



Matrix Integrative Analysis (MIA) of Multiple Genomic Data for Modular Patterns

Jinyu Chen^{1,2} and Shihua Zhang^{1,2,3*}

¹ NCMIS, CEMS, RCSDS, Academy of Mathematics and Systems Science, CAS, Beijing, China, ² School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, China, ³ Center for Excellence in Animal Evolution and Genetics, Chinese Academy of Sciences, Kunming, China

The increasing availability of high-throughput biological data, especially multi-dimensional genomic data across the same samples, has created an urgent need for modular and integrative analysis tools that can reveal the relationships among different layers of cellular activities. To this end, we present a MATLAB package, Matrix Integration Analysis (MIA), implementing and extending four published methods, designed based on two classical techniques, non-negative matrix factorization (NMF), and partial least squares (PLS). This package can integrate diverse types of genomic data (e.g., copy number variation, DNA methylation, gene expression, microRNA expression profiles, and/or gene network data) to identify the underlying modular patterns by each method. Particularly, we demonstrate the differences between these two classes of methods, which give users some suggestions about how to select a suitable method in the MIA package. MIA is a flexible tool which could handle a wide range of biological problems and data types. Besides, we also provide an executable version for users without a MATLAB license.

Keywords: bioinformatics, multi-dimensional genomics, matrix integrative analysis, data integration, module discovery, non-negative matrix factorization (NMF), partial least squares (PLS)

OPEN ACCESS

Edited by:

Fengfeng Zhou,
Jilin University, China

Reviewed by:

Xiaowei Wu,
Virginia Tech, United States
Zhaohui Steve Qin,
Emory University, United States
Yaping Yang,
University of Georgia, United States

*Correspondence:

Shihua Zhang
zsh@amss.ac.cn

Specialty section:

This article was submitted to
Bioinformatics and Computational
Biology,
a section of the journal
Frontiers in Genetics

Received: 12 February 2018

Accepted: 11 May 2018

Published: 29 May 2018

Citation:

Chen J and Zhang S (2018) Matrix Integrative Analysis (MIA) of Multiple Genomic Data for Modular Patterns. *Front. Genet.* 9:194. doi: 10.3389/fgene.2018.00194

INTRODUCTION

Biological systems are very complex, consisting of diverse components interacting with each other cooperatively. A key problem of biology is to investigate the relationships among different layers of cellular activity and thereby gain deep understanding of the underlying regulatory mechanisms. Meanwhile, the rapid development of high-throughput genomics technologies has accelerated the generation of large-scale genomic data at multiple levels simultaneously on the same samples. For example, The Cancer Genome Atlas (TCGA) project (The Cancer Genome Atlas Research Network, 2008) provides copy number variation, DNA methylation, microRNA and gene expression profiles for the same set of tumor samples. Thus, it is an essential and valuable task to develop tools that can explore combinatorial relationships among multiple layers of cellular activities. To this end, we present a MATLAB package, Matrix Integration Analysis (MIA), which implements four methods for modular and integrative analysis. The four methods are extensions of two classical techniques—non-negative matrix factorization (NMF) and partial least squares (PLS).

In this paper, we firstly provide a brief review about these two classes of methods and their applications in bioinformatics as well as four methods in the MIA package. We then focus on the details about MIA including its implementation, input data structures, and outputs. We also give a very detailed user manual about this package in our online guide file. Moreover, we summarize

the characteristics of each method in the MIA package when applying to real data matrices, which provides users some practical suggestions about method selection. Particularly, in order to demonstrate the differences between these two classes of methods clearly, we compare NMF-based methods with PLS-based methods by applying them to some sets of simulated data matrices.

MATERIALS AND EQUIPMENT

Brief Review of NMF

The NMF technique, as an unsupervised and part-based learning method, has been increasingly applied to diverse fields including bioinformatics [e.g., high-dimensional genomic data analysis (Brunet et al., 2004; Kim and Park, 2007; Devarajan, 2008; Zhang et al., 2011, 2012; Li et al., 2012; Chen and Zhang, 2016)]. It decomposes a non-negative matrix X of size $m \times n$ into two non-negative matrices—a basis matrix $W \in \mathbb{R}^{m \times k}$ and a coefficient matrix $H \in \mathbb{R}^{k \times n}$, such that $X \approx WH$, where $k < \min\{m, n\}$ (Paatero and Tapper, 1994; Lee and Seung, 1999). That is, data X is explained as a positive linear combination of basis vectors (the columns of W). To find an appropriate decomposition for matrix X , a number of cost functions are proposed (Paatero and Tapper, 1994; Lee and Seung, 1999, 2000; Cichocki et al., 2006, 2008; Dhillon and Sra, 2006; Fevotte et al., 2009). Two types of cost functions are often used to measure the distance between the original data matrix X and the reconstructed matrix WH . One is based on the Euclidean distance, that is, $\min_{W,H} \|X - WH\|_F^2$ (Paatero and Tapper, 1994; Lee and Seung, 2000); the other is based on Kullback-Leibler (KL) divergence, that is, $\min_{W,H} \sum_{i,j} \left(X_{ij} \log \frac{X_{ij}}{(WH)_{ij}} - X_{ij} + (WH)_{ij} \right)$, where X_{ij} represents the element of matrix X (Lee and Seung, 1999, 2000). Factored matrices W and H could be used to identify the underlying substructures in data matrix X .

The simplicity and efficiency of NMF make it have wide applications. For example, Brunet et al. (2004) and Moffitt et al. (2015) applied NMF to gene expression data to discover cancer subtypes. Compared to other traditional clustering methods, such as hierarchical clustering and k -means, NMF is able to capture local structures and provides a robust clustering of samples. Besides, NMF is able to identify genomic modules involved in similar biological functions using cancer genomic data, such as gene modules (Kim and Tidor, 2003) and mutation signature modules (Nik-Zainal et al., 2012; Alexandrov et al., 2013a,b; Kasar et al., 2015), which help us understand the underlying pathogenic mechanism of different types of cancers.

Several variants of NMF have been proposed by adding distinct constraints. For example, Feng et al. (2002) imposed sparseness constraints to matrix H and locality constraints to matrix W on the basis of KL divergence-type objective function, that is, $\min_{W,H} \sum_{i,j} \left(X_{ij} \log \frac{X_{ij}}{(WH)_{ij}} - X_{ij} + (WH)_{ij} \right) + \alpha \sum_{i,j} (W^T W)_{ij} - \beta \sum_i (HH^T)_{ii}$ to learn spatially localized parts-based representation of visual patterns. In order to achieve feature selection, it is useful to explicitly control the sparse degree of W and H by imposing sparsity constraints (e.g., L_0 -, L_1 -, or

L_2 -norm penalty) to W and H (Kim and Park, 2007; Peharz and Pernkopf, 2012). These sparse NMF methods not only perform the reconstruction of original data X , but also extract highly localized patterns. Orthogonal constraints are also commonly used in NMF (Ding et al., 2006; Wang et al., 2008; Strazar et al., 2016), that is, $W^T W = I$ and (or) $H^T H = I$. Orthogonal NMF leads to rigorous clustering interpretation. Recently, Strazar et al. (2016) applied an orthogonal joint NMF framework to integrate multiple data for deciphering RNA binding patterns and predictions.

Network-regularized NMF (Cai et al., 2008; Wang et al., 2008; Zhang et al., 2011) incorporates prior knowledge such as interaction networks between features in data X into NMF framework such that linked features in the networks are sufficiently close to each other in the new representation space. For example, Zhang et al. (2011) utilized the same skill to add graph constraints about gene-gene and gene-microRNA interaction networks to NMF framework, and identified gene-microRNA co-modules which function cooperatively in the biological system.

Moreover, in order to simultaneously integrate more than one type of genomic data across the same set of samples, Zhang et al. (2012) proposed a joint NMF (jNMF) technique, which simultaneously decomposes multiple genomic data matrices X_i into a common basis matrix W and individual metagene matrix H_i . Based on H_i , they identified a number of multi-dimensional modules, each of which comprises a set of genes, microRNAs and methylation markers. Ray and Bandyopadhyay (2016) proposed a NMF based approach to integrate multiple biological networks, including gene co-expression network, PPI network and GO semantic similarity network, to predict the interactions between HIV-1 and human proteins.

Besides two-factor NMF, three-factor NMF is also an important class of matrix factorization techniques (Ding et al., 2006; Wang et al., 2008; Pei et al., 2015; Zitnik and Zupan, 2015; Zitnik et al., 2015), that is, $X \approx FSG$. Such format provides a framework to perform biclustering of the rows and columns of matrix X by matrices F and G , respectively. Factored matrix S not only provides an additional degree of freedom making the approximation tightly, but also could indicate the relations between identified clusters. Recently, Zitnik and Zupan (2015) proposed a data fusion model by matrix factorization (DFMF), which can integrate multiple relationships between multiple object types as well as constraints for these object types. They applied it to fuse 11 data matrices including gene expression profiles, gene annotations with GO terms, and KEGG pathways and so on, to predict gene functions.

