf{f}^{(J)}$ and their associated singular values $\lambda^{(1)}, ..., \lambda^{(J)}$, we can plot the decline of the singular values as a bar chart much like in Fig. 1.2. Even further, if we compute all available PC scores (up to n if $\mathbf{X} \in \mathbb{R}^{n \times p}$ where $n \ll p$), then we can

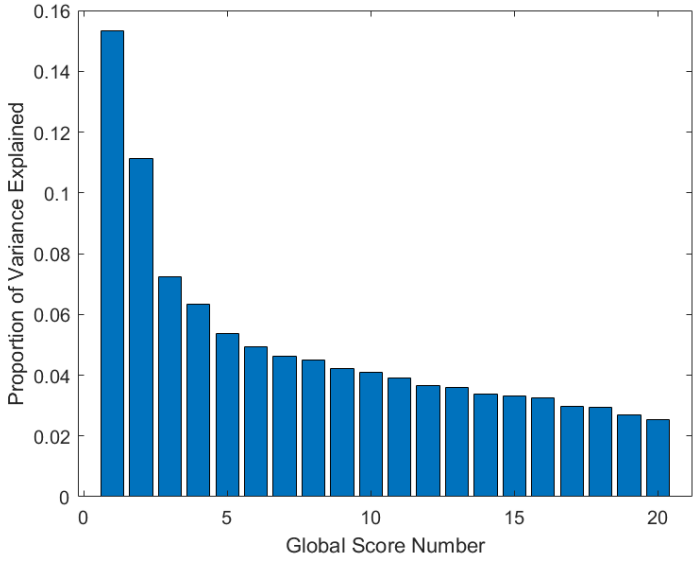


Figure 2.3: Plot of the proportion of variance explained by each global score from performing MCIA on the same cut version of the NCI-60 cancer cell line dataset used in Figures 2.1 and 2.2.

compute the proportion of variance explained by the score $f^{(k)}$ as

$$\pi_k = \frac{\lambda^{(k)}}{\text{var}(\mathbf{X})} = \frac{\lambda^{(k)}}{\sum_{j=1}^{n} \lambda^{(j)}}. \tag{2.24}$$

Figure 2.3 illustrates plotting π_k for the global scores of the cut NCI-60 multi-omics dataset.

Chapter 3

Computing MCIA Through NIPALS

While we have illustrated the standard eigensystem-based approach to computing MCIA often found in literature, this is not the only way to compute MCIA. More modern treatments of MCIA have suggested an alternative method through a modified version of the **N**onlinear **I**terative **PA**rtial **L**east **S**quares (NIPALS) family of methods (e.g. the supplementary materials of [12]). While some have illustrated how NIPALS can compute a related dimensionality reduction method (e.g. [5]), there has yet to be an explicit explanation of how MCIA can be computed with the method. Thus, this chapter illustrates computing MCIA through the modified NIPALS method, shows that it does indeed satisfy the objective function of MCIA, and illustrates some advantages it has over the standard eigensystem approach.

3.1 Iterative Stage of NIPALS

Like the eigensystem approach to MCIA discussed previously the modified NIPALS approach can also be divided into separate 'computation/iteration' and a 'deflation' stages. The deflation stage for modified NIPALS is exactly the same as in Section 2.5, but the computation stage is different.

Table 3.1 gives an overview of the iteration stage for MCIA via modified NIPALS. Given a multi-block data matrix $\mathbf{X} = [\mathbf{X}_1|...|\mathbf{X}_N] \in \mathbb{R}^{n \times p}$and a random starting vector $\mathbf{q} \in \mathbb{R}^{n \times 1}$, the columns of each block are projected onto $\mathbf{q}$ to form vectors of block coefficients $\mathbf{a}_k$ for $k = 1, ..., N$ we may call 'block loadings' (step 1). These coefficients then define N linear combinations $\mathbf{t}_k, k = 1, ..., N$ of the columns of each block matrix called the 'block scores' (step 3). After normalization, these N vectors can be organized into an $n \times N$ matrix (step 4). By projecting the block scores onto $\mathbf{q}$, we obtain a single $N \times 1$ vector of projection coefficients $\mathbf{w}$ called the block **weights** (steps 5 and 6). Finally, these block weights define a linear combination of

the block scores, which we then set as our new starting vector $\mathbf{q}$.

0.) Let $\mathbf{q} \in \mathbb{R}^{n\times 1}$ be an arbitrary starting vector
1.) Compute block loadings $\mathbf{a}_k = \frac{\mathbf{X}_k^T\mathbf{q}}{\mathbf{q}^T\mathbf{q}}$ for each block $k = 1, ..., N$
2.) Normalize the block loadings to $\mathbf{a}_k^T\mathbf{a}_k = 1$ for $k = 1, ..., N$
3.) Compute block scores $\mathbf{t}_k = \mathbf{X}_k\mathbf{a}_k$ for $k = 1, ..., N$
4.) Create an $n \times N$ matrix of block scores $\mathbf{T} = [\mathbf{t}_1 \vert ... \vert \mathbf{t}_N]$
5.) Compute global weight vectors $\mathbf{w} = \frac{\mathbf{T}^T\mathbf{q}}{\mathbf{q}^T\mathbf{q}}$
6.) Normalize the global weight vectors to $\mathbf{w}^T\mathbf{w} = 1$ for $k = 1, ..., N$
7.) Set starting vector to $\mathbf{q} = \mathbf{T}\mathbf{w}$ and return to step 1.)
Repeat until the global and block scores converge in the chosen convergence metric.

Table 3.1: Steps for the iteration stage of the modified NIPALS approach to MCIA. Adapted from [5].

