MOSS: MULTI-OMIC INTEGRATION WITH SPARSE VALUE DECOMPOSITION

Contents

Statistical background and algorithms

Models of omic integration: *MOSS* performs data integration by creating an extended matrix $\mathbf{Z}$, as a set of normalized omic blocks $\mathbf{Z}_l$ appended column-wise, such as

$$\mathbf{Z} = \left[\frac{1}{||\mathbf{Z}_1||_2^2}\mathbf{Z}_1 \quad \cdots \quad \frac{1}{||\mathbf{Z}_t||_2^2}\mathbf{Z}_t\right],$$

where rows represent subjects or samples, and columns omic features. The subindex t is an arbitrary integer representing the number of omic blocks, and $||.||_2^2$ is the square of the Frobenius norm of a matrix. Then, the following model is adjusted

$$\mathbf{Q} = \mathbf{W\Sigma} + \boldsymbol{\varepsilon} \quad (1),$$

where $\mathbf{Q} = \mathbf{YU}$, $\mathbf{W} = \mathbf{ZV}$, $\mathbf{U}$ and $\mathbf{V}$ are orthonormal loadings matrices with *K* columns, $\mathbf{\Sigma}$ is a matrix of effects, and $\boldsymbol{\varepsilon}$ is a matrix of uncorrelated residuals, not following any distribution. Model (1) becomes a partial least square (PLS) whenever an omic block is used as a multivariate response, say $\mathbf{Y} = \mathbf{Z}_l$. On the other hand, model (1) becomes a principal component analysis (PCA) when $\mathbf{Y}$ is the identity matrix. In all cases, (1) is iteratively solved by least squares to find $\mathbf{Q}$,$\mathbf{W}$, and $\mathbf{\Sigma}$. Additionally, the columns of $\mathbf{Z}$ are assumed to have zero means and unit variances.

Models with covariates: If covariates are included in the model (1), *MOSS* removes their effects by pre-multiplying each omic block by $\mathbf{I}_n - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-}\mathbf{X}^T$, where $\mathbf{X}$ is a matrix with covariates as columns, and $\mathbf{I}_n = \mathbf{diag}(\underbrace{1, \ldots, 1}_{n-times})$. We have used this method in a multi-omic classification of tumors across different cancer types, where the tumors' tissues of origin were used as covariates (González-Reymúndez and Vázquez, 2020).

Estimation of parameters: To estimate $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{\Sigma}$, *MOSS* minimizes the following loss function:

$$L = \|\mathbf{YU} - \mathbf{ZV\Sigma}\|_2^2 + \lambda_U(\alpha_U\|\mathbf{U}\|_1 + (1 - \alpha_U)\|\mathbf{U}\|_2^2) + \lambda_V(\alpha_V\|\mathbf{V}\|_1 + (1 - \alpha_V)\|\mathbf{V}\|_2^2) \quad (2),$$

where $\mathbf{Y}$ and $\mathbf{Z}$ are either the original or covariates-adjusted omics. The first term in the above sum is the ℓ^2 norm of the approximation of $\mathbf{Q}$ by $\mathbf{W\Sigma}$, subject to $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}_K$. The second and third terms are Elastic Net (EN) penalties (Zou *et al.*, 2005) on the elements of $\mathbf{U}$ and $\mathbf{V}$. The EN penalty balances well-established techniques of variable selection (zeroing out the noise and redundant signal between omic features) and shrinkage (to account for the high number of omic features that often exceed the number of samples). The expression $||.||_1$ corresponds to the ℓ^1 norm. Here $\lambda_. = q(\nu_.)$, is considered as a monotonically decreasing function of $\nu_.$ (the number of desired elements different from zero) onto positive real numbers $\lambda_.$, and $\alpha_.$ is any number between zero and one. The value of $\alpha_.$ balances shrinking and variable selection.