For the convenience of users, NMF including its variants has been implemented in several standard programming languages. For example, Gaujoux and Seoighe (2010) provided an R package implementing several NMF optimization algorithms and Qi et al. (2009) developed a NMF analysis plug-in in BRB-ArrayTools implemented in R for microarray gene expression analysis. There also exist a number of MATLAB packages for NMF, such as the package developed by Brunet et al. (2004) for metagenes and molecular pattern discovery; the NMF Toolbox developed by Li and Ngom (2013), which contains a variety of techniques for

biological data mining; bioNMF proposed by Pascual-Montano et al. (2006), which is a standalone tool for classical NMF and contains a user-friendly graphical interface to demonstrate data analysis results.

For NMF, the number of latent factors k in the decomposition is a key parameter which needs to be pre-determined. For the application that using NMF to perform clustering, people often determine k based on a consensus clustering matrix by means of cophenetic correlation coefficient (Brunet et al., 2004; Zitnik and Zupan, 2015) and the stability of clustering. In addition, the distance between the original matrix X and the reconstructed matrix WH can also be used as a measure to select k (Kim and Tidor, 2003; Zitnik and Zupan, 2015; Zitnik et al., 2015). Recently, Wu et al. (2016) selected k when the learned basis matrix W achieves the lowest instability under different initial starting points in NMF algorithm. Particularly, in biological analysis, we could select k based on prior knowledge about the input data matrix X or functional enrichment analysis.

Brief Review of PLS

PLS is a multivariate regression method used to find the relations between an input matrix $X \in \mathbb{R}^{m \times n_1}$ and a response matrix $Y \in \mathbb{R}^{m \times n_2}$, both of which have the same rows (samples) (Wold et al., 2001; Rosipal and Kramer, 2006; Lê Cao et al., 2008; Chun and Keles, 2010; Li et al., 2012; Chen and Zhang, 2016). Compared to the classical linear regression, it works well for the data with small sample size and a large number of variables ($m < n_1$). PLS decomposes matrices X and Y both with zero-mean variables into the form: $X = TP^T + E$, $Y = UQ^T + F$, where T , U are $m \times k$ matrices of k latent components describing the original data matrices in a low-dimensional space, and T , U can be constructed as a linear transformation of X , Y , respectively (that is, $T = XW$; $U = YC$); P , Q represent loading matrices to measure the relationships between the original variables and latent ones; E , F are residual matrices. PLS aims to maximize the covariance between X and Y by means of latent components T and U . One common type objective function of PLS is $\max_{w,c} [\text{cov}(t, u)]^2 = [\text{cov}(Xw, Yc)]^2$, subject to $w^T w = c^T c = 1$. Here, w and c are respectively one column of weight matrices W and C ; similarly, t and u are respectively one column of latent component matrices T and U . If the columns of X and Y are zero-mean, $\text{cov}(Xw, Yc)$ could be calculated by $w^T X^T Y c / m$. Since $[\text{cov}(Xw, Yc)]^2 = \text{var}(Xw) [\text{corr}(Xw, Yc)]^2 \text{var}(Yc)$, thus latent variables t and u identified by PLS simultaneously take into account the requirements of maximal correlation between X and Y like Canonical Correlation Analysis (CCA), explaining as much variance as possible in both X - and Y -space like Principal Components Analysis (PCA). This model can be solved by the non-linear iterative partial least squares (NIPALS) algorithm (Wold, 1975; Wold et al., 2001; Rosipal and Kramer, 2006). Note that it only extracts one latent component t and u in one round of calculation. Loading vectors p and q are computed as the regression coefficients of X on t and Y on u , respectively: $p = X^T t / (t^T t)$ and $q = Y^T u / (u^T u)$.

Before calculating the next pair of latent variables t and u , matrices X and Y need to be deflated by subtracting their

rank-one approximations. Different models have used different deflation forms (Rosipal and Kramer, 2006). If people want to model the symmetric relation between the two blocks X and Y , X and Y are deflated as $X = X - tp^T$, $Y = Y - uq^T$; If a regression model is needed, the computed latent components (column vectors in matrix T) are treated as good predictors of Y . Assume that a linear relation between t and u exists, $U = TD + H$, where D is a diagonal matrix and H is a residual matrix, thus X and Y are updated as $X = X - tp^T$, $Y = Y - tt^T Y / (t^T t)$.

PLS has been widely applied to various problems in bioinformatics (Boulesteix and Strimmer, 2005; Singh et al., 2016). For example, Boulesteix and Strimmer (2005) utilized PLS regression method to predict transcription factor activities by combining mRNA expression (Y) and DNA-protein binding data (X). Besides, in order to perform feature selection for high dimensional data, sparse PLS (SPLS) technique has been developed (Lê Cao et al., 2008; Chun and Keles, 2010), which imposes sparsity penalties to weight vectors w and c . For example, Lê Cao et al. (2008) used Lasso penalty to select biologically interpretable genes. Morine et al. (2011) used SPLS regression to assess relationships between dietary components and gene expression levels. Huang et al. (2004) also proposed a sparse PLS named Penalized PLS (PPLS) method, in which they selected key features by means of soft thresholding process to shrink the coefficients of some features to zero. Their results indicate the selected features by PPLS could provide more accurate prediction than traditional PLS and a random forest method. Furthermore, Lique et al. (2016) proposed group PLS and sparse group PLS models, which take into account the group effects due to relationships among predictors (e.g., genes in the same pathway). However, they only consider non-overlapping groups, which may need to improve due to the popularity of overlapping groups in practice.

In addition, PLS can be combined with classification methods to solve classification tasks. For examples, Nguyen and Rocke performed a two-group tumor classification task on the microarray gene expressions by combining PLS and discrimination analysis (Nguyen and Rocke, 2002b). Furthermore, they extended this method to solve the multi-class cancer classification problem (Nguyen and Rocke, 2002a). Moreover, a number of methods combining SPLS with different classification models such as linear discriminant analysis, support vector machine and random forest are developed (Gutkin et al., 2009; Chung and Keles, 2010; Lê Cao et al., 2011). They could construct more accurate and efficient classifiers with selected features.

PLS has been widely implemented. For example, the Statistics and Machine Learning Toolbox module `plsregress` in MATLAB carries out PLS regression. The `pls` package provided by Mevik and Wehrens implements several PLS algorithms in R (Mevik and Wehrens, 2007). The `gpls` package (Ding, 2016) is able to accomplish classification using generalized partial least squares for two-group and multi-group classification. Particularly, `plsgenomics` developed by Boulesteix et al. is a PLS-based genomic analysis R package (Boulesteix et al., 2015), mainly implementing PLS methods for classification with microarray data and prediction of transcription factor activities

from combined ChIP-chip analysis. Lê Cao et al. developed an R package, *integrOmics*, implementing SPLS, which could simultaneously integrate two types of “omics” variables that are measured on the same samples and realize variables selection using lasso penalization (Lê Cao et al., 2009). Chung et al. developed the *spls* package to perform sparse PLS regression and classification in R (Chung et al., 2013).

PLS also needs to determine the number of latent components. Cross-validation (CV) strategy is usually used to solve it (Wold et al., 2001; Boulesteix and Strimmer, 2005). Besides, the robustness of a PLS algorithm is also an important issue. It is well-known that the two popular algorithms NIPALS and SIMPLS for solving PLS are very sensitive to outliers in the data (Cummins and Andrews, 1995; Gil and Romera, 1998; Hubert and Vanden Branden, 2003; Serneels et al., 2005). Several robust PLS algorithms have been proposed including using iteratively reweighting technique, that is, assigning outliers detected by some way with low weights (Cummins and Andrews, 1995; Serneels et al., 2005); and estimating a robust covariance matrix instead of the previous one used in PLS (Gil and Romera, 1998; Hubert and Vanden Branden, 2003).

As for NMF and PLS, they are both powerful methods to capture the inherent structures of data. The optimization problems corresponding to them are not convex, thus it is very hard to find a global optimal solution. Besides, the number of latent components k needs to be pre-determined for both NMF and PLS. However, there are several differences between them. Firstly, NMF aims at exploring local patterns in one data matrix X . The identified local patterns are some features sharing high signals across a same subset of samples. PLS emphasizes regression analysis exploring the relationships between two types of features in matrices X and Y . The highly correlated features in X and Y are selected. Secondly, the constraints for them are distinct. NMF requires input data are non-negative whereas PLS needs centered data across samples as input just like simple linear regression.

MIA: A Matlab Package for Matrix Integrative Analysis

The wide applications of NMF and PLS methods prove that NMF and PLS are two types of powerful tools for analyzing genomic data. Recently, we have developed four promising methods, by extending classical NMF and PLS, for detecting multi-dimensional modules (md-modules) in diverse genomic data (as well as prior network data) (Table 1): jNMF (joint NMF) (Zhang et al., 2012), SNMNMf (Sparse Network-regularized Multiple NMF) (Zhang et al., 2011), sMBPLS (sparse Multi-Block PLS) (Li et al., 2012), and SNPLS (Sparse Network-regularized PLS) (Chen and Zhang, 2016). Currently, practical tools to integrate multi-dimensional genomic data are still lacking. To this end, we develop a unified Matrix Integration Analysis (MIA) package, implementing these four methods as a set of MATLAB functions, to facilitate their adoption, promotion and evaluation (Figure 1).

