# ARBic: An All-Round Biclustering Algorithm for Analyzing Gene Expression Data

**Xiangyu Liu**
Shandong University
**Zhengchang Su**
The University of North Carolina at Charlotte
**Guojun Li** ( gjli@sdu.edu.cn )
Shandong University

**Research Article**

**Keywords:** gene expression data, bicluster, trend-preserving bicluster, algorithm

**Posted Date:** October 14th, 2021

**DOI:** https://doi.org/10.21203/rs.3.rs-936551/v1

# Abstract

**Background:** Identifying significant biclusters of genes with specific expression patterns is an effective approach to reveal functionally correlated genes in gene expression data. However, existing algorithms are limited to finding either broad or narrow biclusters but both due to failure of balancing between effectiveness and efficiency.

**Methods:** We developed a new algorithm ARBic which can accurately identify any meaningful biclusters of shape no matter broad or narrow in a large scale gene expression data matrix, even when the values in the biclusters to be identified have the same distribution as that the background data has. ARBic is developed by integrating column-based and row-based strategies into biclustering procedure. The column-based strategy borrowed from ReBic, a recently published biclustering tool, prefers to narrow bicluters. The row-based strategy newly designed in this article by repeatedly finding a longest path in a specific directed graph prefers to broader ones.

**Result and Conclusion:** When tested and compared to other seven salient biclustering algorithms on simulated datasets, ARBic achieved recovery, relevance and f1-scores 29% higher than the second best algorithm. Furthermore, ARBic substantially outperforms all of them on real datasets and robusts to noises, shapes of biclusters and types of datasets.

Code: https://github.com/holyzews/ARBic

Data: https://doi.org/10.5281/zenodo.5121018


# 1. Introduction

With the development of high-throughput sequencing technology, a large amount of gene expression data have been produced using RNA-seq technologies [1]. The recent single-cell RNA-seq (scRNA-seq) can even gauge all transcripts in tens of thousands of cells at once [2]. These massive expression data provide an opportunity to find co-expressed genes under specific conditions, which form bases for various functional analyses, including associating genes of unknown functions with biological processes, prioritizing candidate disease genes, and discerning transcriptional regulatory networks [3]. Traditional gene clustering algorithms attempt to find groups of genes that are co-expressed across all conditions or cell types. However, if genes are co-expressed only under certain conditions or in certain cell types, biclusters that are formed by genes under certain conditions or in cell types should be identified as they are more appropriate to represent such co-expressed genes.

Mathematically, Biclustering is defined as to find a submatrix in a real number matrix where the rows of the submatrix are correlated under the columns of the submatrix. Identifying gene expression patterns under different conditions (single cells or samples such as cancerous tissues at different developmental stages), one may find genes involved in specific biochemical pathways responsible for cellular phenotypes, or dysregulated during the development of cancer. The biclustering problem can be traced

back to Morgan et al. [4] and Hartigan [5] who attempted to partition a numerical matrix into submatrices whose values were as similar as possible. Since Cheng and Church [6] firstly introduced a biclustering algorithm for gene expression data analyses, lots of biclustering algorithms [7-14] have been developed. However, there are still many challenges in designing biclustering algorithms [15]. However, biclustering, which clusters genes and conditions or cell types simultaneously, is a computationaly challenging problem.

The first is to define biclusters such that they better represent genes that are functionally related. Chen and Church [6] defined a bicluster as a group of genes with an almost constant pattern expression across certain conditions, which was reasonable when only up- or down-regulated events are concerned. However, functionally related genes may not necessarily all up- or down- regulated, some genes may show opposite direction of regulation. To identify functionally-related genes with more complex expression patterns, various definition of biclusters(Supplementary Figure S1) have been proposed, such as row constant, column constant, shift, scale, shift + scale, and trend-preserving patterns [15, 16]. The biclusters of trend-preserving patterns are more inclusive and more biologically meaningful. A bicluster is said to be trend-preserved if and only if any pair of rows/genes within the bicluster are trend-preserved. Two rows are said to be trend-preserved if and only if they are either order-preserved or order-reversed. Two rows (vectors) x and y are said to be order-preserved if and only if any two corresponding components have the same rank (with respect to the numerical value) in their respective rows, and order-reversed if and only if x and -y are order-preserved [16]. Although many biclustering algorithms have been developed to find order-preserving biclusters, their accuracy and computational efficiency remain low, and none of them is able to handle both broad and narrow biclusters [17].