Letting $\mathbf{t}_k^{(s)}$ and $\mathbf{q}^{(s)}$ denote the vectors $\mathbf{t}_k$ for $k = 1, ..., N$ and $\mathbf{q}$ after s iterations, it can be shown (Appendix A in [5]) that this process generates a monotonic sequence

$$\sum_{k=1}^{N} \mathrm{cov}^2\left(\mathbf{t}_k^{(s)}, \frac{\mathbf{q}^{(s)}}{\sqrt{\mathrm{var}(\mathbf{q}^{(s)})}}\right) \leq \sum_{k=1}^{N} \mathrm{cov}^2\left(\mathbf{t}_k^{(s+1)}, \frac{\mathbf{q}^{(s+1)}}{\sqrt{\mathrm{var}(\mathbf{q}^{(s+1)})}}\right). \tag{3.1}$$

Thus, at the limit $s \to \infty$, this process exactly maximizes the sum of squared covariances

$$\sum_{k=1}^{N} \mathrm{cov}^2\left(\mathbf{t}_k, \mathbf{q}\right) \qquad\qquad \mathrm{var}(\mathbf{q}) = 1 \tag{3.2}$$

which is identical to the MCIA optimization criterion in (2.9) if we identify $\mathbf{t}_k$ with the block scores $\mathbf{f}_k = \mathbf{X}_k\mathbf{a}_k$ and $\mathbf{q}$ with the global scores $\mathbf{f}$ in the limit. Thus, NIPALS as presented does indeed compute MCIA.

Furthermore, this insight provides us with an easy stopping criterion for the NIPALS algorithm. At iteration s, define the number

$$a^{(s)} = \sum_{k=1}^{N} \mathrm{cov}^2\left(\mathbf{t}_k^{(s)}, \frac{\mathbf{q}^{(s)}}{\sqrt{\mathrm{var}(\mathbf{q}^{(s)})}}\right) \tag{3.3}$$

then by (3.1) the sequence $a^{(1)}, a^{(2)}, ...$ is monotonic and is also bounded since the

variance of the dataset is finite. Thus, the sequence converges and a suitably small tolerance on the Cauchy criterion $a^{(s+1)}-a^{(s)}$ can determine when to halt the NIPALS process.

3.2 Choice of Deflation Step and Consensus PCA

To replicate MCIA fully, multiple orders of global and block scores must be generated as in (2.7) and (2.8). For MCIA, the deflation step is exactly identical to (2.18) after $\mathbf{a}_k$, $\mathbf{f}_k$, and $\mathbf{f}$ have been computed from the NIPALS iteration. Thus, after removing all data in the direction of the block loadings, we can apply the NIPALS iteration to the deflated data matrix exactly as before.

However, it is notable that this is not the only appropriate choice of deflation method for the NIPALS iteration. A related method called 'Consensus Principal Component Analysis' (CPCA, [5]) uses the same NIPALS iteration as above but deflates by the global **scores** instead of the block loadings, i.e.

$$\widetilde{\mathbf{X}}_k^{(j+1)} = \mathbf{X}_k^{(j)} - \frac{\mathbf{f}^{(j)}(\mathbf{f}^{(j)})^T}{(\mathbf{f}^{(j)})^T\mathbf{f}^{(j)}}\mathbf{X}_k^{(j)} \qquad k = 1, .., N \tag{3.4}$$

where $(f^{(j)})$ is the j^{th} order global score and $\widetilde{\mathbf{X}}_k^{(j)}$ is the j^{th} order deflated data block matrix. As before, these deflated blocks can be arranged into a deflated super-matrix $\widetilde{\mathbf{X}}^{(j+1)} = [\widetilde{\mathbf{X}}_1^{(j+1)}|...|\widetilde{\mathbf{X}}_N^{(j+1)}]$ to compute the $(j+1)$ order scores and loadings.

The low-dimensional representations of data generated from MCIA and CPCA can differ greatly, but the relative advantages of each method are not well-understood [6]. As a consequence of the new deflation method (3.4), the block loadings in CPCA are no longer assured to be orthogonal as with (2.18). However, there are some datasets where the choice to deflate by global scores yields a better separation of clusters in the low-dimensional representation (see Figures 3 and 4 in [6]).

3.3 Eigensystem vs. NIPALS Approach

While the parallels between the NIPALS iteration and the eigensystem approach may not be immediately obvious, it is important to note that both methods ultimately generate the same sets of block and global scores. We have shown in (3.2) that NIPALS does indeed satisfy the optimization criterion that characterizes MCIA, and this can qualitatively be seen by comparing the graphs in Figures 2.1 and 2.2 with Figure 2 in [12]. The former were generated using the author's implementation of NIPALS in `MATLAB` (see Appendix B), while the latter used the eigensystem approach via the popular `Omicade4` R package.

However, there are practical differences between the two approaches. The most obvious is the iterative nature of NIPALS, which yields the option to stop the calculation once results are 'good enough'. Moreover, the NIPALS stopping criterion is directly tied to the covariance-based MCIA optimization criterion in (2.9). This makes the stopping point for NIPALS easily interpretable in terms of the covariance between the block and global scores.

