

We provide all the algorithms described in this paper in this section. Note that we have referred to genes as “columns” and experiments or arrays as “rows”, interchangeably. The expression matrix contains n rows and p columns.

KNN imputation

The following KNN imputation algorithm has been verified to produce the same estimates as those described by Troyanskaya et al. (2001). Implementation of KNN imputation in C++ is provided by Troyanskaya et al. at <http://smi-web.stanford.edu/projects/helix/pubs/impute/>. Our implementation in Matlab as well as documentations are freely available for download <http://stat.tamu.edu/~dnguyen/supplemental.html>.

KNN Imputation Algorithm

1. Specify the number of neighbors, K .
2. Determine the m columns with MVs, \mathcal{L} , and determine the MV’s row labels within each column: \mathcal{L}_j , for $j \in \mathcal{L}$.
3. FOR $j \in \mathcal{L}$ DO
 - Assign target gene: \mathbf{x}_j .
 - FOR $v \in \mathcal{L}_j$ DO
 - a. Find candidate gene set for MV row label v :
 $\mathcal{C}_v = \{\text{column } k : k \text{ has an available value in row } v; k = 1, \dots, p, k \neq j\}$.
 - b. Compute distance of each candidate gene to the target gene:
 $d_{jl} \equiv d(\mathbf{x}_j, \mathbf{x}_l), l \in \mathcal{C}_v$.
 - c. Find the candidate genes corresponding to the K “smallest” distances: \mathcal{C}_v^* .
 - d. Estimate the MV by a weighted average of the expression values of the K nearest candidate genes in row v , with weights w_{jl} depending on $d_{jl}, l \in \mathcal{C}_v^*$:
 $\hat{y}_{vj} = \sum_{l \in \mathcal{C}_v^*} w_{jl} y_{vl}$.

END
END

Constructing PLS gene components

The following PLS algorithm is used to obtain PLS weights, components, and related quantities for PLS imputation. The input data sets are standardized to mean 0 and variance 1: $x_{ij} \leftarrow (x_{ij} - m_j^x)/s_j^x$ and $y_{ij} \leftarrow (y_{ij} - m_j^y)/s_j^y$. The test data is standardized using training data: $x_{ij}^* \leftarrow (x_{ij}^* - m_j^x)/s_j^x$. PLS is not invariant to scaling of the variables. It is recommended that variables to be standardized in the absence of prior information about the importance of the variables. For gene expression data the target and candidate genes are standardized to have mean 0 and variance 1 across the available samples. That is, the target gene \mathbf{x}_j^A and the candidate genes (columns of \mathbf{X}_j^A) are standardized.

PLS Algorithm

1. Input training data pair (\mathbf{X}, \mathbf{Y}) and test data \mathbf{X}^* .
2. Input the number of PLS components, K_P .

3. Set convergence criterion ϵ , say $\epsilon = 1\text{E-}12$ and set $\mathbf{X}_1 = \mathbf{X}$ and $\mathbf{Y}_1 = \mathbf{Y}$.
4. FOR $k = 1$ to K_P DO
 - Set \mathbf{u} to be the first column of \mathbf{Y}_k and initialize δ .
 - a. WHILE $\delta > \epsilon$ DO
 - $\mathbf{w} = \mathbf{X}'_k \mathbf{u} / \mathbf{u}' \mathbf{u}$ and scale \mathbf{w} to unit length.
 - $\mathbf{t} = \mathbf{X}_k \mathbf{w}$. PLS components.
 - $\mathbf{c} = \mathbf{Y}'_k \mathbf{t} / \mathbf{t}' \mathbf{t}$ and scale \mathbf{c} to unit length.
 - $\mathbf{u} = \mathbf{Y}_k \mathbf{c}$.
 - $\delta = (\mathbf{w} - \mathbf{w}_{\text{prev}})'(\mathbf{w} - \mathbf{w}_{\text{prev}})$, \mathbf{w}_{prev} is the previous value of \mathbf{w} .
 - END
 - b. Compute and save relevant quantities:
 - $\mathbf{c}_k = \mathbf{c}$.
 - $\mathbf{p}_k = \mathbf{X}'_k \mathbf{t} / (\mathbf{t}' \mathbf{t})$ and scale \mathbf{p}_k to unit length.
 - $\mathbf{t}_k = \mathbf{t} c_p$, $c_p = (\mathbf{p}'_k \mathbf{p}_k)^{0.5}$.
 - $\mathbf{w}_k = \mathbf{w} c_p$.
 - $b_k = \mathbf{u}' \mathbf{t} / (\mathbf{t}' \mathbf{t})$.
 - Residual Matrices: $\mathbf{X}_{k+1} = \mathbf{X}_k - \mathbf{t}_k \mathbf{p}'_k$, $\mathbf{Y}_{k+1} = \mathbf{Y}_k - b_k \mathbf{t}_k \mathbf{c}'_k$.
 - END
5. Compute test components based on training information from 5.
 - Set $\mathbf{X}_1^* = \mathbf{X}^*$ and compute $\mathbf{t}_1^* = \mathbf{X}_1^* \mathbf{w}_1$. Subsequent test components are computed as $\mathbf{t}_k^* = \mathbf{X}_k^* \mathbf{w}_k$ where $\mathbf{X}_k^* = \mathbf{X}_{k-1}^* - \mathbf{t}_{k-1}^* \mathbf{p}_{k-1}$, for $k = 1, \dots, K_P - 1$.

