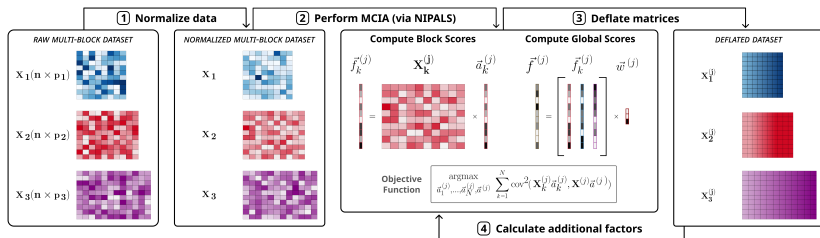


# nipalsMCIA: Flexible Multi-Block Dimensionality Reduction in R via Nonlinear Iterative Partial Least Squares

Anna Konstorum

Center for Computing Sciences, Institute for Defense Analyses

March 20, 2025



# Acknowledgments

*Yale University*  
*Yale School of Medicine*



*Department of Pathology*  
Edel Aron  
Steven Kleinstein

*Tufts University*



*Department of Mathematics*  
Maximilian Mattessich (now at Northwestern)  
Misha Kilmer

*University of California, San Diego*



*La Jolla Institute for Immunology*  
Joaquin Reyna (now at VCU)  
Ferhat Ay

**NIH Grants**  
U19 AI 089992  
R21AI176204



National Institute of  
Allergy and  
Infectious Diseases

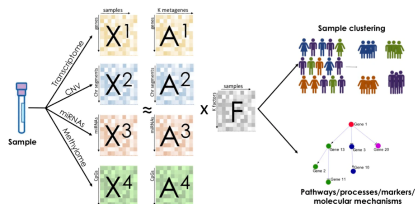


CENTER FOR COMPUTING SCIENCES



# Multi-view dimension reduction in biology

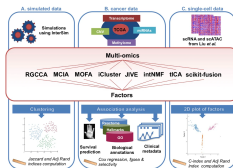
- It is now commonplace to collect multi-omics data (such as mRNA, microRNA, DNA methylation, proteomics) from bulk and single-cell tissue sources.
- Crucial objectives for analysis of multi-omic data include: (1) classifying and clustering samples; (2) understanding potential underlying molecular determinants of subtype segregation; (3) predicting clinical outcomes.
- Dimension reduction methods decompose omics into a shared low-dimensional latent space, and can help to achieve objectives of multi-omic studies<sup>1</sup>.



<sup>1</sup>Cantini, Laura, et al. "Benchmarking joint multi-omics dimensionality reduction approaches for the study of cancer." Nature communications 12.1 (2021): 124.

# Multiple co-Inertia Analysis (MCIA)

- Multiple co-Inertia Analysis (MCIA) is one method for multi-view (a.k.a. multi-block, joint dimensionality) reduction.
- It has shown good performance in a heuristic comparison of multi-omic integration methods for multi-omic cancer data (Cantini et al., 2021).
- It is part of a larger family of sequential multiblock component methods: Regularized Canonical Correlation Analysis (RGCCA)<sup>2</sup>.



<sup>2</sup>Tenenhaus, Michel, Arthur Tenenhaus, and Patrick JF Groenen. "Regularized generalized canonical correlation analysis: a framework for sequential multiblock component methods." *Psychometrika* 82.3 (2017): 737-777.

# Notation and generalized objective function for multi-block dimension reduction and MCIA

- Scalars, vectors, and matrices are represented in lowercase script ( $a$ ), lowercase script with a vector symbol ( $\vec{a}$ ), and bold uppercase script ( $\mathbf{A}$ ), respectively. The  $i^{th}$  column vector of a matrix  $\mathbf{A}$  is denoted  $\vec{a}^{(i)}$ .
- In a multi-block dataset, each block  $\mathbf{X}_k \in \mathbb{R}^{n \times p_k}$  where  $n$  is the number of samples, and  $p_k$  the number of features for block  $k$ .
- Take  $\mathbf{X} = [\mathbf{X}_1 | \mathbf{X}_2 | \dots | \mathbf{X}_K]$  as the concatenated dataset with  $K$  blocks.
- For low-dimensional rank  $R$ ,  $\mathbf{X}_k^{(j)} \in \mathbb{R}^{n \times p_k}$ ,  $j = 1, \dots, R$ , is the deflated data matrix at order  $j$ .

