**Supplementary materials for the manuscript**

**ARBic: an all-round biclustering algorithm for analyzing gene expression data**

Xiangyu Liu1,2, Zhengchang Su4, Guojun Li1,2,3,\*

1Research Center for Mathematics and Interdisciplinary Sciences, 2School of Mathematics, Shandong University, Jinan 250100, China; 3School of Mathematical Science, Liaocheng University, Liaocheng 252000, China; 4Department of Bioinformatics and Genomics, The University of North Carolina at Charlotte, Charlotte, NC 28223, USA.

**Contents:**

[1. Definitions of bicluster types 2](#_Toc77241676)

[2. Counterexample of UniBic 2](#_Toc77241677)

[3. Find the longest directed path in *Gij* 3](#_Toc77241678)

[4. Find an integer k to partition background matrix into subsets of rows and columns 4](#_Toc77241679)

[5. An example of row-based stratey 5](#_Toc77241680)

[6. ARBic outperformed the existing tools for finding biclusters with different patterns 6](#_Toc77241681)

[7. Details of real datasets used in the maintext 8](#_Toc77241682)

[8. Data preprocessing step 8](#_Toc77241683)

1. **Definitions of bicluster types**

A bicluster is said to be **constant** if all the elements in the bicluster are same, **shift** if each rowcan be obtained from a base row by adding a number to the base row, **scale** if each row can be obtained from a base row by multiplying the base row by a number, and **scale-shift** if each row can be obtained from a base row by scaling and shifting simultaneously. Obviously, the first three types can be viewed as special cases of scale-shift pattern. A bicluster is said to be **trend-preserved** if there is a column permutation such that each permutated row is monotonic, i.e. increased or decreased (Figure S1 E). Obviously, trend-preserved biclusters are most general and meaningful local structures in gene expression data including scale-shift patterns as special cases. If we by *πj* denote the typical value under condition *j*, by $α\_{i}$ and $β\_{i} $represent adjustments used in patterns for each gene *i*, then the elements bij of a shift-scale bicluster can be written as *bij=αi×πj+βi* (Figure S1 D). Then constant (Figure S1 A) , shift (Figure S1 B) and scale (Figure S1 C) biclusters are all special cases of the scale-shift pattern with specific $α\_{i}$ or/and $β\_{i}$.

|  |  |  |  |
| --- | --- | --- | --- |
|  | *c*1 | *c*2 | *c*3 |
| *g*1 | 1 | 1 | 1 |
| *g*2 | 1 | 1 | 1 |
| *g*3 | 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | *c*1 | *c*2 | *c*3 |
| *g*1 | 1 | 2 | 3 |
| *g*2 | 2 | 3 | 4 |
| *g*3 | 3 | 4 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | *c*1 | *c*2 | *c*3 |
| *g*1 | 1 | 2 | 3 |
| *g*2 | 2 | 4 | 6 |
| *g*3 | 3 | 6 | 9 |

 **A** Constant **B** Shift **C** Scale

|  |  |  |  |
| --- | --- | --- | --- |
|  | *c*1 | *c*2 | *c*3 |
| *g*1 | 2 | 3 | 4 |
| *g*2 | 3 | 5 | 7 |
| *g*3 | 4 | 7 | 10 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | *c*1 | *c*2 | *c*3 |
| *g*1 | 1 | 7 | 5 |
| *g*2 | 2 | 6 | 4 |
| *g*3 | -1 | -5 | -2 |

  **D** Shift-Scale **E** Trend-Preserved

**Figure S1** Five types of bicluster types. a)Constant bicluster with *πj = 1, j =1, 2, 3; αi = 1, βi = 0,* *i = 1, 2, 3*. b) Shift bicluster with *πj = j* for *j = 1, 2, 3; αi = 1, βi = i - 1* for *i=1, 2, 3.* c) Scale bicluster with *πj =* *j* for *j* = *1, 2, 3; αi = i, βi = 0* for *i = 1, 2, 3.* d) Shift-scale bicluster with *πj = j* for *j* = *1,2,3* and *αi = i, βi = 1* for *i = 1, 2, 3*. **e.** Trend-preserved bicluster with *bi2 ≥ bi3 ≥ bi1* for *i = 1, 2*; *b32 ≤ b33 ≤ b31*.

1. **Counterexample of UniBic**

The biclustering algorithm UniBic we previously developed attempted to discover biclusters of trend-preserved patterns. It applied the Longest Common Subsequence (LCS) finding algorithm to each pair of rows of an index matrix to locate seeds. However, the seed location procedure was not globally optimized. Figure S2 shows a counterexample where the whole bicluster itself is trend-preserved, but then UniBic can only output {{*1, 2*}, {*1, 3, 4, 5*}} or {{*1, 2*}, {*1, 2, 4, 5*}}.

****

**Figure S2** Counterexample of UniBic.The whole bicluster itself is trend-preserved, but UniBic outputs {{*1, 2*}, {*1, 3, 4, 5*}} or {{*1, 2*}, {*1, 2, 4, 5*}}.