3.3.1 Block Score Weights

Another benefit of NIPALS is that it defines the global scores as a linear combination of the block scores. From step 7 in Table 3.1, at the end of the NIPALS iteration process we can express the global score $\mathbf{f}^{(j)}$ of order j as a linear combination of the block scores $\mathbf{f}_k^{(j)}$ of order j, i.e.

$$\mathbf{f}^{(j)} = \mathbf{T}^{(j)}\mathbf{w}^{(j)} \tag{3.5}$$

where

$$\mathbf{T}^{(j)} = \begin{pmatrix} | & & | \\ \mathbf{f}_1^{(j)} & \cdots & \mathbf{f}_N^{(j)} \\ | & & | \end{pmatrix} \tag{3.6}$$

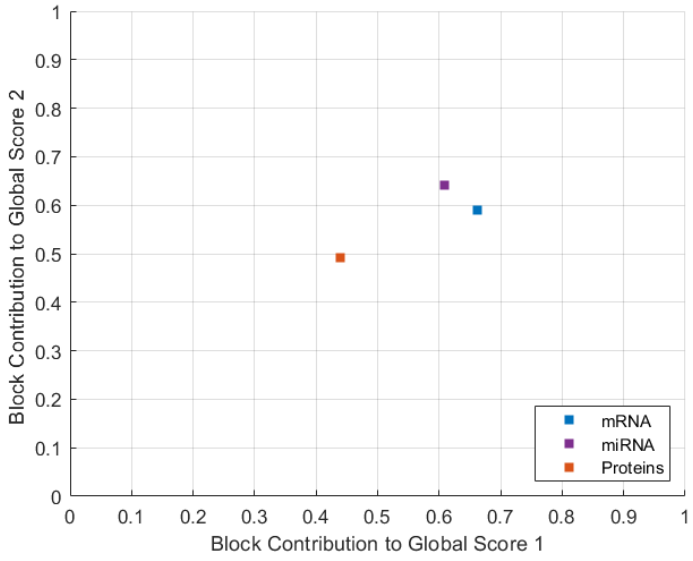


Figure 3.1: Plot of the block weights from MCIA computed via the NIPALS approach. Blocks that contribute more to the first/second global scores are plotted further in the horizontal and vertical directions respectively.

is the matrix of the j^{th} order block scores and $\mathbf{w}^{(j)}$ is a column vector of block score weights.

This gives us another insight into the contribution of each block to the global structure through the weight vector $\mathbf{w}^{(j)}$. Specifically, the coefficient represented by the k^{th} entry of $\mathbf{w}^{(j)}$ indicates how much the score for block k contributes to the global score for the relevant order j. We can again plot these coefficients as coordinates, illustrated in Fig. 3.1. Blocks that contribute most to the first order global score will thus be plotted farthest away from the origin in the horizontal direction, while block that contribute most to the second order global score are plotted farther away in the vertical direction.

Note that it is possible to compute the same linear combination $\mathbf{f} = \mathbf{Tw}$ using the eigensystem approach. Thus, Fig. 3.1 is not unique to NIPALS. However, the NIPALS approach requires no additional calculation as these weight vectors as they are found naturally in the iteration step.

Chapter 4

Benchmarking MCIA on Synthetic Data

Regardless of the method for computing MCIA, it is clear we have a number of choices to make when actually implementing the method. This ranges from the choice of block level initialization to the choice of deflation step (technically resulting in the related 'Consensus PCA' method). Although there have been attempts to compare at least the choice of deflation method (e.g. [6]), there has not yet been rigorous comparison of the different data pre-processing steps for MCIA. Thus, this chapter presents a basic benchmarking approach that can be expanded for more rigorous study of MCIA.

4.1 Parameters Under Investigation

4.1.1 Data Normalization

As discussed in Section 1.3, there are two common approaches to pre-processing data in MCIA. The first, illustrated by (2.1), is to standardize the variables in each block to have zero mean and unit variance. The blocks are then divided by the square root of the total number of variables (or equivalently the sum of the block's squared singular values) for block-level normalization. This is represented concisely by

$$x_{ij} \longmapsto \frac{x_{ij} - \mu_{\vec{x}_j}}{\sqrt{\text{var}(\vec{x}_j)}} \qquad \mathbf{X}_k \longmapsto \frac{1}{\sqrt{p_k}}\mathbf{X}_k \qquad k = 1, ..., N \tag{4.1}$$

where x_{ij} is the $(i,j)^{\text{th}}$ entry in the global dataset $\mathbf{X}$, and $\mathbf{X}_k \in \mathbb{R}^{n \times p_k}$ is the k^{th} data block in the multi-block structure. We will call this 'correlation' pre-processing since it is similar to the steps for correlation-matrix PCA.

The alternative approach, used in implementations such as the `Omicade4` R package, is to perform MCIA on centered column profiles as in (2.2). The block-level normalization then consists of dividing each block by the sum of its squared singular

values to ensure unit contribution to the total dataset variance. This is represented concisely as

$$(x_k)_{ij} \longmapsto \frac{(x_k)_{ij}}{\sum (x_k)_{:,j}} - \frac{\sum (x_k)_{i,:}}{\sum (x_k)_{:,:}} \qquad \mathbf{X}_k \longmapsto \mathbf{X}_k \mathbf{Q}_k \qquad (4.2)$$
$$\mathbf{X}_k \longmapsto \frac{1}{\sqrt{\Lambda_k}} \mathbf{X}_k \qquad k = 1, ..., N,$$

where now $(x_k)_{ij}$ refers to an entry specifically in the k^{th} data block, $\sum (x_k)_{:,j}$ is the sum of the j^{th} column of block k, $\sum (x_k)_{i,:}$ is the sum of the i^{th} row of block k, $\sum (x_k)_{:,:}$ is the sum of all entries in block k, and Λ_k is the sum of all squared singular values of $\mathbf{X}_k$. $\mathbf{Q}_k$ is the diagonal column-weighting matrix defined in (2.3). We call this 'column profile' pre-processing given that it converts the data entries to centered column profiles.