# Notation and generalized objective function for multi-block dimension reduction and MCIA

The objective function solved by MCIA is

$$\operatorname{argmax}_{\vec{a}_1^{(j)}, \dots, \vec{a}_N^{(j)}, \vec{a}^{(j)}} \sum_{k=1}^N \operatorname{cov}^2(\mathbf{X}_k^{(j)} \vec{a}_k^{(j)}, \mathbf{X}^{(j)} \vec{a}^{(j)}), \quad (\vec{a}_k^{(i)})^T \vec{a}_k^{(j)} = \delta_{ij}.$$

A few notes,

- It can be shown that  $\vec{f}^{(i)} = \mathbf{X}^{(i)} \vec{a}^{(i)} = \mathbf{X} \vec{a}^{(i)}$  (and similarly for  $\vec{f}_k^{(i)}$ ), therefore we observe that we can write the global scores as  $\mathbf{F} = \mathbf{X} \mathbf{A}$  and the block scores as  $\mathbf{F}_k = \mathbf{X}_k \mathbf{A}_k$ .
- Earlier implementations of MCIA calculate the  $i^{th}$  global loading  $\vec{a}^{(i)}$  via the first principal component loading of the  $i^{th}$  order deflated data matrix  $\mathbf{X}^{(i)}$  (i.e. via direct eigendecomposition).
- Instead of taking  $\operatorname{cov}^2()$ , one can use another function  $g(\mathbf{X}_k, \mathbf{X})$  and constraints on the loadings or global scores (RGCCA).

# Nonlinear Iterative Partial Least Squares (NIPALS)

- Nonlinear Iterative Partial Least Squares (NIPALS) is an iterative method for computing the principal components in Principal Components Analysis (PCA).
  - Each PC is calculated sequentially.
  - Can be modified to accomodate missing values.
- It had been previously been extended to the family of MCIA methods, but convergence was not proven.
- In 2011, Hanafi and Qannari proved<sup>3</sup> monotonic convergence for the extension of NIPALS to MCIA and CPCA (a related method).

---

<sup>3</sup>Hanafi, Mohamed, Achim Kohler, and El-Mostafa Qannari. "Connections between multiple co-inertia analysis and consensus principal component analysis." *Chemometrics and intelligent laboratory systems* 106.1 (2011): 37-40.

# Nonlinear Iterative Partial Least Squares (NIPALS)

Steps for the iteration stage of the modified NIPALS algorithm (adapted from Hanafi and Qannari).

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

Repeat until the global and block scores converge in the chosen convergence metric.

Given a set of block loadings of rank  $j$ ,  $\{\vec{a}_1^{(j)}, \dots, \vec{a}_N^{(j)}\}$ , deflated blocks are computed via  $\mathbf{X}_k^{(j+1)} = \mathbf{X}_k^{(j)} - \mathbf{X}_k^{(j)} \vec{a}_k^{(j)} \left( \vec{a}_k^{(j)} \right)^T$ .

# nipalsMCIA

We introduce `nipalsMCIA`<sup>4</sup>, an implementation in R/Bioconductor of MCIA for joint dimensionality reduction that solves the objective function using NIPALS.

Our contributions:

- Implementation of MCIA using NIPALS.
- Complexity analysis of NIPALS vs. eigendecomposition for MCIA.
- Computation time comparison of `nipalsMCIA` and other implementations for a variety of datasets.
- Return variance explained and options for out-of-sample embedding.
- Ease of use for computational biologists: visualizations, vignettes, and extensive documentation.
- Useful for applied mathematicians who may be interested to extend or experiment with the implementation: the software is build in a modular, well-documented, and accessible fashion.