1. **Find the longest directed path in *Gij***

For a direct graph *Gij*defined by two vectors, we find its longest paths by traversing an acyclic graph $G\_{ij}^{\*}$ obtained by contracting each clique into a node. Firstly, we assign to each node in *Gij*a weight of number 1, and contract each clique in *Gij* into a node of $G\_{ij}^{\*}$ with weight being the sum of node weights of the clique. Then we find the path with the maximum weight sum of nodes in directed acyclic graph$G\_{ij}^{\*}$. At last by blooming the cliques, we get the node set of longest paths in *Gij*. We find the path with the maximum weight sum of nodes in directed acyclic graph$G\_{ij}^{\*}$ by using dynamic programming:

1. Initialize the node weight *w*(*v*) in$G\_{ij}^{\*}$. The weight of a contracted node is the number of nodes in the clique while the others are all the number 1.
2. Let *V* be the set of all nodes, *S* the set of vertices with indegree 0, and *Ms[vi]* the maximum weight among the paths started at nodes in *S* and ended at node *v*i ( *v*i∈*V*\*S*)*.* Let M be a set of *Ms[vi]*, for all *v*i∈*V*\*S*. Let *Mp[vi]* represent the parent node of *v*i defined by the path ended in *v*i of weight *Ms[vi]*.
3. For each *vj∈V\S* in Topological Sorting order (a topological sorting order of a directed graph is a linear ordering of its vertices such that for every directed edge *v*i*v*j from vertex *v*i to vertex *v*j*, v*icomes before *v*j in the ordering) do:

*Ms[vj]=max{Ms[vi]+w(vj):* *vi* is a parent node of *v*j}, where *w*(*v*j) is the weight of *v*j. Set *Mp[vj]=vi* if *v*i = arg *Ms[vj]*.

1. Find the maximum value *Ms[vt]* in M*.* The desired path in$ G\_{ij}^{\*} $can be traced back through *Mp[vt]*. Then the longest path in *Gij* can be obtained through blooming the cliques in the desired path.
2. **Find an integer k to partition background matrix into subsets of rows and columns**

A bicluster is thought to be a significant bicluster, if it can’t be simply random generated by background matrix . For a background matrix $A\_{n×m}$, the probability that A has an order-preserved submatrix of size *t×s* is estimated by the technique established in [1]. For a randomly selected row, the probability that the entries in its s columns support rightly or reversely a given column index permutation is *2/s*!. As each row is independent with each other, and there are *m···(m-s+1)* ways to choose a complete model of size s, the probability *P* of having *t* rows supporting the column index permutation is estimated as:

$P=\left\{\begin{array}{c}U,if U<1\\1, if U\geq 1\end{array}\right.$ , $U=m\cdots (m-s+1)C\_{n}^{t}(\frac{2}{s!})^{t}(1-\frac{2}{s!})^{n-t}$

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size\s | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| n=1000, m=50 | 21 | 12 | 8 | 6 | 5 | 5 | 5 | 4 | 4 | 3 | 3 |
| n=5000, m=50 | 47 | 20 | 12 | 8 | 6 | 5 | 5 | 4 | 4 | 4 | 3 |
| n=10000, m=50 | 71 | 27 | 14 | 9 | 7 | 6 | 5 | 4 | 4 | 4 | 3 |
| n=15000, m=50 | 93 | 32 | 16 | 10 | 7 | 6 | 5 | 4 | 4 | 4 | 4 |
| n=1000, m=100 | 23 | 13 | 9 | 7 | 6 | 5 | 4 | 4 | 4 | 4 | 4 |
| n=5000, m=100 | 50 | 22 | 13 | 9 | 7 | 6 | 5 | 5 | 4 | 4 | 4 |
| n=10000, m=100 | 76 | 29 | 16 | 10 | 8 | 7 | 6 | 5 | 5 | 4 | 4 |
| n=15000, m=100 | 98 | 35 | 18 | 11 | 8 | 7 | 6 | 5 | 5 | 4 | 4 |

**Figure S3** The value of row number t for different column number s when U < 0.05.

From the Figure S3 we can see that in real data, when the input matrices are of about 10,000 rows, *k=t= 4* would be sufficient for us to anchor a seed for row-based strategy.

Because the column-based strategy aims to identify very narrow biclusters, we set a small *k* (*k=t=3*) for this strategy to anchor seeds. We don’t select *k=t=2*, because it’s a trivail condition.