Brief Review of the Four Methods in MIA

The four methods are developed for integrative and modular analysis for multiple genomic data as well as prior network data. We briefly review these methods and clarify some differences among them in terms of input data types, data formats, and module characteristics.

jNMF

jNMF enables users to simultaneously factor two- or multi-dimensional genomic data of the same set of samples. It discovers local patterns across the same subset of samples in multiple data matrices simultaneously. Note that matrices with negative elements must be transformed into non-negative matrices before analysis to satisfy the non-negative constraint. In this package, such matrices are transformed by $X^* = [\max(X, 0), \max(-X, 0)]$, where each row represents one sample (users may adopt other ways to transform a matrix). More importantly, we extend the implementations in Zhang et al. (2012) to be suitable for any number of matrices, that is,

$$\min_{W, H_i} \sum_{i=1}^n \|X_i - WH_i\|_F^2, \text{ s.t. } W \geq 0, H_i \geq 0, i = 1, \dots, n.$$

SNMNMf

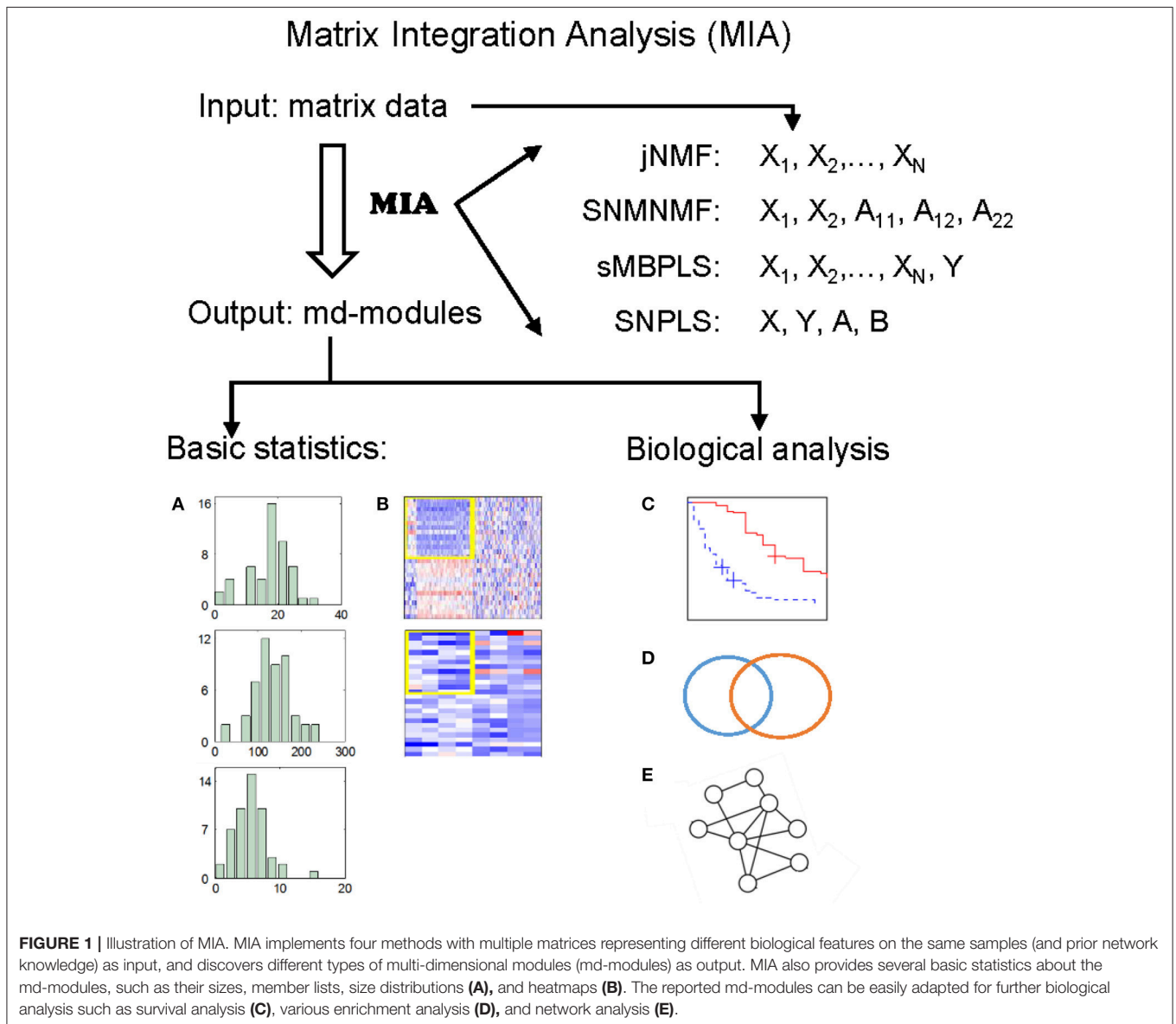
SNMNMf integrates prior networks relating to input variables into jNMF for pairwise case. For example, a gene-gene interaction network and a microRNA-gene interaction network have been adopted into this framework for discovering microRNA-gene co-modules (Zhang et al., 2011). By expressing networks as network-regularized penalties, SNMNMf makes the variables linked in these networks more likely to be placed into the same module, which will be biologically interpretable. Compared with the model in Zhang et al. (2011), SNMNMf implemented in this package can incorporate the links within one type of variables ($A_{11} \in \mathbb{R}^{n_1 \times n_1}$, $A_{22} \in \mathbb{R}^{n_2 \times n_2}$) and between the two types of variables ($A_{12} \in \mathbb{R}^{n_1 \times n_2}$) in a more flexible manner, where A_{11} , A_{22} , A_{12} are adjacency matrices for the three networks, that is,

$$\begin{aligned} \min_{W, H_i} \sum_{i=1}^2 \|X_i - WH_i\|_F^2 - \sum_{1 \leq i < j \leq 2} \lambda_{ij} \text{Tr}(H_i A_{ij} H_j^T) \\ + \gamma_1 \|W\|_F^2 + \gamma_2 \left(\sum_i \|h_i^{(1)}\|_1^2 + \sum_j \|h_j^{(2)}\|_1^2 \right), \\ \text{ s.t. } W \geq 0, H_i \geq 0, i = 1, 2. \end{aligned}$$

where $h_i^{(1)}$ and $h_j^{(2)}$ denote the i -th and j -th columns of H_1 and H_2 respectively; λ_{ij} , γ_1 , and γ_2 are parameters respectively controlling the degree of network constraints, the growth of W and the sparsity of H_i , which all need to be pre-defined by users according to their input data. The term in this objective function, $\text{Tr}(H_i A_{ij} H_j^T) = \sum_{p,q} a_{pq} h_p^{(i)T} h_q^{(j)}$, enforces “must-link” constraint such that features with known interactions in A_{ij} have similar coefficient profiles in matrix H_i and H_j . We note that users can choose the networks they prefer to use in this framework by setting the corresponding parameters non-zero.

TABLE 1 | Brief summary of the four methods in MIA.

Method	Description	Non-negative constraint	Network penalty	No. of matrices	References
jNMF	Factorize multi-dimensional genomic data simultaneously to reveal multi-dimensional modules in an unsupervised manner.	Yes	No	Multiple	Zhang et al., 2012
SNMNMf	Incorporate prior networks into jNMF for two types of data to enhance co-module discovery.	Yes	Yes	Pairwise	Zhang et al., 2011
sMBPLS	Extend sparse PLS regression model for simultaneously analyzing multi-dimensional genomic data to reveal multi-dimensional regulatory modules.	No	No	Multiple	Li et al., 2012
SNPLS	Incorporate prior networks into sMBPLS for pairwise data matrices to reveal co-module patterns.	No	Yes	Pairwise	Chen and Zhang, 2016



sMBPLS

sMBPLS extends standard PLS to discover associations between multiple input matrices and a response matrix in a sparse manner (Li et al., 2012). This method in the MIA package is implemented to explore the regression relationships between any number of input matrices and one response matrix, that is,

$$\begin{aligned} \max_{w_i, q, b_i} \quad & \text{cov}(t, u) - \sum_{i=1}^n \lambda_i \|w_i\|_1 - \mu \|q\|_1 \\ \text{with } t_i = & X_i w_i, \quad u = Yq, \quad \text{and } t = \sum_{i=1}^n b_i t_i \\ \text{s.t. } \|w_i\|_2 = & 1, \quad \|q\|_2 = 1, \quad \|b\|_2 = 1, \quad i = 1, \dots, n. \end{aligned}$$

where λ_i , μ control the degrees of sparsity of weight vectors w_i, q . In MIA, users need to provide several candidate values for each parameter and the program is able to select a group of proper parameters for input data by means of a cross-validation procedure.