---

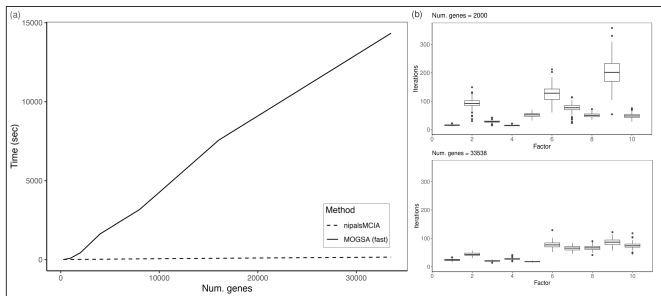
<sup>4</sup>Mattessich, Max, et al. "nipalsMCIA: flexible multi-block dimensionality reduction in R via nonlinear iterative partial least squares." *Bioinformatics* 41.1 (2025): btaf015.

# Complexity analysis and comparison to other methods

Summary of the computational and memory complexities of computing a single MCIA factor on a data matrix  $\mathbf{X} \in \mathbb{R}^{n \times m}$  with  $N$  blocks across three different implementations.  $S$  represents the number of steps for the NIPALS iteration to converge.

| Implementation:           | nipalsMCIA   | omicade4 and MOGSA  | MOGSA (fast svd)       |
|---------------------------|--------------|---------------------|------------------------|
| Computational Complexity: | $O(nmS)$     | $O(nm \min(n, m))$  | $O(\min(n, m)^3 + nm)$ |
| Memory Complexity:        | $O(Nn + Nm)$ | $O(n^2 + m^2 + nm)$ | $O(\min(n, m)^2)$      |

Computation time comparison for nipalsMCIA and MOGSA (fast svd) for decomposition of 10X Genomics PBMC data ( $R = 10$  factors).

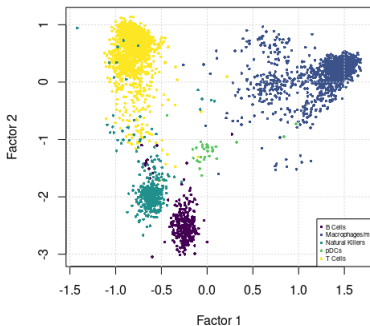


# Computation time comparison

Computation time (in seconds) comparison for different MCIA implementations and datasets.

| Implementation   | Dataset (dimension)              |                                                 |                                              |                                        |
|------------------|----------------------------------|-------------------------------------------------|----------------------------------------------|----------------------------------------|
|                  | NCI60<br>21 x (12895, 547, 7016) | 10x single cell (filtered)<br>4193 x (2000, 32) | 10x single cell (full)<br>4193 x (33538, 32) | HCA single cell<br>145810 x (2000, 35) |
| nipalsMCIA       | 2.3                              | 15.32                                           | 289.46                                       | 671.78                                 |
| MOGSA            | 0.53                             | 519.55                                          | NA                                           | NA                                     |
| MOGSA (fast svd) | 0.38                             | 434.84                                          | 13,840.66                                    | DNC                                    |
| Omicade          | 2.66                             | 1089.53                                         | NA                                           | NA                                     |

Global factor plot of the 10x single cell data using the first two factors, colored by the annotated cell types (mDCs refers to myeloid dendritic cells, pDCs to plasmacytoid dendritic cells).



# Variance explained & out-of-sample embedding

## Property 1

$$\text{var}(\vec{f}^{(i)}) = \lambda^{(i)},$$

where  $\vec{f}^{(i)}$  is the first principal component of  $\mathbf{X}^{(i)}$  and  $\lambda^{(i)}$  is the largest eigenvalue of  $\mathbf{C}_X^{(i)} = \frac{1}{n-1}(\mathbf{X}^{(i)} - \bar{\mathbf{X}}^{(i)})^T(\mathbf{X}^{(i)} - \bar{\mathbf{X}}^{(i)})$ .