1. **An example of row-based stratey**

The input matrix is shown in Figure S4. We choose a pair of rows (*i, j*) as the initial seed. Following the rules in main text, we construct graph *Gij* based on the rows i and j. Then we contract the clique consisting of nodes *e* and *d* into a new node *g,* forming the graph *G\* ij* (see FigureS5). We found the longest path(*g→b→c→a*) in *G\* ij*by applying the dynamic programming algorithm on the graph *G\* ij*, and a longest path (*e→d→b→c→a* or *d→e→b→c→a*) by blooming the clique, and thus a seed ({*i,j*}, {*a, b, c, d, e*}) in *Gij*. We then add the row *k* to the current seed to get the next seed (bicluster) of one more row by constructing a graph *Gijk* with nodes *a*, *b*, *c*, *d*, *e* based on the rows *i* and *k* using the same way as used in constructing *Gij* (see FigureS6). The longest path *e→d→b→c* in *Gijk* provides the trend-preserved bicluster {{*i,j,k*},{*b,c,d,e*}}.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *a* | *b* | *c* | *d* | *e* | *f* |
| *I* | 1 | 2 | 2 | 3 | 3 | 0 |
| *J* | 2 | 6 | 5 | 8 | 8 | 7 |
| *K* | 10 | 7 | 4 | 8 | 11 | 1 |

**Figure S4** Input matrix.



 *Gij*  *Gij\**

**Figure S5** The construction of graphs *Gij* and $G\_{ij}^{\*}$.Following the rules mentioned in main text, the graph *Gij* is constructed under all columns using rows *i* and *j,* and the acyclic graph$ G\_{ij}^{\*}$ is obtained by contracting the clique which consists of two nodes *d* and *e* into a new node *g*.



*Gik*

**Figure S6** Theconstruction of graph *Gijk* usingrows *i* and *k*.Following the rules mentioned in main text, the graph *Gijk* is constructed on the nodes/columns (*a*, *b*, *c*, *d*, *e*) based on the rows *i* and *k* in the same way as used in constructing *Gij*.

1. **ARBic outperformed the existing tools for finding biclusters with different patterns**



Figure S7 Comparisons of the eight tools on the six datasets containing various bicluster patterns. a) Results on the dataset implanted with trend-preserving biclusters. b) Results on the dataset implanted with column constant biclusters. c) Results on the dataset implanted with row constant biclusters. d) Results on the dataset implanted with scale biclusters. e) Results on the dataset implanted with shift biclusters. f) Results on the dataset implanted with shift-scale biclusters.

1. **Details of real datasets used in the maintext**

Table S1. The summary of datasets used in the study

|  |  |  |  |
| --- | --- | --- | --- |
| **Datasets** | **#genes** | **#conditions** | **Download links** |
| **E. coli Colombos** | **2093** | **2470** | **https://doi.org/10.5281/zenodo.1157938** |
| **E. coli DREAM5** | **2442** | **805** | **https://doi.org/10.5281/zenodo.1157938** |
| **Yeast GPL2529** | **3178** | **3025** | **https://doi.org/10.5281/zenodo.1157938** |
| **Yeast DREAM5** | **3292** | **536** | **https://doi.org/10.5281/zenodo.1157938** |
| **Human SEEK GPL5175** | **4436** |  **2308** | **https://doi.org/10.5281/zenodo.1157938** |
| **Pathwasys** |  |  | **https://doi.org/10.5281/zenodo.1157938** |

1. **Data preprocessing step**

In the regulation of gene expression, the expression level of genes in an up-regulated state will be higher than that of genes that are normally expressed. The expression level of genes in a down-regulated state will be lower than that of normally expressed genes. Generally speaking, when analyzing gene expression data, differentially expressed genes have more biological significance. In order to bicluster the differential expression values, we need to preprocess the input data.

 For the expression value of a gene under all conditions, we take the median of gene expression under all conditions as the normal expression value of the gene. Then the expression value of the gene up-regulated (down-regulated) is the expression value far away from the normal expression level (median). Let $q$ be the ratio of what we think of as up-regulation and down-regulation expression values. The selection process of upper and lower data is as follows:

1. For any gene, the expression levels of the gene are in non-strict ascending order according to the expression value (the expression level) as$a\_{i,1}$*…*$ a\_{i,s}…a\_{i,l}… a\_{i,n-s}$*,…*$a\_{i,n}$,where $l=\frac{m}{2}, s=nq$.

2. Let $d=min⁡(a\_{i,l}-a\_{i,s}$*,*$ a\_{i,n-s}-a\_{i,l})$, the gene expression of our behavior is divided into three expression intervals $\left[a\_{i,1},a\_{i,l}-d\right]$,$ \left(a\_{i,l},-d,a\_{i,l}+d\right)$,$ \left[a\_{i,l}+d,a\_{i,m}\right]$, respectively represent the expression interval of the gene, the normal expression interval and the expression interval.

3. Set the expression values in the interval $\left(a\_{i,l},-d,a\_{i,l}+d\right)$ as 0, which means we ignore the normal expression of the gene.

The existence of experimental errors causes some errors in gene expression values. In order to reduce the influence of errors on bi-clustering, we adopt a grouping strategy to preprocess the expression data.

We equally partition all the up-regulated expression values of a gene in decreasingly ordered into r intervals, and we set the expression values belonging to the ith interval to be the integer i. We equally partition all the down-regulated expression values of a gene in increasingly ordered into r intervals, and we set the expression values belonging to the ith interval to be the integer -i.

**References:**

1. Ben-Dor, A., et al., *Discovering local structure in gene expression data: the order-preserving submatrix problem.* Journal of computational biology, 2003. **10**(3-4): p. 373-384.