4.1.2 Deflation Step

In addition to the choice of normalization, we have a choice of methods in the deflation step that is applicable to both the eigensystem and NIPALS approaches presented. To compute MCIA, we can choose to deflate the data matrix block-by-block via the block loadings (2.18). Alternatively, we may choose to deflate with global scores (3.4) and thus compute CPCA as discussed above. This choice has been investigated previously, although it has not yet been explored when combined with changes in data pre-processing.

4.1.3 Labeling of Variants

The current options under investigation leave four 'variants' of MCIA and CPCA to benchmark. These variants are labeled according to table 4.1.

Method Name	**Data Pre-Processing**	**Deflation Method**
MCIA 1	Correlation	Block Loadings
MCIA 2	Column Profiles	Block Loadings
CPCA 1	Correlation	Global Scores
CPCA 2	Column Profiles	Global Scores

Table 4.1: Labels for variants of MCIA and CPCA benchmarked.

4.2 Benchmarking Approach

There are many different approaches and metrics that have been used to benchmark various aspects of dimensionality-reduction methods. An ideal investigation would apply a breadth of different methods to highlight the performance of the variants identified in table 4.1. However, for an initial analysis, we settle simply for benchmarking the ability of each variant to pick up clusters of data through their global score. To do this, we first generate artificial multi-omics data with known clusters, apply each variant to the data, cluster on the resulting embeddings, and compare with the 'known' clusters. While this does not involve one of the main selling points of MCIA — the block-scores — it does test a highly-important feature for an exploratory data analysis.

4.2.1 Generating Data

To generate enough multi-omics data with known clusters, we used an *R* package that creates synthetic datasets according to eight different benchmark profiles developed by [14]. In each benchmark, the global data matrix has three blocks each with data (and noise) generated according to a different distribution. Specifically,

$$\mathbf{X} = [\mathbf{X}_{\text{Gaussian}}|\mathbf{X}_{\text{Beta}}|\mathbf{X}_{\text{Binary}}] \tag{4.3}$$

where $\mathbf{X}_{\text{Gaussian}}$, $\mathbf{X}_{\text{Beta}}$, and $\mathbf{X}_{\text{Binary}}$ contain artificial clusters distributed by the Gaussian, Beta, and Bernoulli distributions respectively. Each of the eight different benchmarks changes the parameters of these distributions as well as the level of noise in the data, the number of clusters, the proportion of variables relevant to each cluster, and the size of each cluster. See Appendix A1.1 in [14] for a more thorough explanation of the dataset generation process. Table 4.2 is replicated from Table 2 in [14], and illustrates the differences between each of these benchmarks.

In keeping with the choices made by [14], we generate 50 simulations for each of

Benchmark	Data Type	Noise	% Relevant Variables	Variables per Cluster
1	Gaussian Binary Beta	0.2 0.01 0.3	0.5% 1% 2%	10, 20, 5 ,25
2	Gaussian Binary Beta	0.5 0.01 0.3	0.5% 1% 2%	10, 20, 5 ,25
3	Gaussian Binary Beta	0.1 0.05 0.3	0.5% 1% 2%	10, 20, 5 ,25
4	Gaussian Binary Beta	0.2 0.02 1.5	0.5% 1% 2%	10, 20, 5 ,25
5	Gaussian Binary Beta	0.2 0.01 0.3	1% 2% 4%	10, 20, 5 ,25
6	Gaussian Binary Beta	0.2 0.01 0.3	0.5% 1% 2%	25, 25
7	Gaussian Binary Beta	0.1 0.01 0.3	0.5% 1% 2%	25, 25, 25
8	Gaussian Binary Beta	0.1 0.01 0.3	0.5% 1% 2%	25, 25, 25, 25

Table 4.2: Parameters that describe the eight benchmarks used to evaluate clustering with MCIA and its variants. Replicated from Table 2 in [14].

the benchmarks above. For each simulation, we apply the variants under investigation to generate a two-dimensional embedding of the data via the global scores. We then run the default k-means clustering algorithm in `MATLAB` to compute clusters on the embeddings, which we compare with the known clusters.

4.2.2 Measuring Clustering Performance

To evaluate clustering performance, we use a metric called the 'Adjusted Rand Index' (ARI) to compare the computed clustering with the known clusters in the data. Briefly, given a set of n objects $S = \{O_1, ...O_n\}$, we may represent a clustering of the objects as a collection of disjoint subsets of S. For instance, if the objects in S are divided into L clusters, we would have the clustering $V = \{V_1, ..., V_L\}$ where $V_i \subset S$

for all $i = 1, ..., L$ and $V_i \cap V_j = \varnothing$ if $i \neq j$. For two clusterings $U = \{U_1, ..., U_R\}$ and $V = \{V_1, ..., V_L\}$ of the same set of n objects, the Adjusted Rand Index is calculated (see Eq. 5 in [7]) by

$$\mathrm{ARI}(U,V) = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i\cdot}}{2} \cdot \sum_j \binom{n_{\cdot j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}\right] - \left[\sum_i \binom{n_{i\cdot}}{2} \cdot \sum_j \binom{n_{\cdot j}}{2}\right] / \binom{n}{2}} \tag{4.4}$$

for $i = 1, ..., R$ and $j = 1, ..., L$, where $\binom{n}{2} = n(n-1)/2$, n_{ij} is the number of objects in $U_i \cap V_j$, $n_{i\cdot}$ is the size of cluster U_i, and $n_{\cdot j}$ is the size of cluster V_j.

It can be computed directly that $\mathrm{ARI}(U,V) = 1$ if clusterings U and V match (up to a change in label). This represents a version of the Rand index is 'corrected' (see [7] Section 2) so that the expectation of the ARI is zero if the objects are clustered randomly. Thus, while it is never above 1, there is no lower bound on the values that the ARI can take. Values closer to 1 indicate the two clusterings are more similar, while more negative values indicate that the clusterings are more dissimilar than if one of the clusters was generated randomly. Hence, the ARI provides a metric for how close a computed clustering is to the true clustering.

