Documentation of Muscle codes

In this document, we provide a full description of each code we provide for Muscle. From now on, a code or function will be denoted with ***bold face and Italic style***, and a directory will be denoted with only **bold face**. A configuration file will be denoted with only *Italic style.*

There are mainly two directories containing codes that implement Muscle. To provide a brief overview, ***Muscle.sh*** in **wrapper** directory runs both the ***Preprocessing.R*** and ***Muscle_wrapper.R*** based on configurations given by *config_file_model.R* and *config_file_preprocess.R*, and each of the ***Preprocessing.R*** and ***Muscle_wrapper.R*** loads required functions in the **functions** directory. Below, we provide the full algorithm of Muscle on this document, which is equivalent to the one existing in Supplementary materials of Muscle.

Algorithm 1 Muscle ALS algorithm

Input: scHi-C tensors $\mathcal{Y}_{chr} \in \mathbb{R}^{l_{chr} \times l_{chr} \times C}$, methylation matrices $\mathbf{Y}^{CG}, \mathbf{Y}^{CH} \in \mathbb{R}^{\sum_{chr} l_{chr} \times C}$, scHi-C loci loading rank K_{chr}, $\forall chr \in [Chr]$, data modality common rank R.

Output: scHi-C loci loadings $\mathbf{A}_{chr,r}, B_{chr,r} \in \mathbb{R}^{l_{chr} \times K_{chr}}$, methylation loci loadings $\mathbf{v}_r^{CG}, \mathbf{v}_r^{CG} \in \mathbb{R}^{\sum_{chr} l_{chr}}$, and data modality common cell loading vector $\mathbf{c}_r \in \mathbb{R}_+^C$, $\forall chr \in [Chr]$ and $\forall r \in [R]$.

Initialize the decomposition objects $\tilde{\mathcal{Y}}_{chr} \leftarrow \mathcal{Y}_{chr}$ $\forall chr \in [Chr]$, $\tilde{\mathbf{Y}}^{CG} \leftarrow \mathbf{Y}^{CG}$, $\tilde{\mathbf{Y}}^{CH} \leftarrow \mathbf{Y}^{CH}$.

for $r = 1$ to R **do**.

  Initialize $\hat{\mathbf{A}}_{chr,r}$, and $\hat{\mathbf{B}}_{chr,r}$ by rank $(K_{chr}, K_{chr}, 1)$ Higher-order SVD algorithm on $\tilde{\mathcal{Y}}_{chr}$ $\forall chr \in [Chr]$. Note here that the core tensor size is absorbed into $\hat{\mathbf{A}}_{chr,r}$.

  Initialize $\hat{\mathbf{v}}_r^k$ by first left singular vector of $\tilde{\mathbf{Y}}^k$ (singular value is absorbed), $\forall k \in \{CG, CH\}$.

  while the convergence criterion is not met **do**

    Update $\hat{\mathbf{c}}_r \leftarrow \frac{\left(\frac{1}{N_h}\sum_{chr} \mathbf{Y}_{chr}^T \mathbf{X}_{chr} + \frac{1}{N_m}(\tilde{\mathbf{Y}}^{CG})^T \mathbf{v}_r^{CG} + \frac{1}{N_m}(\tilde{\mathbf{Y}}^{CH})^T \mathbf{v}_r^{CH}\right)_+}{\left\|\left(\frac{1}{N_h}\sum_{chr} \mathbf{Y}_{chr}^T \mathbf{X}_{chr} + \frac{1}{N_m}(\tilde{\mathbf{Y}}^{CG})^T \mathbf{v}_r^{CG} + \frac{1}{N_m}(\tilde{\mathbf{Y}}^{CH})^T \mathbf{v}_r^{CH}\right)_+\right\|_2}$,

    with $\mathbf{Y}_{chr} = \text{unfold}_3\left(\tilde{\mathcal{Y}}_{chr}\right) \in \mathbb{R}^{l_{chr}^2 \times C}$ and $\mathbf{X}_{chr} = [(\hat{\mathbf{A}}_{chr,r} \odot \hat{\mathbf{B}}_{chr,r})\mathbf{1}_{K_{chr}}] \in \mathbb{R}^{l_{chr}^2}$.

    Update $(\hat{\mathbf{A}}_{chr,r}, \hat{\mathbf{B}}_{chr,r}) \leftarrow \text{Eigen}_{K_{chr}}(\tilde{\mathcal{Y}}_{chr} \times_3 \hat{\mathbf{c}}_r^T)$, $\forall chr \in [Chr]$. Note here that the eigenvalues are absorbed into $\hat{\mathbf{A}}_{chr,r}$.