SNPLS

SNPLS is designed for one input matrix ($X \in \mathbb{R}^{m \times n_1}$) and one response matrix ($Y \in \mathbb{R}^{m \times n_2}$). It adopts network-regularized constraints via the adjacency matrices $A_{11} \in \mathbb{R}^{n_1 \times n_1}$ of a given network G_1 for X and/or $A_{22} \in \mathbb{R}^{n_2 \times n_2}$ of another network G_2 for Y . This method in MIA enables users to make use of the network structures about input data X and response data Y if available flexibly, that is,

$$\begin{aligned} \max_{w, q} \quad & \text{cov}(t, u) - \gamma_1 w^T L_1 w - \gamma_2 q^T L_2 q - \lambda \|w\|_1 - \mu \|q\|_1 \\ \text{with } t = & Xw, \quad u = Yq. \\ \text{s.t. } \|w\|_2 = & 1, \quad \|q\|_2 = 1. \end{aligned}$$

where $L_i = D_i^{-\frac{1}{2}} (D_i - A_{ii}) D_i^{-\frac{1}{2}}$, $i = 1, 2$. D_i is the degree matrix of graph G_i , that is, the elements of D_i : $d_{kk} = \sum_j a_{jk}$ and $d_{jk} = 0$, for $j \neq k$. Similar with sMBPLS, the candidates of parameters γ_1 , γ_2 , λ , μ also need to be given, and the program will choose a proper group automatically.

STEPWISE PROCEDURES

Implementation

To run MIA, all the input genomic data need to be combined into a single data matrix in which the rows correspond to samples and the columns correspond to genomic features. Different types of genomic data correspond to different sets of columns in the input matrix. Prior biological networks can also be given as inputs of SNMNMF and SNPLS. They are represented as network-regularized constraints for enhancing module discovery. One can run the main function of MIA with a pre-selected method to automatically perform computations on the input data matrix. The outputs include a set of text files and figures to describe the discovered md-modules (Supplementary Materials and Guide). The installation of the MIA package is as follows:

- 1) Download the MIA package in the website: <http://page.amss.ac.cn/shihua.zhang/software.html>.

- 2) Unzip the package into a specific directory (e.g., 'D:/'), and set the work path of MATLAB (e.g., 'D:/MIA/').
- 3) Load the data. In the MIA package, there is a folder named 'InputData' storing the demo input data for the four methods. For example, we load the data named 'InputDataForSNMNMF.mat' as input of SNMNMF.


```
>> load('InputData/InputDataForSNMNMF.mat','Input');
```
- 4) Run the main function *MIA.m* with a desired method. For example, we select SNMNMF for analyzing the loaded data.


```
>> MIA(Input, 'SNMNMF');
```

Then, MIA automatically performs all computations and saves all the results into the directory 'MIA/SNMNMF/SNMNMF_Results/'.

Besides, we also provide an executable version of MIA that does not require a MATLAB license. Next, we will describe how to define its input data and what outputs include. Its detailed implementation can be found in Supplementary Materials.

Input Data

MIA implements all four methods using the same structure variable to describe input data. This variable, named *Input*, includes the following components (Supplementary Materials):

Input.data: A matrix storing all the multi-dimensional data sequentially (e.g., $\text{Input.data} = [X_1, \dots, X_N]$). Each row corresponds to a specific sample and each column to a feature. The set of column indexes for each type of genomic data are recorded in *Input.XBlockInd*.

Input.XBlockInd: A matrix of size $N \times 2$. The two elements in the i -th row give the start and end column indexes in *Input.data* for the i -th matrix - X_i ($i = 1, \dots, N$).

Input.YBlockInd: Its format is similar to *Input.XblockInd*, storing the column indexes of response matrix Y in *Input.data* for sMBPLS and SNPLS methods.

Input.netAdj: A symmetric adjacency matrix of a given network used for SNMNMF and SNPLS, where the features have the same order as in *Input.data*. This network combines the interactions between and within the variables in multiple types of data matrices. For example, for SNMNMF, $\text{Input.netAdj} = [A_{11}, A_{12}; A_{12}^T, A_{22}]$, where A_{11}, A_{22} are respectively adjacency matrices for the interaction networks about features in data matrices X_1, X_2 ; A_{12} is for the interaction network between the two types of features. The element of this matrix equals to 1 for linked features in the network, and 0 otherwise.

Input.SampleLabel: A vector recording sample labels.

Input.FeatureLabel: A vector recording feature names in *Input.data*. The i -th label corresponds to the i -th feature in *Input.data*.

Input.FeatureType: A vector recording the feature types in *Input.data*. For example, $\text{Input.FeatureType} = \{\text{'Gene expression'}, \text{'microRNA expression'}, \text{'DNA methylation'}\}$.

Input.params: A structure variable, storing all the parameters used in a specific method (e.g., the pre-defined number of md-modules, the parameters in its objective function).

For these four methods, there are three common parameters, including

- *Input.params.NCluster*: The pre-defined number of md-modules. For example, we may set *Input.params.NCluster* = 50.
- *Input.params.maxiter*: The maximal number of iterations in the algorithm. For example, we may set *Input.params.maxiter* = 100.
- *Input.params.tol*: The precision for convergence of the algorithm. For example, we may set *Input.params.tol* = 10^{-6} .

Besides, each method has its own specific parameters. Here, we take SNMNMF as an example. Others can be found in Supplementary Materials.

For SNMNMF, its specific parameters contain:

- *Input.params.nloop*: The number of repeating times to run this algorithm. To obtain a robust and good solution, this algorithm is run for multiple times repeatedly, and the solution with the minimal value of objective function is accepted. For example, we may set *Input.params.nloop* = 10.
- *Input.params.thrd_module*: A non-negative vector of size 1×3 to select features in md-modules. *Input.params.thrd_module(i+1)* is the threshold for selecting the *i*-th type of features in *Input.data* ($i = 1, 2$). The first one is for selecting samples. The larger they are, the smaller number of features are selected. Users can set it based on the proper size of md-modules they think. For example, we may set *Input.params.thrd_module* = [1,1,1].
- *Input.params.thrNet11*, *Input.params.thrNet12*, *Input.params.thrNet22*: Three non-negative parameters are respectively set for the network constraints about network A_{11} , A_{12} , A_{22} in the objective function. User can choose networks they prefer to use in the framework by setting the corresponding parameters non-zero. For example, if *Input.params.thrNet11* = 0, network A_{11} will not be used.
- *Input.params.thrXr*, *Input.params.thrXc*: Two non-negative numbers are set for W -related and H_i -related constraints respectively in the objective function. They control the degree of sparsity of matrices W and H_i . For example, we may set *Input.params.thrXr* = 10, *Input.params.thrXc* = 10.

In addition, for the components that are not used in certain methods (e.g., *Input.YBlockInd* in jNMF and SNMNMF and *Input.netAdj* in jNMF and sMBPLS), users can set them null or just ignore them.

We note that MIA is able to partition *Input.data* into corresponding data matrices as input for each method automatically. We also provide a set of demo data in the folder named *InputData*. Here, we provide an example for constructing the input data used in SNMNMF. Suppose that one wants to identify 50 microRNA-gene co-modules by integrating gene expression profiles ($X_1 \in \mathbb{R}^{385 \times 12456}$) and microRNA expression profiles ($X_2 \in \mathbb{R}^{385 \times 559}$) across the same set of tumor samples, as well as gene interaction network G_1 and gene-microRNA interaction network G_2 . Network G_1 can be expressed by adjacency matrix $A_{11} = (a_{ij})_{12456 \times 12456}$, where $a_{ij} = 1$ if gene *i* and gene *j* are linked in network G_1 . Similarly, G_2 is expressed by adjacency matrix $A_{12} \in \mathbb{R}^{12456 \times 559}$. If microRNA interaction network is

not available, the corresponding adjacency matrix A_{22} is defined as $A_{22} = 0 \in \mathbb{R}^{559 \times 559}$.