The second is to define an objective function whose optimization provides biclusters of functionally related genes. Although many objective functions have been proposed, most of them cannot effectively select candidate co-expression genes that are functionally correlated under certain conditions. For instance, CC[6] and UniBic[16] use an objective function that is related to bicluster sizes; OPSM[18] and EBIC[15] defined an objective function based on the probability of biclusters to be randomly generated; QUBIC2[19] employed the KL score based on the difference between the distributions of biclusters and background. Moreover, the existing algorithms lack the ability to eliminate co-expressed genes that are related to false positives. Therefore, it is imperative to develop a new biclustering algorithm that can find a set of candidate co-expressed genes under certain conditions with the false discovery rate of co-expressed genes as small as possible.

The third is to design an efficient algorithm to optimize the objective function as biclustering has been proved to be NP-hard even for the simplest case with constant values[20]. Although probabilistic algorithms such as Plaid[21], FABIA[22], ISA[23], CC[6], BBC[9], are relatively computationally efficient, their accuracy is low. Moreover, they are not suitable to identify narrow biclusters with many rows but only a few columns. Heuristic algorithms such as xMOTIFs[24], CPB[11], UniBic[16], QUBIC[25] and QUBIC2[19], can achieve more accurate results, but are still limited because they start with a seed that cannot be guaranteed to be global optimum, and they are not able to cope with narrower biclusters either.

Very recently, two algorithms EBIC[15] and RecBic[26]) were developed to identify narrower biclusters. However, they are not suitable for finding broader ones in dataset matrices with a large number of columns.

To overcome these limitations, we designed a new algorithm ARBic capable of extracting both narrower and broader biclusters encoded in huge dataset matrices based on a new objective function and a graph-theoretic optimization procedure, and compensated for the shortage inherited from UniBic (Supplementary Figure S2). The ARBic was developed by integrating our earlier algorithm RecBic targeted to narrower biclusters[26] as one of two components of ARBic. It can automatically decide which of the two components to be used based on the data structure for better performance. When tested on simulated data, ARBic achieved recovery, relevance and f1-scores at least 29% higher on average than the best existing algorithm. In addition, ARBic outperforms existing algorithms on real datasets in accuracy and robusts to noises, bicluster shapes, and dataset types.

# 2. Methods

ARBic consists of two components for identifying narrow biclusters and broad biclusters, respectively. We used our earlier algorithm RecBic[26] to find narrow biclusters, which is highly effective and efficient for recognizing trend-preserving narrow biclusters. To identify broad ones, we took a different strategy. Intuitively, if a pair of background rows in a data matrix both pass through a true bicluster, the probability that they locate a genuine seed for the bicluster should be high. Therefore, we can identify true biclusters by growing genuine seeds, while the growth of false seeds would phase out. Clearly, a critical step of this strategy is to optimize a trend-preserving seed for each pair of rows in the data matrix. An optimized seed, i.e., a trend-preserving 2×L submatrix with L maximized, is likely to be genuine if the pair of rows both pass a bicluster to be detected, or to be false, otherwise. Mathematically, finding such an optimized 2×L submatrix is a combinatorial optimization problem with itself being quite interesting. We note that this problem can be exactly transformed into an easy problem of finding the longest path in a pseudo direct acyclic graph (DAG). Therefore, ARBic simply calls the existing algorithm RecBic to find narrow biclusters if the data matrix is relatively narrow, and finds longest paths in pseudo DAGs otherwise, thereby identifying broad biclusters. ARBic integrates the data preprocessing step used in QUBIC(see Supplementary Materials for details). The data preprocessing step of QUBIC is proved to be biologically significant for preprocessing gene expression data [16, 19, 25]. Before detailing ARBic, we first describe the subroutine of seed optimization for finding broad biclusters.