4.3 Results of Benchmarking

For each variant tested, the benchmarking process output 8 sets of 50 ARI values based on the number of benchmarks and the number of simulations per benchmark. Figure 4.1 illustrates plotting the 50 ARI values as box plots for each of the variants tested across all eight benchmarks. Generally, the methods with values closer to 1 tended to pick out the true clusters better.

The most striking result visible in in Fig. 4.1 is that the two methods with covariance-based initialization (MCIA-1 and CPCA-1) tended to underperform the two centered Column Profile methods (MCIA-2 and CPCA-2) on every benchmark except Benchmark 6. It is unclear why the latter pre-processing method works better, as both methods ensure all blocks have equal weight when contributing to

the total dataset variance. A more sophisticated analysis is needed to determine the exact cause of this discrepancy.

Another clear result is that there does not seem to be a major difference between the two deflation methods — i.e. deflation by block loadings (MCIA) and deflation by global scores (CPCA) appear to be roughly equivalent. This is broadly in agreement with existing literature (e.g. [6]), although it is possible there are different types of data where the deflation method matters more.

Finally, it is unclear why exactly all methods had trouble with Benchmark 6. It is noteworthy that this is the only benchmark with two large clusters, although there is no mathematical reason to expect the method to fail because of this. It should be noted that this benchmark also posed problems for MCIA in the original benchmarking paper (see Figure 4 in [14]).

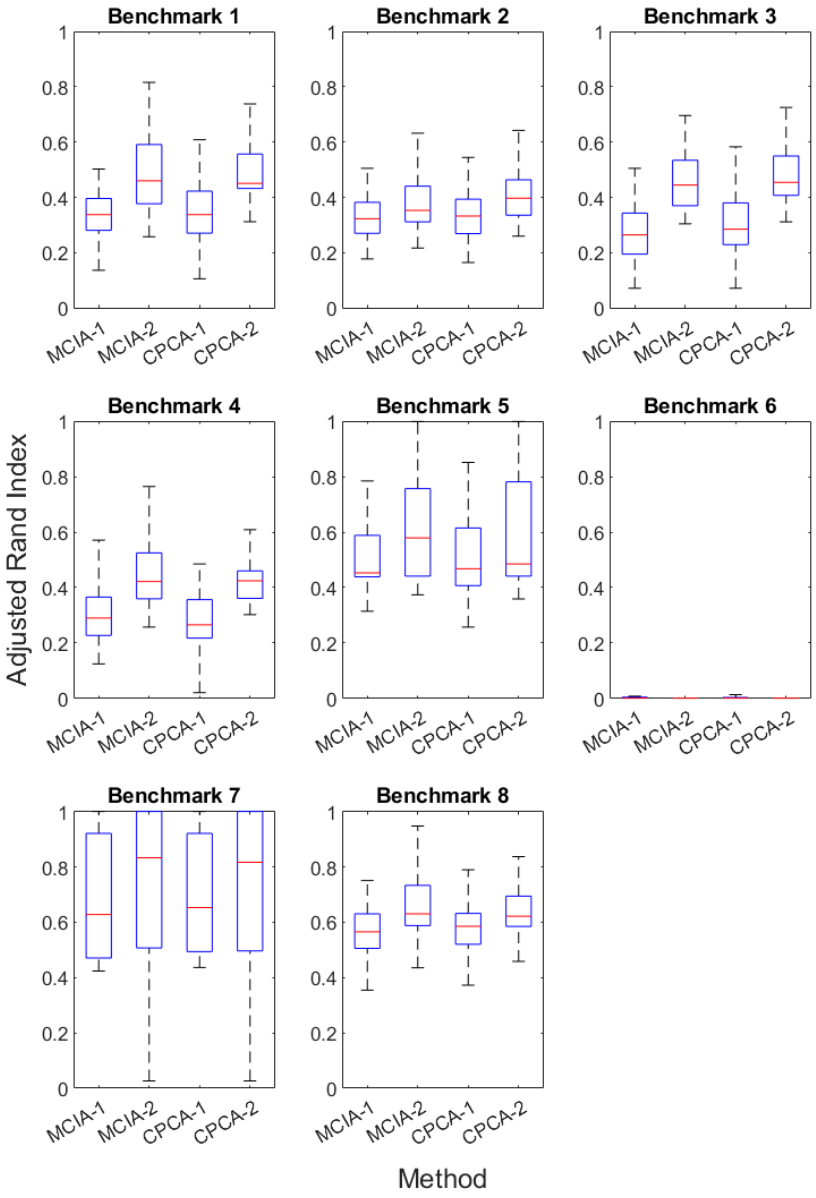


Figure 4.1: Box plots of Adjusted Rand Index values for clustering simulated data for 4 variants of MCIA and CPCA across 8 different benchmarks. Each benchmark is repeated 50 times to generate a range of ARI values. Values closer to 1 indicate a better recovery of the true clusters in the data.

Chapter 5

Conclusions

Starting from the comfortable background of Principal Component Analysis, we have illustrated how to compute MCIA and shown how it is useful as a data analysis method for multi-block datasets. We also adapted a different approach — the modified NIPALS method — to compute MCIA and demonstrated how it satisfied the relevant optimization criteria. Finally, we made an exploratory attempt to benchmark the different initialization and deflation choices for MCIA in clustering on the global scores. We found that the centered row profile initialization seems to outperform the covariance initialization, although could not deduce the reason why this is. There remains a significant amount of further investigation and development that could be done on MCIA.