When sparsity is not imposed (i.e., $\lambda_. = 0$), solutions for (2) are obtained by taking partial derivatives on $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{\Sigma}$**,** and setting them to zero. Considering $\mathbf{B} = \mathbf{Z}^T\mathbf{Y}$, solutions for $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{\Sigma}$ can be obtained from the partial value decomposition of $\mathbf{B}$ of rank K ($\{\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}}, \widetilde{\mathbf{\Sigma}}\} = SVD(\mathbf{B}, K)$). When $\lambda_. > 0$, solutions are obtained iteratively from the following set of equations:

$$\begin{cases} \mathbf{U}^* = \frac{\widetilde{\mathbf{U}} - \frac{1}{2}\lambda_U\alpha_U\mathbf{sign}(\mathbf{U}^*))}{1+\lambda_U(1-\alpha_U)} \\ \mathbf{V}^* = \frac{\widetilde{\mathbf{V}} - \frac{1}{2}\lambda_V\alpha_V\mathbf{sign}(\mathbf{V}^*))}{1+\lambda_V(1-\alpha_V)} \end{cases} \quad (3).$$

Equations in (3) are solved using the algorithm in (Shen and Huang, 2008) extended to include the EN parameter $\alpha_.$, where $\mathbf{U}^* = \mathbf{U\Sigma}$ and $\mathbf{V}^* = \mathbf{\Sigma V}$. By default, *MOSS* calls the function *svd* from the base R package to compute the SVD (3). However, for large and

very large matrices, *MOSS* uses packages *irlba* (Baglama *et al.*, 2018) and *bigstatsr* (Privé *et al.*, 2018) instead, respectively. To ensure that $\mathbf{U}$ and $\mathbf{V}$ are orthonormal in each iteration, the QR decomposition of $\mathbf{U}^*$ and $\mathbf{V}^*$ ($\mathbf{U}^* = \mathbf{L}^{(U)}\mathbf{R}^{(U)}$ and $\mathbf{V}^* = \mathbf{L}^{(V)}\mathbf{R}^{(V)}$) is used. In each iteration, $\mathbf{U}$ is recovered as $\mathbf{U} = \mathbf{L}^{(U)}\mathbf{D}^{(U)}$, where $\mathbf{D}^{(U)} = \mathbf{diag}(d_1, d_2, \ldots, d_K)$, and $d_k = 1/||\mathbf{L}_k^{(U)}||_2$ (the inverse of the norm of each column of $\mathbf{L}^{(U)}$). The same steps are used to recover $\mathbf{V}$. After a fixed number of iterations, or at convergence, the final values of $\mathbf{U}$ and $\mathbf{V}$ are used to recover $\mathbf{\Sigma}$ ($\mathbf{\Sigma} = \mathbf{U}^T\mathbf{B}\mathbf{V}^T$).

Tuning hyperparameters: The value $\lambda_{\cdot}$ is tuned following (Shen and Huang, 2008), modified to tune λ_U, λ_V, or both. Briefly, $\lambda_{\cdot}$ is chosen as the $\nu_{\cdot}$-th order statistic of $\mathbf{U}$, or $\mathbf{V}$, where ν_U and ν_V are fixed numbers representing the desired number of samples and features loadings different from zero (i.e., the degrees of sparsity), respectively. Then, the proportion of variance explained (PEV) by each value, on a grid of values of ν_U and ν_V, is calculated. The trajectory of PEV across values of ν_U and ν_V is then used to select an "optimal" $\lambda_{\cdot}$ value to solve (2). This selection is made automatically via two alternative methods. The first method uses the first empirical partial derivative of PEV $\left(\frac{\partial PEV}{\partial \nu_{\cdot}}\right)$ to choose the value of $\nu_{\cdot}$ at which the change in PEV is maximum ("liberal" method). The second method chooses the value of $\nu_{\cdot}$ at which the change in PEV stabilizes ("conservative" method). Similarly, *MOSS* displays a classic plot of $\sigma_1, \ldots, \sigma_K$ (Scree plot), to visualize the change in variance explained by each latent dimension. The number of latent dimensions K is not tuned by *MOSS* internally. However, automatic suggestions are provided based on the above heuristics. By default, the process of tuning the degree of sparsity is done in series. However, this can be changed by setting argument *nu.parallel=TRUE*. This option uses

package *future.apply* to allow for simple parallel distribution of tasks on a local machine or computer cluster (Bengtsson, 2020).