## Property 2

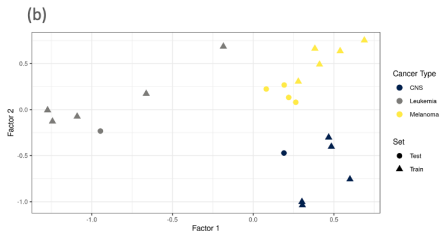
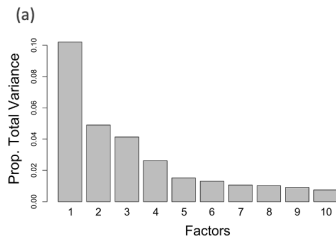
Any set of order- $i$  scores/loadings vectors  $\left\{ \vec{f}_k^{(i)}, \vec{f}_k^{(i)}, \vec{a}_k^{(i)}, \vec{a}_k^{(i)} \mid k = 1, \dots, N \right\}$  computed to satisfy the MCIA criteria satisfy the relations

$$\vec{f}^{(i)} = \mathbf{X}^{(i)} \vec{a}^{(i)} = \mathbf{X} \vec{a}^{(i)} \qquad \vec{f}_k^{(i)} = \mathbf{X}_k^{(i)} \vec{a}_k^{(i)} = \mathbf{X}_k \vec{a}_k^{(i)} \qquad k = 1, \dots, N,$$

where  $\mathbf{X}^{(i)}$  is the data at the  $i^{\text{th}}$  deflation step and  $\mathbf{X}$  is the original data.

# Variance explained & out-of-sample embedding

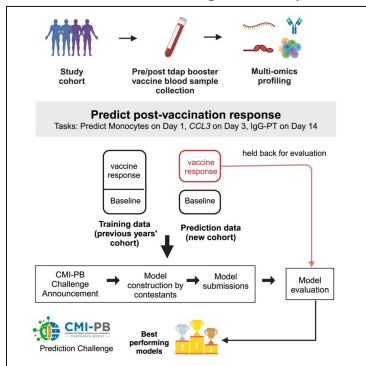
(a) Proportion of variance explained for first 10 factors of 10x single cell data. (b) Global scores for the first two factors of the NCI60 data for both the training (triangle) and test (circle) sets



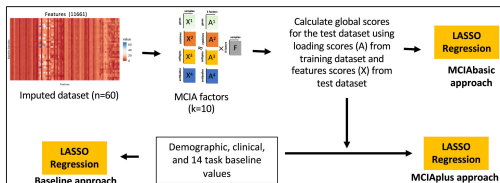
# Early applications (out-of-sample embedding)

- Used out-of-sample MCIA embedding factors as input to a predictive model of response to Pertussis vaccination.