5.1 Further Evaluation of MCIA Variants

An obvious extension of this approach to benchmarking could be applying the full benchmarking process given by [14]. This would include testing clustering performance on multiple dimensions of low-dimensional embedding, using an automated process to choose the number of clusters, and computing receiver operating characteristic (ROC) curves to evaluate variable importance selection. It would also be useful to derive a metric that takes into account the block scores generated by MCIA, which have been ignored in most benchmarking attempts despite being a key feature of the method.

In addition to simulated data, there is also the opportunity to evaluate the different variations of MCIA on real data. The version of MCIA computed by the `Omicade4` R package has shown good performance on cancer datasets compared to other multi-block methods (e.g. in [1]). Thus, it would be interesting if different variations perform better or worse in these real-world scenarios.

5.2 Adapting MCIA to Tensors

A key limitation of MCIA as it stands is its restriction to two-dimensional datasets — i.e. data without a time dimension. Thus, the method is not easily applicable to the large number of bioinformatics datasets that track variables at different time-points. A closely-related method to MCIA called 'Regularized Generalized Canonical Correlation Analysis' (RGCCA) has recently been adapted to blocks with a tensor structure (in [4]). The resulting method — called Multiway Generalized Canonical Correlation Analysis (MGCCA) — may provide a way to extend MCIA to multi-block tensors.

We can briefly illustrate this potential development path. In the multi-way framework, a multi-block tensor dataset now consists of a collection of N data tensors $\{\underline{\mathbf{X}}_1, ..., \underline{\mathbf{X}}_N\}$, where $\underline{\mathbf{X}}_l$ has dimension $I \times J_l \times K_l$. As with matrix MCIA, note that the number of samples (I) is the same across all blocks while the number of variables (J_l) and time points (K_l) may vary. We further define $\mathbf{X}_l$ to be the first mode matricized version of $\underline{\mathbf{X}}_l$, i.e.

$$\mathbf{X}_l = [\mathbf{X}^l_{:,:,1}|...|\mathbf{X}^l_{:,:,K_l}] \in \mathbb{R}^{I \times J_l K_l} \tag{5.1}$$

where $\mathbf{X}^l_{:,:,k}$ is the k^{th} frontal slice of $\mathbf{X}_l$. MGCCA then solves the the optimization criterion

$$\underset{\mathbf{w}_1,...,\mathbf{w}_N}{\operatorname{argmax}} \sum_{k,l=1}^{N} (c_{kl})\, g(I^{-1}\mathbf{w}_k^T \mathbf{X}_k^T \mathbf{X}_l \mathbf{w}_l) \tag{5.2}$$

$$\text{subject to} \quad \mathbf{w}_l^T \mathbf{M}_l \mathbf{w}_l = 1 \qquad l = 1, ..., N, \tag{5.3}$$

where g is a continuously differentiable function, c_{kl} is a constant picked to describe a 'connection' between data blocks (see [16]), $\mathbf{M}_l$ is a positive definite matrix, and $\mathbf{w}_l \in \mathbb{R}^{K_l \cdot J_l}$ is a weight vector. In the proposed MGCCA framework, the weight vector $\mathbf{w}_l$ is the Kronecker product of weights associated with slices of the tensor

block $\underline{\mathbf{X}}_l$ along the K_l and J_l directions, i.e.

$$\mathbf{w}_l = \mathbf{w}_l^K \otimes \mathbf{w}_l^J \qquad l = 1, ..., L. \tag{5.4}$$

Although (5.2) is not exactly comparable to the MCIA optimization criterion (2.9), choosing $g(x) = x^2$, $c_{kl} = 1$, $\mathbf{M}_l = \mathbf{I}$, and identifying the weight vectors $\mathbf{w}_l$ with the block loadings appears to yield an optimization criterion similar to performing MCIA on the mode-1 unfoldings of each tensor in the multi-block structure. More work would need to be done to finish the adaptation, since (5.2) doesn't differentiate between block and global scores as MCIA does. A benefit of adapting MGCCA to compute 'tensor MCIA' is the fact that MGCCA comes with its own iterative computation method (Algorithm 1 in [4]), meaning a computation method for 'tensor MCIA' might be relatively easy to derive.

Ultimately MCIA has proven to be a highly successful method in multi-block data analysis. There remains much to understand regarding the choice of data normalization and deflation, and we have only completed a first approach at benchmarking a few of these different choices. An adaptation of MCIA to tensor datasets could be particularly helpful for bioinformatics applications, helping to keep MCIA at the forefront of multi-block exploratory data analysis.

Appendix A

Table of Symbols and Notation

$\mathbf{X}$	Dataset matrix or global data matrix (MCIA) typically in $\mathbb{R}^{n \times p}$
p	The number of columns/variables/features of the global data matrix
n	The number of rows/samples/observations of the global and block data matrices
$\mathbf{X}_k$	k^{th} Data block matrix in the multi-block structure (MCIA)
p_k	The number of columns/variables/features of the h^{th} block data matrix
N	The number of blocks in the multi-block structure of $\mathbf{X}$
$\mathbf{f}^{(j)}$	PC score of order j (PCA) or global score of order j (MCIA)
$\mathbf{a}^{(j)}$	PC loading of order j (PCA) or global loading of order j (MCIA)
$\mathbf{f}_k^{(j)}$	Block score for block k of order j
$\mathbf{a}_k^{(j)}$	Block loading for block k of order j
$\mathbf{S}$ or $\mathbf{S_X}$	Covariance matrix of dataset $\mathbf{X}$
$\mathbf{T}^{(j)}$	The $n \times N$ matrix with columns consisting of the order-j block scores

Table A.1: Table of symbols and notation.