Handling large data sets: MOSS calculates a truncated SVD using the *irlba* package (Baglama *et al.*, 2018). However, whenever data sets are too big to be handled in RAM, omic blocks can be passed to *MOSS* as Filed-backed Big Matrices (FBM), and randomized SVD is performed. For this, package *bigstatsr* (Privé *et al.*, 2018) must be installed. Alternatively, omic blocks passed to *MOSS* as 'matrix' or 'array' can be internally turned into FBM. In our experience, this can speed up computations when matrices are small enough to fit in memory but still too large to be handled in a reasonable time. Another way *MOSS* speeds up computations is by calculating $\mathbf{B} = \mathbf{Z}^T\mathbf{Y}$ by using a low rank decomposition of $\mathbf{Z}$, e.g., $\mathbf{Z} = \mathbf{ADC}$, such as $\mathbf{B}$ can be calculated by multiplying two matrices of smaller dimensions, $\mathbf{C}^T\mathbf{D}$ and $\mathbf{A}^T\mathbf{Y}$.

Cluster analysis: *MOSS* can use the columns $\mathbf{Q}$ to detect clusters of samples via Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester *et al.*, 1996). DBSCAN identifies groups of densely packed points without specifying the number of clusters a priori. In *MOSS*, neighborhoods of nearby points can then be tuned by evaluating different cluster partitions over a grid of values of *e*, a hyperparameter controlling the neighborhood. *MOSS* chooses the number of "optimal" clusters that maximizes the Silhouette score (Rousseeuw, 1987) over a grid of possible *e* values, as in (Taskesen *et al.*, 2016).

Visualization of clusters: *MOSS* uses *t* - Stochastic Neighbor Embedding (tSNE) to display the left singular vectors $\mathbf{Q}$ (Taskesen *et al.*, 2016). Essentially, tSNE projects multiple dimensions onto a lower-dimensional display while conserving local

neighborhoods (eventually representing data clusters) (van der Maaten and Hinton, 2008). tSNE is an effective technique to reveal clusters (Linderman and Steinerberger, 2017). The algorithm has two fundamental parameters: *perplexity* (which accounts for the adequate number of local neighbors) and *cost* (related to the difference between the neighborhood's distribution in the higher and lower dimensional spaces). Since low costs are more likely to reveal clusters, *MOSS* tunes the tSNE projection by choosing the map of minimum cost among multiple random starts of the algorithm.

EXAMPLE OF USAGE AND SYNTAXIS

Simulated data: For the following example we use the R package *MOSim* (Tarazona and Martinez, 2021) and accompanying omic data therein to simulate three omic blocks. Simulated data consisted of gene expression as read counts from RNA sequencing (RNA-seq), micro-RNA sequencing (miRNA-seq), and DNAase I activity as Assay for Transposase-Accessible Chromatin (ATAC) sequencing data of (DNase-seq) were simulated. Signal effects were imposed by assigning 5% miRNA-seq and DNase-seq features regulating the expression of 15% of total genes across three clusters of samples.

Analysis: The following code assumes omic blocks are stored in the object *out_sim* and passed to the function *moss*. A detailed description of the argument is presented in Table 1. In this example, we use PLS, where the first omic block corresponding to gene expression is taken as a response, while the remaining blocks, miRNA, and DNase-seq sequence, are taken as predictors.

```
set.seed(345)
out_moss <- moss(data.blocks = out_sim,
                 method = "pls",
                 resp.block = 1,
```

```
        scale.arg = TRUE,
        norm.arg = TRUE,
        K.Y = 3,
        nu.v = seq(1,200,by=2),
        nu.u = seq(1,100,by=2),
        alpha.v = 0.5,
        alpha.u = 0.5,
        K.X = 50,
        use.fbm = TRUE,
        nu.parallel = TRUE,
        tSNE = TRUE,
        cluster = TRUE,
        plot =TRUE)
```

All the blocks are scaled and normalized (unless *scale.arg* and *norm.arg* are set to FALSE). The elements of $\mathbf{U}$ represent loadings of the response features (genes, in this case) and the elements of $\mathbf{V}$ represent loadings of the predictors' features (miRNAs and DNAase peaks). The number of columns for both $\mathbf{U}$ and $\mathbf{V}$ is set to three (*K.Y* = 3). The degrees of sparsity in both cases are tuned in increments of two.