For example, we could define input data *Input* as below and then save it as '*InputDataForSNMNMF.mat*' (Figure S1):

```
Input.data = [X1, X2];
Input.XBlockInd = [1,12456;12457,13015];
Input.YBlockInd = [];
Input.netAdj = [A11, A12; A12^T, A22];
Input.SampleLabel = {'TCGA-24-1105-01A'; ...;
                    'TCGA-13-0793-01A'};
Input.FeatureLabel = {'SFRS8'; ...; 'SCN3A'; 'hsa-mir-488';
                    ...; 'hsa-mir-874'};
Input.FeatureType = {'Gene'; 'miRNA'};
Input.params.NCluster = 50;
Input.params.maxiter = 100;
Input.params.tol = 10^-6;
Input.params.nloop = 10;
Input.params.thrd_module = [1,1,1];
Input.params.thrNet11 = 10^-4; Input.params.thrNet12 = 0.01;
Input.params.thrNet22 = 0;
Input.params.thrXr = 10; Input.params.thrXc = 10;
```

Output Results

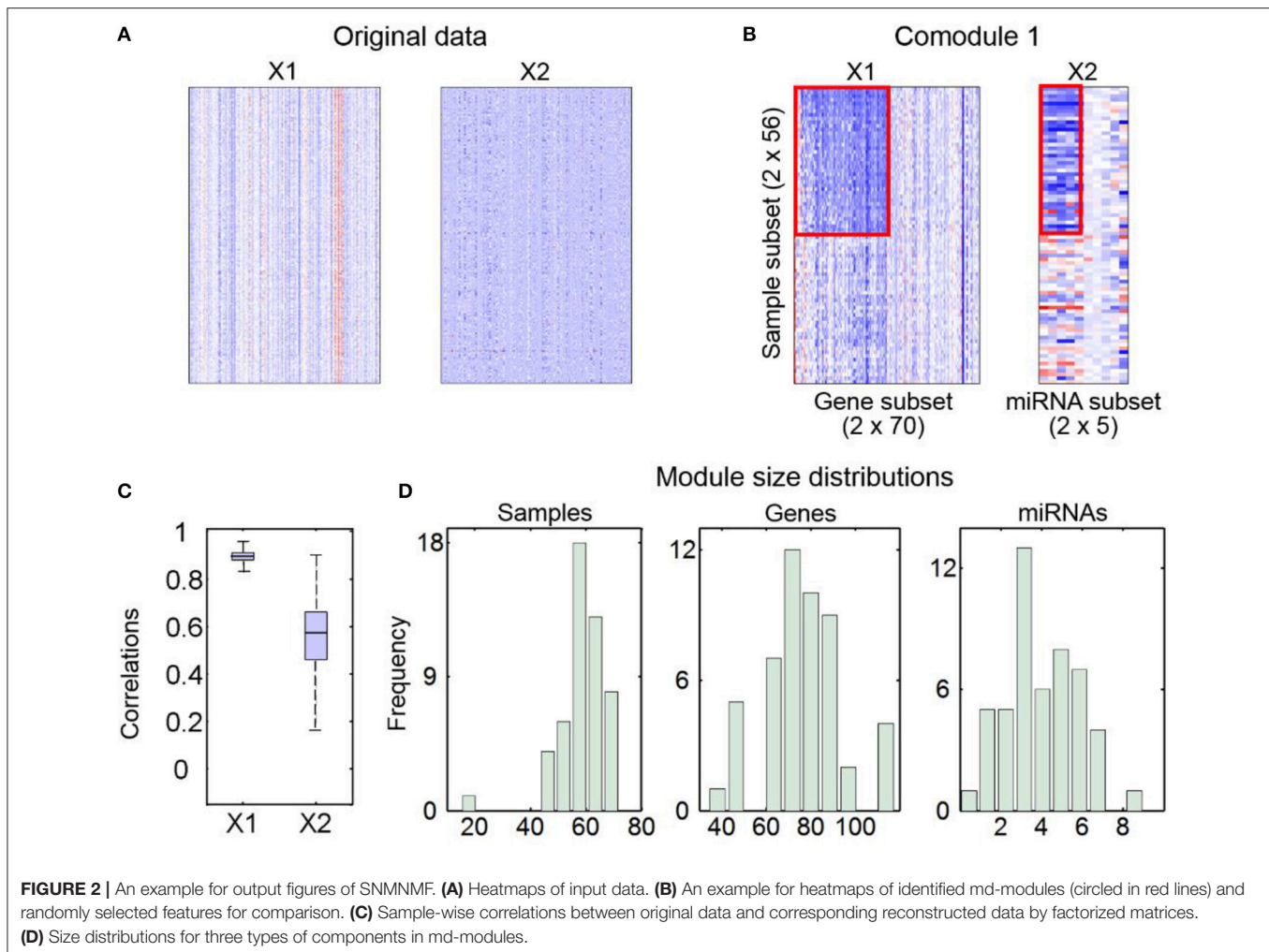
Given input data and desired method, MIA automatically performs all computations and saves all the results in a specific folder named ****_Results*, where ***** represents the pre-selected method name. Here, we continue taking SNMNMF for example (for other methods, please refer to Supplementary Materials). We load the data constructed above and type these commands in the command window of MATLAB:

```
>> load('InputDataForSNMNMF.mat','Input')
>> MIA(Input, 'SNMNMF');
```

All the results are saved in the directory '*MIA/SNMNMF/SNMNMF_Results*'. Referring to the results, there are four parts (Figures S2–S4):

- 1) The first part is a MATLAB data file, named '*SNMNMF_Results.mat*', saving key variables calculated by this pre-selected method.
- 2) The second part contains some figures, including heatmaps of input data (**Figure 2A**) and identified md-modules (**Figure 2B**), sample-wise correlations between input data and reconstructed data (**Figure 2C**) and size distributions for multi-type module members (**Figure 2D**).
- 3) The third part includes two text files, recording module members and objective function values during iterations.
- 4) The last part includes some folders, named as ****Lists*, where ***** represents the feature type defined by input variable '*Input.FeatureType*' (e.g., "Gene," "microRNA"). In each folder, there are a number of text files, each of which is a list of one type of components in one identified md-module.

Users can further analyze the biological significance of identified md-modules. For example, using the signals calculated by MIA,



they can stratify patients into groups and then conduct Kaplan-Meier survival analysis to judge whether this module is related to clinical characteristics. They can also perform various functional enrichment analysis on the set of md-modules to gain insights into their functions. They may also conduct network analysis using tools like IPA, and/or construct a multi-level network for each md-module to explore the underlying relationships among features further.

RESULTS

Biological Applications

A number of biological analysis tools have been designed for one or two types of genomic data (Shen et al., 2009; Xu et al., 2015; Kowalski et al., 2016; Qin et al., 2016). Systematic analysis of multiple types of data for discovering biological relevant combinatorial patterns are currently limited. Here, the MIA package enables users to study the complex relationships and/or modular characteristics among multiple types of variables by integrating diverse types of large-scale omics data. For example, Zhang et al. applied jNMF to DNA methylation profiles, microRNA expression, and gene expression data from TCGA ovarian cancer dataset (Zhang et al., 2012). They identified

a number of md-modules consisting of mRNA, microRNA, and methylation markers. These md-modules reveal multi-level vertical associations and cooperative functional effects. Besides, these md-modules can stratify patients into groups with distinct clinical characteristics.

Zhang et al. used SNMNMf to identify microRNA-gene regulatory co-modules by integrating gene and microRNA expression profiles as well as gene-gene and microRNA-gene target interaction networks (Zhang et al., 2011). The identified co-modules are enriched in known crucial functional sets, such as nuclear division, immune system process, and so on. Meanwhile, the co-modules are enriched with cancer-regulated genes and microRNA, suggesting the co-modules have strong implication in cancer. Furthermore, from the perspective of network analysis, these co-modules shed light on regulatory circuits.

Li et al. utilized sMBPLS to integrate four types of genomic data and identified multi-layer gene regulatory modules, including copy number variations, DNA methylation markers, microRNAs and genes, each of which constructs a local “gene expression factory” (Li et al., 2012). These modules reveal synergistic functions across multiple dimensions and facilitate regulatory analysis via using the modules to build multi-layer molecular interaction network.

Chen and Zhang applied SNPLS to gene expressions, drug response and gene network data to study (anti-)correlated gene-drug associations (Chen and Zhang, 2016). They identified a number of gene-drug co-modules, which show distinct biological relevance and significant drug-gene connections. These gene-drug co-modules reveal multiple-to-multiple relationships between drugs and their targets, and provide us insights into potential drug targets and drug combinations for cancer therapy.

In addition, MIA can have more applications. For example, it can discover relationships between species by integrating their gene expression profiles across multiple conditions. MIA can also predict transcription factor activities from combined analysis of microarray and ChIP-seq data. In summary, MIA is a flexible tool which can be applied to many problems involving diverse type of features of the same sets of samples.

These four methods of MIA can discover joint modular patterns when applied to distinct types of genomic data. They can be alternatively used in many situations. However, they also have different characteristics. Firstly, depending on whether users have network knowledge, users can select the methods with network constraints (SNMNMf and SNPLS) or without (jNMF and sMBPLS). Secondly, PLS-based methods emphasize regression analysis while NMF-based ones emphasize the identification of local patterns among data. Just like applications of jNMF (Zhang et al., 2012) and SNMNMf (Zhang et al., 2011), the integrative genomic data blocks are regarded equally. We can infer the identified genomic markers function cooperatively in biological system, but could not conclude that the dysregulation of some features leads to activity change of the others. Contrasting to NMF-based methods, PLS-based methods aim to describe the relationships between predictors in X and response variables in Y via regression analysis, expecting to predict how response variables change when predictors change, where X and Y are not treated symmetrically. For example, in Li et al. (2012), the identified modules by sMBPLS actually indicate the gene components are regulated by other components—copy number variations, DNA methylation markers and microRNAs, whose changes probably contribute to the changes of gene expression. Thus, before choosing method, we should have an assessment for relationships between data blocks. Thirdly, the constraints for these two classes of methods are distinct. NMF-based methods require non-negativity for input data matrices. For normalized genomic data, in some cases, they contain negative elements, and need to pre-process to satisfy the non-negativity constraint, which is unavoidable to change original data structure more or less, and leads to some effects on results. As for PLS-based methods, all the data blocks need to be centered across samples, which is a common way of normalization for genomic data.