Appendix B

MATLAB Code

B.1 Implementation of the NIPALS in MATLAB

Note that the attached code is simply an implementation of the NIPALS iteration found in section 3, and only represents part of the code necessary to replicate the figures in this paper. The remaining code, including sample data, plotting, and initialization code is available at `https://github.com/Muunraker/MATLAB_MCIA`

```
function [F, Q, F_block, Q_block, EigVals, B_weights] = nipals_multiBlock(X_normalized,..
num_PCs, tol, max_iter, deflationMethod)
% Function to implement the NIPALS method from Hanafi 2010
% Options to perform either Multiple Co-Inertia Analysis or
% Consensus Principle Component Analysis
% * Does NOT include any preprocessing steps on input data.
% * Does NOT normalize output vectors.
% Inputs:
%         * X_normalized    Vertical cell array of NORMALIZED data matrices
%         (each has n rows)
%         * num_PCs         Number of PCs desired in output
%         * tol             Scalar tolerance in stopping criterion for NIPALS
%         * max_iter        Maximum number of iterations allowed in NIPALS algorithm
%         * deflationMethod Choice of block deflation method:
%                               'block' to use block loadings for deflation
%                               (MCIA - default)
%                               'global' to use global scores for deflation (CPCA)
% Outputs:
%         * F               Matrix of global scores
```

```
%         * Q               Matrix of global loadings
%         * F_block         Cell array of local score matrices
%         * Q_block         Cell array of local loading matrices
%         * EigVal          Array of eigenvalues corresponding to first
%                           'num_PCs' global scores
%         * B_weights       Matrix where i^th column has the block
%                           contributions to i^th global score
if nargin < 5
deflationMethod = 'block';
end
% Getting variable and sample numbers for each dataset
num_datasets = length(X_normalized);
% Extract number of samples in datasets (i.e. # rows)
num_samples = size(X_normalized{1},1);
% Array with number of variables in each block
num_variables = zeros(1,num_datasets);
for i = 1:num_datasets
num_variables(1,i) = size(X_normalized{i},2);
end
% Total number of variables across whole dataset:
total_vars = sum(num_variables,'all');
% Global score/loadings storage matrices
F = zeros(num_samples,num_PCs);% Matrix of global scores
% - each column is one global score
Q = zeros(total_vars,num_PCs); % Matrix of global loadings
% - each column is one global loading (F = X Q)

% Block score storage cell array
F_block = cell(1,num_datasets);
% CELL index specifies the dataset
```

```
% For cell index i: F_block{i} = [f_i^(1) | f_i^(2) | ... | f_i^(num_PCs)]
for i = 1:num_datasets
F_block{i} = zeros(num_samples, num_PCs);
end

% Block loadings storage cell array
Q_block = cell(1,num_datasets);
% CELL index specifies the dataset
% For cell index i: Q_block{i} = [q_i^(1) | q_i^(2) | ... | q_i^(num_PCs)]
for i = 1:num_datasets
X_i = X_normalized{i};
num_vars = size(X_i); num_vars = num_vars(2);
Q_block{i} = zeros(num_vars, num_PCs);
end

%% NIPALS implementation (see Hanafi et. al. 2010)
X_deflated = X_normalized; % Creating copy of data to run deflation process
% Creating data super-matrix
X_super = X_normalized{1};
if num_datasets > 1
for i = 2:num_datasets
X_super = [X_super X_normalized{i}];
end
end
% Creating eigenvalue list
EigVals = zeros(1,num_PCs);
% Creating block weight array
B_weights = zeros(num_datasets,num_PCs);

% Main NIPALS iteration
```

```
for j = 1:num_PCs % repeat deflation/iteration process for desired number of PCs
%%% Initialization
% matrix to store block scores
T = zeros(num_samples,num_datasets);
% cell array to store block loading vectors
block_loadings = cell(1,num_datasets);
f = ones(num_samples,1); % arbitrary starting vector
iter = 0; % iteration count
as_old = 0; % stopping criterion variable
err = 10; % tolerance metric
if ~exist('max_iter')
max_iter = 10000; % iteration limiter if not specified
end

%%% NIPALS iteration loop
while err > tol
for i = 1:num_datasets
X_i = X_deflated{i}; % extract ith dataset
q_i = X_i'*f/(f'*f); % compute block loadings
q_i = q_i/norm(q_i); % normalize loading vector
f_i = X_i*q_i; % compute block scores
T(:,i) = f_i; % add block score to matrix;
block_loadings{i} = q_i; % store block loadings
end
w = T'*f/(f'*f); % compute global weights (relative to matrix of
w = w/norm(w); % normalize global weights
f = T*w; % update global score

% calculating stopping criterion:
fnorm = f/sqrt(cov(f)); % unit variance global scores
```

```
% iterating stopping metric
as = 0;
for i = 1:num_datasets
C = cov(T(:,i),fnorm);
C = C(2)^2; % taking the covariance between qnorm and block scores.
as = as + C;
end
% checking stopping metric vs. stopping criterion
err = abs(as - as_old); % error metric
as_old = as;

iter = iter+1;

if iter > max_iter
fprintf('WARNING: exceeded maximum iteration threshold \n')
break
end
end
%%% Updating outputs not used in iteration
for i =1:num_datasets
% record jth order block score in matrix relating to block i:
Fi = F_block{i};
Fi(:,j) = T(:,i);
F_block{i} = Fi;

% record jth order block loading in matrix relating to block i:
Qi = Q_block{i};
Qi(:,j) = block_loadings{i};
Q_block{i} = Qi;
```

```
end

% jth global score -->  jth column of matrix F
F(:,j) = f;
% jth set of block weights --> jth column of B_weights
B_weights(:,j) = w;
% Computing global loadings
Qi = Q_block{1};
q_global = B_weights(1,j)*Qi(:,j);
q_global= q_global';
% compute global loadings by weighted concatenation of block loadings:
if num_datasets > 1
for i = 2:num_datasets
Qi = Q_block{i};
q_global = [q_global (B_weights(i,j)*Qi(:,j))';];
end
end
% jth global loading --> jth column of matrix Q
Q(:,j) = q_global;
% Creating deflated data super-matrix
X_super_deflated = X_deflated{1};
if num_datasets > 1
for i = 2:num_datasets
X_super_deflated = [X_super_deflated X_deflated{i}];
end
end

% Recording first eigenvalue of deflated super-matrix (corresponding to
% covariance matrix)
svs = svd(X_super_deflated);
```

```
EigVals(j) = svs(1);
fprintf(['Completed after ', num2str(iter), ' iterations \n']);

%%% Deflation step
% Deflate using either global scores (for CPCA) or block loadings (for MCIA)
if strcmp('global',deflationMethod)
% Deflation via global scores
for i = 1:num_datasets
X_i = X_deflated{i}; % extract ith dataset
Xnew_i = X_i - (f*f'/(f'*f))*X_i;
X_deflated{i} = Xnew_i;
end
else
% Deflation via block loadings
for i = 1:num_datasets
X_i = X_deflated{i}; % extract ith dataset
% extract block loading matrix related to ith dataset:
q_i = Q_block{i};
Xnew_i = X_i - X_i*q_i(:,j)*(q_i(:,j)');
X_deflated{i} = Xnew_i;
end
end
end
end
```