The following steps are followed to increase the speed of computations. First, the omics blocks are coerced to FBM. Second, the matrix $\mathbf{B}$ is calculated using the decomposition of $\mathbf{Z}$ (see *Handling large data sets*, Materials and Methods) with *K.X* = 50 representing the rank of the decomposition. Third, the degrees of sparsity are tuned in with parallel computations. Finally, the *tsne*, *cluster*, and *plot* arguments tell *MOSS* to project subjects' factors onto two dimensions via tSNE, detect clusters, and plot results.

Table 1: Arguments in function *moss*.

Package's capability	Argument	Description	Object type
Omic blocks	*data.blocks*	Omic blocks to be integrated. In each block, rows represent subjects and columns features.	List with elements of class' matrix' or 'FBM'.
Preliminary options	*scale.arg*	Standardize omics.	Logical
	norm.arg	Normalize omics	Logical
	use.fbm	Should we treat omic blocks as FBM.	Logical
Multivariate method	*method*	Multivariate method.	Character
	resp.block	Omic block used as a response.	Integer
SVD and Sparsity constraints	*K.X*	The number of latent dimensions for predictors.	Integer
	K.Y	The number of responses latent dimension.	Integer
	nu.v	Grid of degrees of sparsity for left Eigenvectors.	Numerical
	alpha.v	Elastic Net parameter for right Eigenvectors.	Numerical
	nu.u	Grid of degrees of sparsity for left Eigenvectors.	Numerical
	alpha.u	Elastic Net parameter for left Eigenvectors.	Numerical
	lib.thresh	Liberal or conservative threshold to tune degrees of sparsity	Logical
	exact.dg	Compute exact degrees of sparsity.	Logical
Visualization and cluster analysis	*plot*	Display plots.	Logical
	tsne	Project latent dimensions with tSNE.	Logical
	cluster	Cluster analysis via DBSCAN.	Logical
Parallel analysis	*nu.parallel*	Tuning degrees of sparsity in parallel.	Logical

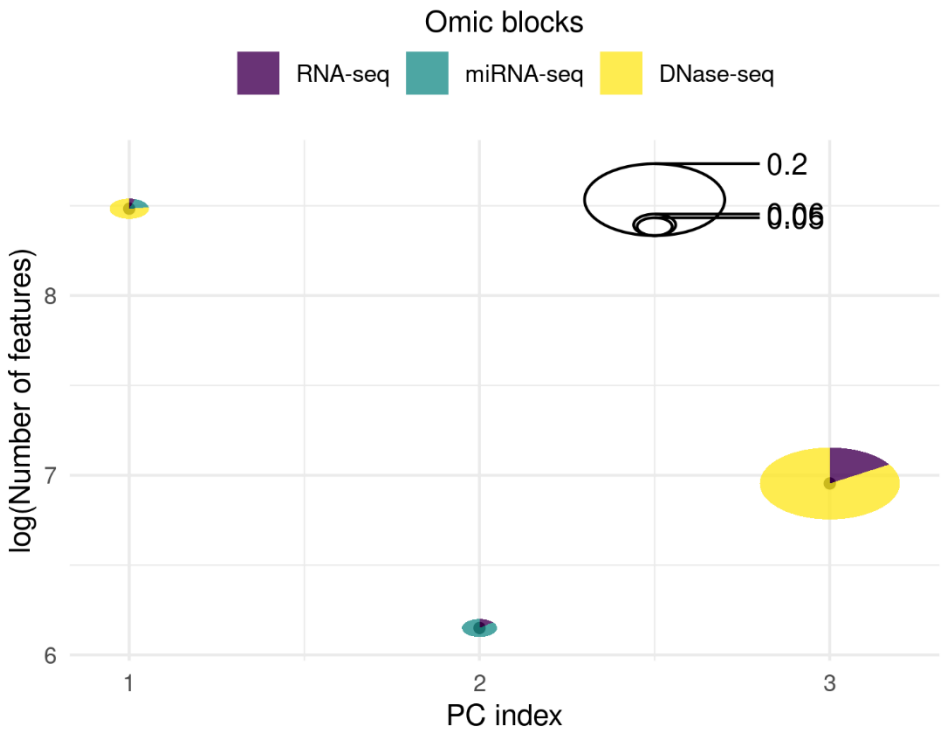

Figure 1: Omic contribution to selected features. The numbers of features selected by latent factors are shown in a log2 scale. The pie charts slides represent the relative contribution of each omic (RNA-seq, miRNA-seq, and DNase-seq) to the features selected by dimension (PC index). The pie charts size represents the quotient between the squared loadings of the selected features by dimension and their standard deviation.