PLS imputation

Estimates from KNN are used as initial estimates for PLS imputation on line (e) of the *PLS Imputation Algorithm* below. The *PLS Algorithm*, given above, is used to obtain PLS weights in and it is called on line (f).

PLS Imputation Algorithm

1. Determine the m columns with MVs, \mathcal{L} .
2. Specify the number of PLS components, K_P , for predicting MVs.
3. FOR $j \in \mathcal{L}$ DO
 - a. Assign target gene: \mathbf{x}_j .
 - b. Determine the MV and available value row labels, M and A , in \mathbf{x}_j respectively.
 - c. Select as candidate genes all columns with available values in the rows of M .
 - d. Partition candidate gene expression matrix, \mathbf{X}_{-j} , by the row labels A and M in $\mathbf{x}_j: (\mathbf{x}_j^A, \mathbf{x}_j^M)$ and $(\mathbf{X}_{-j}^A, \mathbf{X}_{-j}^M)$. Note that, by construction, \mathbf{X}_{-j}^* is complete but \mathbf{X}_{-j}^A may not be.
 - e. Fill in missing entries of \mathbf{X}_{-j}^A with estimated values from *KNN algorithm*.
 - f. Obtain PLS training components, $\mathbf{T}^A(j)$, and test components, $\mathbf{T}^*(j)$, i.e. call *PLS Algorithm* with inputs: $(\mathbf{X}, \mathbf{Y}) \leftarrow (\mathbf{X}_{-j}^A, \mathbf{x}_j^A)$ and $\mathbf{X}^* \leftarrow \mathbf{X}_{-j}^M$.
 - g. Estimate MVs by fitting a regression model of \mathbf{x}_j^A on training components $\mathbf{T}^A(j)$ and predict MVs \mathbf{x}_j^M using fitted model with test components $\mathbf{T}^*(j)$.
- END

OLS imputation

Lines 1-3(a)-3(c) of the *OLS Imputation Algorithm* below are identical to the *KNN Imputation Algorithm*, except for each MV K estimates are obtained via linear regression prediction (loop d). The final estimate of the MV is an average of the K estimates.

OLS Imputation Algorithm

1. Specify the number of neighbors, K .
2. Determine the m columns with MVs, \mathcal{L} , and determine the MV's row labels within each column: \mathcal{L}_j , for $j \in \mathcal{L}$.
3. FOR $j \in \mathcal{L}$ DO
 Assign target gene: \mathbf{x}_j .
 FOR $v \in \mathcal{L}_j$ DO
 a. Find candidate gene set for MV row label v :
 $\mathcal{C}_v = \{\text{column } h : h \text{ have available values in row } v; h = 1, \dots, p, h \neq j\}$.
 b. Compute distance of each candidate gene to the target gene:
 $d_{jl} \equiv d(\mathbf{x}_j, \mathbf{x}_l), l \in \mathcal{C}_v$.
 c. Find the candidate genes corresponding to the K "smallest" distances: \mathcal{C}_v^* .
 d. FOR $k \in \mathcal{C}_v^*$ DO
 Regress \mathbf{x}_j on \mathbf{x}_k using available data between columns j and k to get $\hat{y}_{vj}^{(k)}$.
 END
 e. Estimate the MV, y_{vj} , by a weighted average of the $\hat{y}_{vj}^{(k)}, k \in \mathcal{C}_v^*$.
 END
END
END