Bibliography

[1] Laura Cantini, Pooya Zakeri, Celine Hernandez, Aurelien Naldi, Denis Thieffry, Elisabeth Remy, and Anaïs Baudot, *Benchmarking joint multi-omics dimensionality reduction approaches for the study of cancer*, Nature Communications, 12 (2021), p. 124.

[2] D Chessel and M Hanafi, *Analyses de la co-inertie de K nuages de points*, Revue de statistique appliquée, 44 (1996), pp. 35–60. Publisher: Société de statistique de France.

[3] Stéphane Dray and Anne-Béatrice Dufour, *The **ade4** Package: Implementing the Duality Diagram for Ecologists*, Journal of Statistical Software, 22 (2007).

[4] Arnaud Gloaguen, Cathy Philippe, Vincent Frouin, Giulia Gennari, Ghislaine Dehaene-Lambertz, Laurent Le Brusquet, and Arthur Tenenhaus, *Multiway generalized canonical correlation analysis*, Biostatistics, 23 (2022), pp. 240–256.

[5] Mohamed Hanafi, Achim Kohler, and El-Mostafa Qannari, *Connections between multiple co-inertia analysis and consensus principal component analysis*, Chemometrics and Intelligent Laboratory Systems, 106 (2011), pp. 37–40.

[6] Sahar Hassani, Mohamed Hanafi, El Mostafa Qannari, and Achim Kohler, *Deflation strategies for multi-block principal component analysis revisited*, Chemometrics and Intelligent Laboratory Systems, 120 (2013), pp. 154–168.

[7] Lawrence Hubert and Phipps Arabie, *Comparing partitions*, Journal of Classification, 2 (1985), pp. 193–218.

[8] Ian T Jolliffe and Jorge Cadima, *Principal component analysis: a review and recent developments*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374 (2016), p. 20150202. Publisher: The Royal Society Publishing.

[9] H. Kriegel, P. Kroger, M. Renz, and S. Wurst, *A Generic Framework for Efficient Subspace Clustering of High-Dimensional Data*, in Fifth IEEE International Conference on Data Mining (ICDM'05), Houston, TX, USA, 2005, IEEE, pp. 250–257.

[10] Hongfang Liu, Petula D'Andrade, Stephanie Fulmer-Smentek, Philip Lorenzi, Kurt W. Kohn, John N. Weinstein, Yves Pommier, and William C. Reinhold, *mRNA and microRNA Expression Profiles of the NCI-60 Integrated with Drug Activities*, Molecular Cancer Therapeutics, 9 (2010), pp. 1080–1091.

[11] Chen Meng, Bernhard Kuster, Aedín C Culhane, and Amin Moghaddas Gholami, *A multivariate approach to the integration of multi-omics datasets*, BMC Bioinformatics, 15 (2014), p. 162.

[12] Chen Meng, Oana A. Zeleznik, Gerhard G. Thallinger, Bernhard Kuster, Amin M. Gholami, and Aedín C. Culhane, *Dimension reduction techniques for the integrative analysis of multi-omics data*, Briefings in Bioinformatics, 17 (2016), pp. 628–641.

[13] Jacques Pagès, Christian Asselin, René Morlat, and J Robichet, *L'analyse factorielle multiple dans le traitement des donnees sensorielles. Application a des vins rouges de la vallee de la Loire.*, Sciences des aliments, 7 (1987), pp. 549–571.

[14] Morgane Pierre-Jean, Jean-François Deleuze, Edith Le Floch, and Florence Mauger, *Clustering and variable selection evaluation of 13 unsupervised methods for multi-omics data integration*, Briefings in Bioinformatics, 21 (2020), pp. 2011–2030.

[15] Roman Rosipal, in Chemoinformatics and Advanced Machine Learning Perspectives: Complex Computational Methods and Collaborative Techniques, Huma Lodhi and Yoshihiro Yamanishi, eds., IGI Global, 2011, pp. 169–189.

[16] Arthur Tenenhaus and Michel Tenenhaus, *Regularized generalized canonical correlation analysis for multiblock or multigroup data analysis*, European Journal of Operational Research, 238 (2014), pp. 391–403.