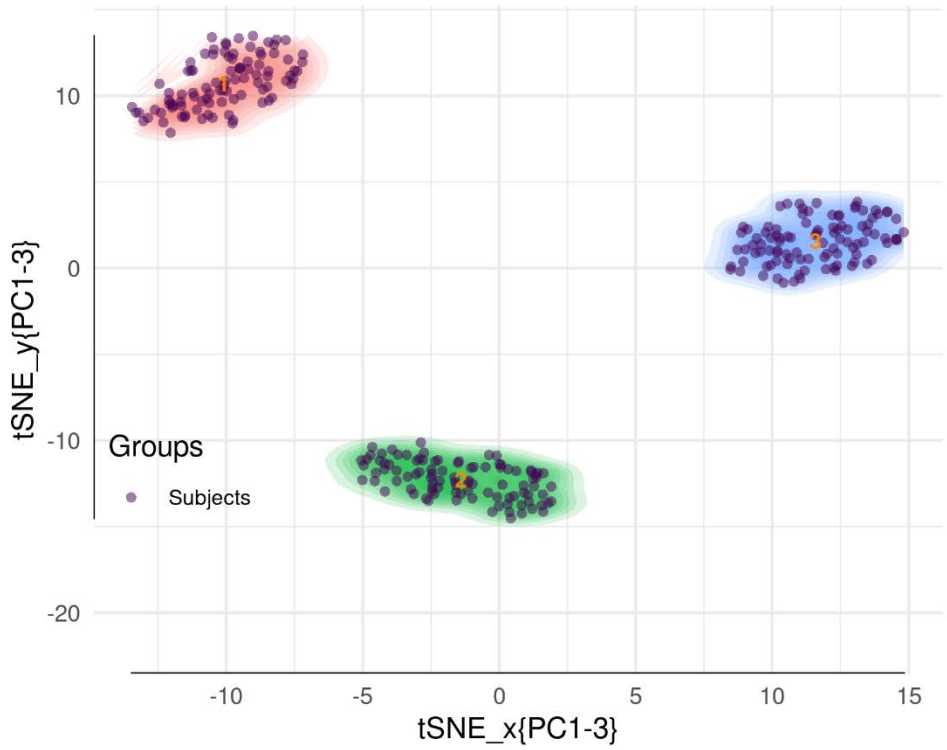

Figure 2: Cluster analysis.

References

- Baglama,J. *et al.* (2018) irlba: Fast Truncated Singular Value Decomposition and Principal Components Analysis for Large Dense and Sparse Matrices. *CRAN R Proj.*
- Bengtsson,H. (2020) A Unifying Framework for Parallel and Distributed Processing in R using Futures. *CoRR*.
- Ester,M. *et al.* (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*., pp. 226–231.
- González-Reymúndez,A. and Vázquez,A.I. (2020) Multi-omic signatures identify pan-cancer classes of tumors beyond tissue of origin. *Sci. Rep.*, **10**, 8341.
- Linderman,G.C. and Steinerberger,S. (2017) Clustering with t-SNE, provably. *arXiv.org*.
- van der Maaten,L. and Hinton,G. (2008) Visualizing Data using t-SNE. *J. Mach. Learn. Res.*, **9**, 2579–2605.
- Privé,F. *et al.* (2018) Efficient analysis of large-scale genome-wide data with two R packages: bigstatsr and bigsnpr. *Bioinformatics*, **34**, 2781–2787.
- Rousseeuw,P.J. (1987) Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, **20**, 53–65.
- Shen,H. and Huang,J.Z. (2008) Sparse principal component analysis via regularized low rank matrix approximation. *J. Multivar. Anal.*, **99**, 1015–1034.
- Tarazona,S. and Martinez,C. (2021) Bioconductor - MOSimMulti-Omics

Simulation (MOSim).

- Taskesen,E. *et al.* (2016) Pan-cancer subtyping in a 2D-map shows substructures that are driven by specific combinations of molecular characteristics. *Sci. Rep.*, **6**, 24949.
- Zou,H. *et al.* (2005) Regularization and variable selection via the Elastic Net. *J. R. Stat. Soc. Ser. B*, **67**, 301–320.