## Prediction challenge summary

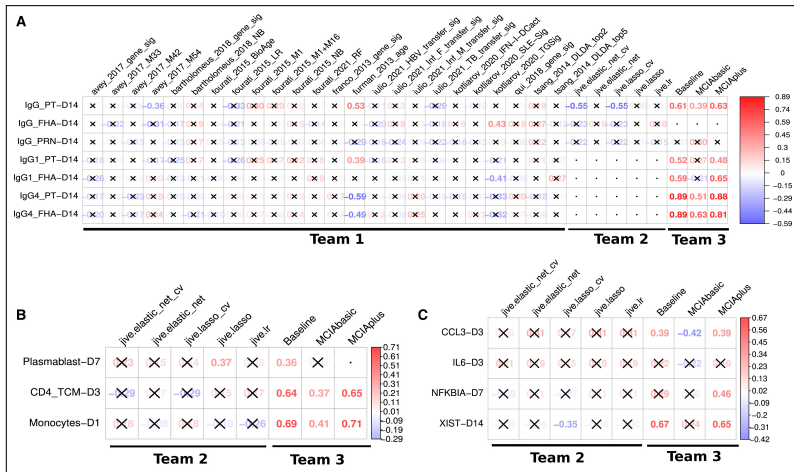


## Semi-supervised predictive model using MCIA (Team 3)



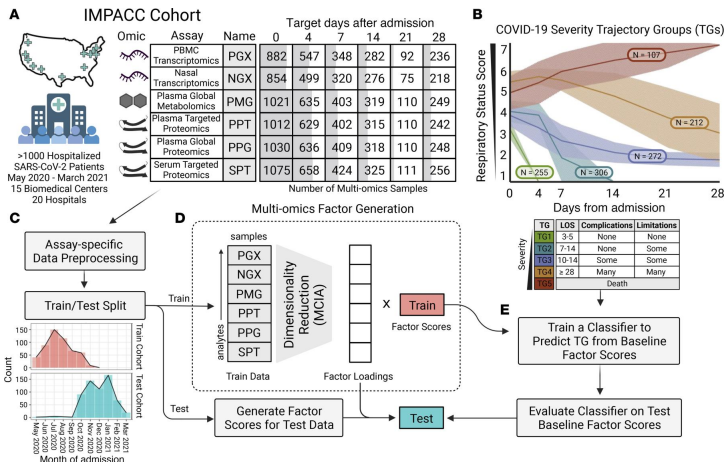
P. Shinde et al. (2024) A multi-omics systems vaccinology resource to develop and test computational models of immunity, Cell Reports Methods 4 (3), 100731.

# Early applications (out-of-sample embedding)



P. Shinde et al. (2024) A multi-omics systems vaccinology resource to develop and test computational models of immunity, Cell Reports Methods 4 (3), 100731.

# Early applications (out-of-sample embedding)



Gygi, Jeremy P., et al. "Integrated longitudinal multiomics study identifies immune programs associated with acute COVID-19 severity and mortality." *The Journal of clinical investigation* 134.9 (2024).

# What we're thinking about

- Incorporating the ability of NIPALS to handle missing values into `nipalsMCIA`.
- Extending `nipalsMCIA` to tensor blocks of data (e.g. instead of  $\text{sample} \times (\text{omic})$  one can have  $\text{sample} \times \text{time} \times (\text{omic})$ ).
- Extending proofs for variance explained and out-of-sample embedding to different deflation strategies (e.g. orthogonality of global scores, etc.).

# Give it a test run!


[About](#) [Learn](#) [Packages](#) [Developers](#)
 Search

[Get Started >](#)
[Home](#) > [Bioconductor 3.20](#) > [Software Packages](#) > [nipalsMCIA](#)

## nipalsMCIA

This is the **released** version of nipalsMCIA; for the devel version, see [nipalsMCIA](#).

### Multiple Co-Inertia Analysis via the NIPALS Method

platforms [all](#) rank [1968 / 2289](#) support [0 / 0](#) in BioC [1.5 years](#) build [ok](#) updated [< 1 month](#) dependencies [102](#)

DOI: [10.18129/B9.bioc.nipalsMCIA](https://doi.org/10.18129/B9.bioc.nipalsMCIA)

**Bioconductor version:** Release (3.20)

Computes Multiple Co-Inertia Analysis (MCIA), a dimensionality reduction (JDR) algorithm, for a multi-block dataset using a modification to the Nonlinear Iterative Partial Least Squares method (NIPALS) proposed in (Hanafi et al. 2019). Allows multiple options for row- and table-level preprocessing, and speeds up computation of variance explained. Vignettes detail application to bulk- and single cell- multi-omics studies.

**Author:** Maximilian Mattessich [cre] , Joaquin Reyna [aut] , Edel Aron [aut] , Ferhat Ay [aut] , Steven Kleinstein [aut] , Anna Konstorum [aut]

**URL:** <https://www.bioconductor.org/packages/release/bioc/html/nipalsMCIA.html>

**Manuscript:** Mattessich, Max, et al. "nipalsMCIA: flexible multi-block dimensionality reduction in R via nonlinear iterative partial least squares." *Bioinformatics* 41.1 (2025): btaf015.  
<https://doi.org/10.1093/bioinformatics/btaf015>