We are eager to show the differences of NMF- and PLS-based methods in detail by applying them to real genomic data. However, it is hard to evaluate their performances, since the golden standards for real data are lacking. Thus, we employ an alternative way to show their difference based on simulated data.

Simulation Study

Here, we apply jNMF and sMBPLS to a set of simulated data matrices X and Y to demonstrate key differences between NMF- and PLS-based methods (Supplementary Materials). We believe

that NMF-based methods tend to identify substructures with high absolute signals, whereas PLS-based ones prefer to discover patterns with strong correlations. Thus, we construct simulated data sets embedding two types of co-module structures—one type is module members with high absolute values across the same subset of samples; the other type is members with strong correlations. According to our assumption that these two types of methods have different preferences when detecting co-modules, two different types of golden standards for co-module members ($G^{(1)}$ and $G^{(2)}$) that jNMF and sMBPLS should identify respectively are created. We name $G^{(1)}$ as NMF-based standard and $G^{(2)}$ as PLS-based standard. If the co-modules identified by jNMF more approximate to $G^{(1)}$ than those by sMBPLS and the co-modules by sMBPLS are more close to $G^{(2)}$ than jNMF, it will verify the differences between these two types of methods as what we think. In the simulated data matrices X and Y , we embed five co-modules (Supplementary Materials), thus $G^{(i)} = \{G_1^{(i)}, G_2^{(i)}, \dots, G_5^{(i)}\}$, $i = 1, 2$, where $G_k^{(i)}$ is the member set for the k -th co-module in the NMF- or PLS-based standard. We use relevance score to measure the degree of similarity between the real co-modules $G^{(i)}$ and the identified ones $M = \{M_1, M_2, \dots, M_5\}$ by jNMF or sMBPLS:

$$\text{Relevance}(M, G^{(i)}) = \frac{1}{5} \sum_{M_j \in M} \max_{G_k^{(i)} \in G^{(i)}} s(M_j, G_k^{(i)}),$$

$$s(M_j, G_k^{(i)}) = \frac{|M_j \cap G_k^{(i)}|}{|M_j \cup G_k^{(i)}|},$$

where $|M_j \cap G_k^{(i)}|$ is the number of features in their intersection, and $|M_j \cup G_k^{(i)}|$ is the number of features in their union.

We can clearly see that the relevance scores of jNMF are higher than sMBPLS under NMF-based standard (Figure 3A, Figures S7A–S11A), which shows the co-modules identified by jNMF are more similar to $G^{(1)}$ than sMBPLS. Thus, jNMF is indeed able to discover some patterns that sMBPLS could not find well. For example, for the first embedded module in simulated data Y (i.e., features from the first to 100th column and from the 201st to 230th column), the first 100 components are highly positive or negative correlated with each other, but among them, the last 50 ones have rather low signals. The other 30 components (from the 201st to 230th column) have high signal with similar magnitude to the first 50 components, but they have weak correlations with the first 100 components. It shows jNMF identifies features from the first to 50th column and from the 201st to 230th column, whereas sMBPLS discovers those from the first to 100th column. On the other hand, sMBPLS performs better than jNMF under PLS-based standard (Figure 3B, Figures S7B–S11B). sMBPLS could identify highly correlated components, not only for samples but also for features in X and Y , although they may have slightly low signals, which jNMF often disregards. For example, the second co-module (i.e., samples from the 111st to 200th row and features from the 201st to 400th column of matrix X and those from the 101st to 200th column of matrix Y) has very low signal values across the first 40 samples in matrices X and Y , but these samples are closely

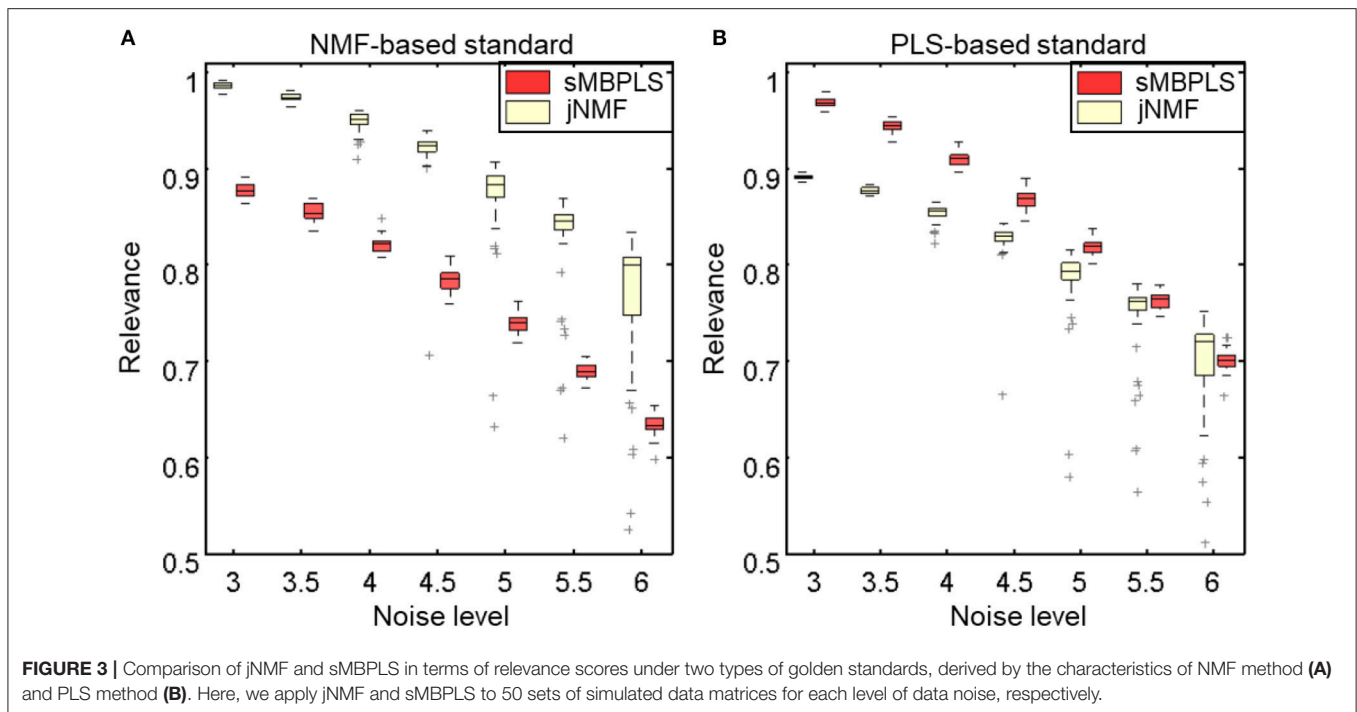


FIGURE 3 | Comparison of jNMF and sMBPLS in terms of relevance scores under two types of golden standards, derived by the characteristics of NMF method (A) and PLS method (B). Here, we apply jNMF and sMBPLS to 50 sets of simulated data matrices for each level of data noise, respectively.

negatively correlated with the other 50 samples. As a result, sMBPLS discovers all these 90 samples as components of the second co-module, while jNMF just identifies the last 50 samples.

Although the accuracy of identified co-modules by jNMF and sMBPLS both decreases with the data noises increasing, the relevance scores for sMBPLS decreases faster than jNMF, which may suggest that sMBPLS is more sensitive to data noise than jNMF. In particular, for the last co-module, the golden standards for both methods are the same, but jNMF performs better than sMBPLS under different levels of data noises (Figure S11). We guess that high correlation pattern is much easier to be affected compared to high signal one with the increase of data noise.

Besides, the scores for sMBPLS gradually approach to jNMF with data noise increasing (Figure 3B). Correlations between features of relatively high signals are more difficult to be weakened by the same level of data noise than those of low signals. Thus, with data noise increasing, jNMF and sMBPLS tend to find similar co-modules with large signals and high correlations.

CONCLUSION AND DISCUSSION

The advance of biological techniques makes multi-dimensional large-scale omics data available, which provides us opportunities to study the complex biological system from the perspective of multi-layer regulatory programs. Therefore, it is necessary to develop a tool to simultaneously integrate multi-dimensional biomedical data. Here, we present a MATLAB package, MIA, to conduct integrative and modular analysis for multi-dimensional genomic data across the same sets of samples as well as prior network knowledge to decode the relationships among different levels of cellular activities.