    Update methylation loci loadings $\mathbf{v}_r^k \leftarrow \tilde{\mathbf{Y}}^k \hat{\mathbf{c}}_r$. $\forall k \in \{CG, CH\}$.

  end while

  Update $\tilde{\mathcal{Y}}_{chr} \leftarrow \tilde{\mathcal{Y}}_{chr} - (\hat{\mathbf{A}}_{chr,r}\hat{\mathbf{B}}_{chr,r}^T) \circ \hat{\mathbf{c}}_r$ and $\tilde{\mathbf{Y}}^k \leftarrow \tilde{\mathbf{Y}}^k - \hat{\mathbf{v}}_r^k \circ \hat{\mathbf{c}}_c$ for $\forall k \in \{CG, CH\}$.

end for

Table 1 Algorithm 1 of Muscle

1. **wrapper** directory

1.1 Muscle.sh

This bash file runs both the ***Preprocessing.R*** and ***Muscle_wrapper.R*** based on configurations given by ***config_file_model.R*** and ***config_file_preprocess.R.*** Specifically, a user can run the following code "bash Muscle.sh" after filling in all the components of ***config_file_model.R*** and ***config_file_preprocess***.

1.2 Preprocessing.R

This code conducts preprocessing of the Muscle by loading 'hic_df.qs' file and desired methylation matrices, e.g., 'data_methy_CG.qs' or 'data_methy_CH.qs' after loading the model configurations from *config_file_preprocess.R*. In brief, it conducts initial imputation of the scHi-C data and conducts debiasing depending on a user's choice. After those steps, it generates scHi-C tensors using ***tensor_generator.R***. Details about the preprocessing is provided in section S3 of the supplementary material of Muscle.

1.3 Muscle_wrapper.R

This code conducts the entire steps of the Algorithm 1 of Muscle in supplementary materials, by taking in preprocessed data from ***Preprocessing.R*** and loading the configuration file *config_file_model.R*. This code is the main function that loads all the required sub-functions within **functions** directory, which will be stated in the following section.

1.4 config_file_preprocess.R

This is a configuration file that a user needs to fill in depending on the user's need for preprocessing. This will be loaded byDetails about how to choose each component is in our GitHub Wiki page.

1.5 config_file_model.R

This is a configuration file that a user needs to fill in depending on the user's need for Muscle model fit. Details about how to choose each component is in our GitHub Wiki page.

2. **functions** directory

2.1 Muscle_functions.R

- ***rankone_Muscle:*** This code is loaded by ***Muscle_wrapper.R*** within the **wrapper** directory and conducts each of the for loops described in line 2-11 of Algorithm 1. Lines 3,4 are conducted by loading ***Initializer.R*** within **functions** directory and lines 7,8 are conducted by loading **ModeAB_learner.R** within **functions** directory. The line 6, corresponding to common cell loading learning is conducted soley by this code.
- ***BTD_combi2*****:** This function is calculates $\left(\hat{A}_{chr,r}\hat{B}_{chr,r}^{T}\right) \circ \hat{c}_r$ in line 10 of Algorithm 1
- ***khatri_rao_matrix2*****:** This function calculates khatri rao product
- ***Fnorm*****:** This function calculates Frobenius norm of a tensor
- ***naomit_matrix*****:** This function deletes rows of a matrix with all zeros.
- ***scalemat*****:** This function scales a matrix.
- ***sparse2dense*****:** This function is a matrix that converts a long format matrix into a square matrix. This is used when generating scHi-C tensor by ***tensor_generator.R***. The code is originally from a R package "DIADEM".
- ***balance*****:** When converting long format matrix into square matrix by **sparse2dense**, this function is used to make the resulting matrix be square. The code is originally from a R package "DIADEM".
- ***hosvd_new*****:** This function conducts the HOSVD utilized in ***Initializer.R*** and it's a modified version of ***hosvd*** function in "rTensor" R package. This function is different in that it conducts truncated SVD instead of full SVD.

2.2 Initializer.R

This code is loaded by function ***rankone_Muscle*** in **Muscle_functions.R** and conducts spectral initialization (HOSVD for tensors and SVD for matrices) described in lines 3 and 4 of the Algorithm 1 provided in the supplementary material.

2.3 ModeAB_learner.R

This code is loaded by function ***rankone_Muscle*** in **Muscle_functions.R** and learns modality specific parameters $\hat{A}_{chr,r}, \hat{B}_{chr,r}, \hat{v}_r^k$ described in lines 7 and 8 of Algorithm 1.

2.4 multiply.cpp

This is a C++ code that conducts faster multiplication of matrices. This function is loaded for every matrix multiplication used in Muscle.

2.5 tensor_generator.R

This is a R code loaded by ***Preprocessing.R*** within **wrapper** directory and generates scHi-C tensor for each chromosome.