Although these four methods in the MIA package are all used to identify modular patterns in data matrices, users could select an appropriate one based on their own purpose and data characteristics. We have demonstrated the specificity of these two types of methods using simulation test, which also provides clues about the method applicability. For example, NMF-based methods tend to identify local patterns with high absolute signals, whereas PLS-based methods are essentially regression methods, thus they are likely to extract the components with strong correlations between predictors and response variables.

Since these problems are not convex, it is very hard to find a global optimal solution. For NMF-based methods, users could run the algorithm for several times repeatedly by setting the input parameter *Input.params.nloop* in MIA, and then the solution with the minimal value of objective function is used for further analysis. Thus, users could obtain a consistent solution by this strategy. When solving the optimization problems for PLS-based methods, the initial iteration point of algorithm have much effects on the solution. A good starting point helps the algorithm converge rapidly. In MIA, instead of generating a random starting point, we choose the solution of the original PLS problem without any constraints as an initial point to speed up the algorithm and meanwhile make its solution more stable.

MIA may run slowly when applying to multiple large-scale data sets. Thus, it is better to filter some features by data pre-processing to reduce the dimensions of input data. In the future, we will further optimize the programs in MIA to make it more efficient for large-scale data integrative analysis. In MIA, we respectively choose the most common used algorithms to solve them. We use multiplicative update rules for NMF-based methods, and non-linear iterative partial least squares (NIPLS) algorithm for PLS-based methods. We will provide users several other efficient optimization algorithms for choice. Besides, R language is widely used among researchers, thus we

will prepare the version implemented in R code soon to facilitate its adoption.

In summary, MIA is very easy to use and does not require high-level programming skills. We expect it will become a routine exploratory tool for diverse biological problems.

AVAILABILITY

The MIA package is available at <http://page.amss.ac.cn/shihua.zhang/software.html>.

REQUIREMENTS

It works in MATLAB R2013a or later in Windows (64-bit). For users without a MALTB license, MATLAB R2015b Runtime for Windows (64-bit) is required to install it.

AUTHOR CONTRIBUTIONS

JC and SZ designed this study. JC and SZ performed the experiments and statistical analysis. JC and SZ interpreted the results and written the manuscript. All authors read and approved the final manuscript.

REFERENCES

- Alexandrov, L. B., Nik-Zainal, S., Wedge, D. C., Aparicio, S. A., Behjati, S., Biankin, A. V., et al. (2013a). Signatures of mutational processes in human cancer. *Nature* 500, 415–421. doi: 10.1038/nature12477
- Alexandrov, L. B., Nik-Zainal, S., Wedge, D. C., Campbell, P. J., and Stratton, M. R. (2013b). Deciphering signatures of mutational processes operative in human cancer. *Cell Rep.* 3, 246–259. doi: 10.1016/j.celrep.2012.12.008
- Boulesteix, A., Durif, G., Lambert-Lacroix, S., Peyre, J., and Strimmer, K. (2015). *plsgenomics: PLS Analyses for Genomics*. R package version 1.3–1.
- Boulesteix, A., and Strimmer, K. (2005). Predicting transcription factor activities from combined analysis of microarray and ChIP data: a partial least squares approach. *Theor. Biol. Med. Model.* 2:23. doi: 10.1186/1742-4682-2-23
- Brunet, J. P., Tamayo, P., Golub, T. R., and Mesirov, J. P. (2004). Metagenes and molecular pattern discovery using matrix factorization. *Proc. Natl. Acad. Sci. U. S. A.* 101, 4164–4169. doi: 10.1073/pnas.0308531101
- Cai, D., He, X. F., Wu, X. Y., and Han, J. (2008). “Non-negative matrix factorization on manifold,” in *ICDM 2008: The 8th IEEE International Conference on Data Mining, Proceedings* (Pisa), 63–72.
- Chen, J., and Zhang, S. (2016). Integrative analysis for identifying joint modular patterns of gene-expression and drug-response data. *Bioinformatics* 32, 1724–1732. doi: 10.1093/bioinformatics/btw059
- Chun, H., and Keles, S. (2010). Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *J. R. Stat. Soc. Ser. B Stat. Methodol.* 72, 3–25. doi: 10.1111/j.1467-9868.2009.00723.x
- Chung, D., Chun, H., and Keles, S. (2013). *Sparse Partial Least Squares (SPLS) Regression and Classification*. R package version 2.2.
- Chung, D., and Keles, S. (2010). Sparse partial least squares classification for high dimensional data. *Stat. Appl. Genet. Mol. Biol.* 9, 17. doi: 10.2202/1544-6115.1492
- Cichocki, A., Lee, H., Kim, Y. D., and Choi, S. (2008). Non-negative matrix factorization with alpha-divergence. *Pattern Recogn. Lett.* 29, 1433–1440. doi: 10.1016/j.patrec.2008.02.016
- Cichocki, A., Zdunek, R., and Amari, S. (2006). “Csiszár’s divergences for non-negative matrix factorization: family of new algorithms,” in *The 6th International Conference on Independent Component Analysis and Blind Signal Separation Proceedings* (Charleston, SC), 32–39.

FUNDING

This work has been supported by the National Natural Science Foundation of China (No. 11131009, 61379092, 61422309, and 61621003), the Strategic Priority Research Program of the Chinese Academy of Sciences (CAS) (No. XDB13040600), the Key Research Program of the Chinese Academy of Sciences (No. KFZD-SW-219), National Key Research and Development Program of China (No. 2017YFC0908405), and CAS Frontier Science Research Key Project for Top Young Scientist (No. QYZDB-SSW-SYS008).

ACKNOWLEDGMENTS

We appreciate Prof Xianghong Jasmine Zhou at University of California, Los Angeles for her generous help on the MIA package.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/fgene.2018.00194/full#supplementary-material>

- Cummins, D. J., and Andrews, C. W. (1995). Iteratively reweighted partial least-squares - a performance analysis by monte-carlo simulation. *J. Chemometr.* 9, 489–507. doi: 10.1002/cem.1180090607
- Devarajan, K. (2008). Nonnegative matrix factorization: an analytical and interpretive tool in computational biology. *PLoS Comput. Biol.* 4:e1000029. doi: 10.1371/journal.pcbi.1000029
- Dhillon, I. S., and Sra, S. (2006). “Generalized nonnegative matrix approximations with bregman divergences,” in *The 18th International Conference on Neural Information Processing Systems Proceedings* (Vancouver, BC), 283–290.
- Ding, B. (2016). *gpls: Classification Using Generalized Partial Least Squares*. R package version 1.38.30.
- Ding, C., Li, T., Peng, W., and Park, H. (2006). “Orthogonal nonnegative matrix tri-factorizations for clustering,” in *The 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Proceedings* (Philadelphia, IL), 126–135.
- Feng, T., Li, S. Z., Shum, H. Y., and Zhang, H. J. (2002). “Local non-negative matrix factorization as a visual representation,” in *The 2nd International Conference on Development and Learning Proceedings* (Cambridge, MA), 178–183.
- Fevotte, C., Bertin, N., and Durrieu, J. L. (2009). Nonnegative matrix factorization with the Itakura-Saito divergence: with application to music analysis. *Neural Comput.* 21, 793–830. doi: 10.1162/neco.2008.04-08-771
- Gaujoux, R., and Seoighe, C. (2010). A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics* 11:367. doi: 10.1186/1471-2105-11-367
- Gil, J. A., and Romera, R. (1998). On robust partial least squares (PLS) methods. *J. Chemometr.* 12, 365–378. doi: 10.1002/(SICI)1099-128X(199811/12)12:6<365::AID-CEM519>3.0.CO;2-G
- Gutkin, M., Shamir, R., and Dror, G. (2009). SlimPLS: a method for feature selection in gene expression-based disease classification. *PLoS ONE* 4:e6416. doi: 10.1371/journal.pone.0006416
- Huang, X., Pan, W., Park, S., Han, X., Miller, L. W., and Hall, J. (2004). Modeling the relationship between LVAD support time and gene expression changes in the human heart by penalized partial least squares. *Bioinformatics* 20, 888–894. doi: 10.1093/bioinformatics/btg499
- Hubert, M., and Vanden Branden, K. (2003). Robust methods for partial least squares regression. *J. Chemometr.* 17, 537–549. doi: 10.1002/cem.822
- Kasar, S., Kim, J., Improgo, R., Tiao, G., Polak, P., Haradhvala, N., et al. (2015). Whole-genome sequencing reveals activation-induced cytidine deaminase

- signatures during indolent chronic lymphocytic leukaemia evolution. *Nat. Commun.* 6:8866. doi: 10.1038/ncomms9866
- Kim, H., and Park, H. (2007). Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics* 23, 1495–1502. doi: 10.1093/bioinformatics/btm134
- Kim, P. M., and Tidor, B. (2003). Subsystem identification through dimensionality reduction of large-scale gene expression data. *Genome Res.* 13, 1706–1718. doi: 10.1101/gr.903503
- Kowalski, J., Dwivedi, B., Newman, S., Switchenko, J., Pauly, R., and Gutman, D. (2016). Gene integrated set profile analysis: a context-based approach for inferring biological endpoints. *Nucleic Acids Res.* 44, e69. doi: 10.1093/nar/gkv1503
- Lê Cao, K. A., Boitard, S., and Besse, P. (2011). Sparse PLS discriminant analysis: biologically relevant feature selection and graphical displays for multiclass problems. *BMC Bioinformatics* 12:253. doi: 10.1186/1471-2105-12-253
- Lê Cao, K. A., Gonzalez, I., and Dejean, S. (2009). Integromics: an R package to unravel relationships between two omics datasets. *Bioinformatics* 25, 2855–2856. doi: 10.1093/bioinformatics/btp515
- Lê Cao, K. A., Rossouw, D., Robert-Granie, C., and Besse, P. (2008). A sparse PLS for variable selection when integrating omics data. *Stat. Appl. Genet. Mol. Biol.* 7:35. doi: 10.2202/1544-6115.1390
- Lee, D. D., and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 788–791.
- Lee, D. D., and Seung, H. S. (2000). “Algorithms for non-negative matrix factorization,” in *The 13th International Conference on Neural Information Processing Systems, Proceedings* (Denver, CO), 556–562.
- Li, W., Zhang, S., Liu, C. C., and Zhou, X. J. (2012). Identifying multi-layer gene regulatory modules from multi-dimensional genomic data. *Bioinformatics* 28, 2458–2466. doi: 10.1093/bioinformatics/bts476
- Li, Y., and Ngom, A. (2013). The non-negative matrix factorization toolbox for biological data mining. *Source Code Biol. Med.* 8:10. doi: 10.1186/1751-0473-8-10
- Liquet, B., de Micheaux, P. L., Hejblum, B. P., and Thiebaut, R. (2016). Group and sparse group partial least squares approaches applied in genomics context. *Bioinformatics* 32, 35–42. doi: 10.1093/bioinformatics/btv535
- Mevik, B.-H., and Wehrens, R. (2007). The pls package: principal component and partial least squares regression in R. *J. Stat. Softw.* 18, 23. doi: 10.18637/jss.v018.i02
- Moffitt, R. A., Marayati, R., Flate, E. L., Volmar, K. E., Loeza, S. G., Hoadley, K. A., et al. (2015). Virtual microdissection identifies distinct tumor- and stroma-specific subtypes of pancreatic ductal adenocarcinoma. *Nat. Genet.* 47, 1168–1178. doi: 10.1038/ng.3398
- Morine, M. J., Tierney, A. C., van Ommen, B., Daniel, H., Toomey, S., Gjelstad, I. M. F., et al. (2011). Transcriptomic coordination in the human metabolic network reveals links between n-3 fat intake, adipose tissue gene expression and metabolic health. *PLoS Comput. Biol.* 7:e1002223. doi: 10.1371/journal.pcbi.1002223
- Nguyen, D. V., and Rocke, D. M. (2002a). Multi-class cancer classification via partial least squares with gene expression profiles. *Bioinformatics* 18, 1216–1226. doi: 10.1093/bioinformatics/18.9.1216
- Nguyen, D. V., and Rocke, D. M. (2002b). Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics* 18, 39–50. doi: 10.1093/bioinformatics/18.1.39
- Nik-Zainal, S., Alexandrov, L. B., Wedge, D. C., Van Loo, P., Greenman, C. D., Raine, K., et al. (2012). Mutational processes molding the genomes of 21 breast cancers. *Cell* 149, 979–993. doi: 10.1016/j.cell.2012.04.024
- Paatero, P., and Tapper, U. (1994). Positive matrix factorization - a nonnegative factor model with optimal utilization of error-estimates of data values. *Environmetrics* 5, 111–126. doi: 10.1002/env.3170050203
- Pascual-Montano, A., Carmona-Saez, P., Chagoyen, M., Tirado, F., Carazo, J. M., and Pascual-Marqui, R. D. (2006). bioNMF: a versatile tool for non-negative matrix factorization in biology. *BMC Bioinformatics* 7:366. doi: 10.1186/1471-2105-7-366
- Peharz, R., and Pernkopf, F. (2012). Sparse nonnegative matrix factorization with L0-constraints. *Neurocomputing* 80, 38–46. doi: 10.1016/j.neucom.2011.09.024
- Pei, Y., Chakraborty, N., and Sycara, K. (2015). “Nonnegative matrix tri-factorization with graph regularization for community detection in social networks,” in *The 24th International Conference on Artificial Intelligence Proceedings* (Buenos Aires), 2083–2089.
- Qi, Q., Zhao, Y., Li, M., and Simon, R. (2009). Non-negative matrix factorization of gene expression profiles: a plug-in for BRB-arraytools. *Bioinformatics* 25, 545–547. doi: 10.1093/bioinformatics/btp009
- Qin, J., Yan, B., Hu, Y., Wang, P., and Wang, J. (2016). Applications of integrative OMICs approaches to gene regulation studies. *Quant. Biol.* 4, 283–301. doi: 10.1007/s40484-016-0085-y
- Ray, S., and Bandyopadhyay, S. (2016). A NMF based approach for integrating multiple data sources to predict HIV-1-human PPIs. *BMC Bioinformatics* 17:121. doi: 10.1186/s12859-016-0952-6
- The Cancer Genome Atlas Research Network (2008). Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature* 455, 1061–1068. doi: 10.1038/nature07385
- Rosipal, R., and Kramer, N. (2006). Overview and recent advances in partial least squares. *Subspace Latent Struct. Feature Select.* 3940, 34–51. doi: 10.1007/11752790_2
- Serneels, S., Croux, C., Filzmoser, P., and Van Espen, P. J. (2005). Partial robust M-regression. *Chemometr. Intell. Lab. Syst.* 79, 55–64. doi: 10.1016/j.chemolab.2005.04.007
- Shen, R., Olshen, A., and Ladanyi, M. (2009). Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis. *Bioinformatics* 25, 2906–2912. doi: 10.1093/bioinformatics/btp543
- Singh, P., Engel, J., Jansen, J., de Haan, J., and Buydens, L. M. (2016). Dissimilarity based partial least squares (DPLS) for genomic prediction from SNPs. *BMC Genomics* 17:324. doi: 10.1186/s12864-016-2651-0
- Strazar, M., Zitnik, M., Zupan, B., Ule, J., and Curk, T. (2016). Orthogonal matrix factorization enables integrative analysis of multiple RNA binding proteins. *Bioinformatics* 32, 1527–1535. doi: 10.1093/bioinformatics/btw003
- Wang, F., Li, T., and Zhang, C. (2008). “Semi-supervised clustering via matrix factorization,” in *The 2008 SIAM International Conference on Data Mining Proceedings* (Atlanta, GA), 1–12.
- Wold, H. (1975). “Path models with latent variables: the NIPALS approach,” in *International Perspectives on Mathematical and Statistical Modeling*, ed H. M. Blalock, A. Aganbegian, F. M. Borodkin, R. Boudon, and V. Capecchi (New York, NY: Academic Press), 307–357. doi: 10.1016/B978-0-12-103950-9.50017-4
- Wold, S., Sjostrom, M., and Eriksson, L. (2001). PLS-regression: a basic tool of chemometrics. *Chemometr. Intell. Lab. Syst.* 58, 109–130. doi: 10.1016/S0169-7439(01)00155-1
- Wu, S., Joseph, A., Hammonds, A. S., Celniker, S. E., Yu, B., and Frise, E. (2016). Stability-driven nonnegative matrix factorization to interpret spatial gene expression and build local gene networks. *Proc. Natl. Acad. Sci. U.S.A.* 113, 4290–4295. doi: 10.1073/pnas.1521171113
- Xu, T., Li, B., Zhao, M., Szulwach, K., Street, R., Lin, L., et al. (2015). Base-resolution methylation patterns accurately predict transcription factor bindings *in vivo*. *Nucleic Acids Res.* 43, 2757–2766. doi: 10.1093/nar/gkv151
- Zhang, S., Li, Q., Liu, J., and Zhou, X. J. (2011). A novel computational framework for simultaneous integration of multiple types of genomic data to identify microRNA-gene regulatory modules. *Bioinformatics* 27, 401–409. doi: 10.1093/bioinformatics/btr206
- Zhang, S., Liu, C. C., Li, W., Shen, H., Laird, P. W., and Zhou, X. J. (2012). Discovery of multi-dimensional modules by integrative analysis of cancer genomic data. *Nucleic Acids Res.* 40, 9379–9391. doi: 10.1093/nar/gks725
- Zitnik, M., Nam, E. A., Dinh, C., Kuspa, A., Shaulsky, G., and Zupan, B. (2015). Gene prioritization by compressive data fusion and chaining. *PLoS Comput. Biol.* 11:e1004552. doi: 10.1371/journal.pcbi.1004552
- Zitnik, M., and Zupan, B. (2015). Data fusion by matrix factorization. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 41–53. doi: 10.1109/TPAMI.2014.2343973

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2018 Chen and Zhang. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.