# 2.1 Seed optimization

The subroutine finds an optimum seed for each pair of rows by identifying the longest path in a pseudo DAG constructed using the pair of rows as follows.

Step 1. Construction of pseudo DAG. Given a pair of rows $\mathrm{r_i}$ and $\mathrm{r_j}$ in the data matrix, we construct a graph $\mathrm{G_{ij}} = \{V, E\}$, where V is the set of vertices representing all the columns of the two rows, and E the

set of directed edgdes formed as follows: we add an edge (s, t) from node s (i.e. $\begin{pmatrix} a_{is} \\ a_{js} \end{pmatrix}$) to node t (i.e. $\begin{pmatrix} a_{it} \\ a_{jt} \end{pmatrix}$) (s ≠ t) if and only if $\begin{pmatrix} a_{is} \\ a_{js} \end{pmatrix} \geq \begin{pmatrix} a_{it} \\ a_{jt} \end{pmatrix}$, i. e. , $a_{is} \geq a_{it}$, and$a_{js} \geq a_{jt}$, an edge (s, t) is bidirectional if and only if $\begin{pmatrix} a_{is} \\ a_{js} \end{pmatrix} = \begin{pmatrix} a_{it} \\ a_{jt} \end{pmatrix}$. Notably, any connected subgraph induced by bidirected edges forms a clique. We contract each clique induced by bidirected edges into a single vertex, the resulting graph $G_{ij}^{*} = \{V^{*}, E^{*}\}$, is thus acyclic (Supplementary Figure S4-S6).

Step 2. Finding the longest directed path in $G_{ij}$. We first find the longest directed path $P^{*} = p_1^{*} \rightarrow p_2^{*} \rightarrow \cdots \rightarrow p_h^{*}$ in $G_{ij}^{*}$ using a dynamic approach, where $p_i^{*}$ is a vertex formed by contracting a clique of $k_i$ vertices $\left\{i_1, i_2, \cdots, i_{k_i}\right\}$. The length of $P^{*}$ is the sum of sizes of the cliques corresponding to the vertices in $P^{*}$, i.e., $|P^{*}| = \left|p_1^{*}\right| + \left|p_2^{*}\right| + \cdots + \left|p_h^{*}\right|$. Blooming the contracted cliques, we obtain the longest directed path $P = p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_h$ in $G_{ij}$, where $p_i = i_1 \rightarrow i_2 \rightarrow \cdots \rightarrow i_{k_i}$(see Supplementary Materials for details). By concatenating paths defined by each $p_i$ ($i = 1, 2, ..., h$), we obtain an optimum seed, a 2×|*P*| submatrix encoded by the pair of rows $r_i$ and $r_j$.

# 2.2 Optimum extension of a seed without noise

Prior to the extension, we need to define an auxiliary seed based on the longest path $P^{*} = p_1^{*} \rightarrow p_2^{*} \rightarrow \cdots \rightarrow p_h^{*}$ in$G_{ij}^{*}$, where $p_i^{*} = \left\{i_1, i_2, \cdots, i_{k_i}\right\}$, $1 \leq i \leq h$, is a vertex obtained by contracting a clique of vertices $i_1, i_2, \cdots, i_{k_i}$ in $G_{ij}$. To do so, we define a mapping $\phi: \{1, 2, ..., n\} \rightarrow \{1, 2, ..., h, N\}$, where $n = |V|$, such that

$$\phi(j) = i, \forall j \in p_i^{*}, i = 1, , 2, ..., h; \phi(j) = N, \forall j \in \{1, 2, ..., n\} \setminus \cup_{i=1}^{h} p_i^{*}.$$

Then the auxiliary seed is defined as a numerical vector *v* with its components defined as follows:

$v_j = i$ if and only if $\phi(j) = i$, and $v_j = D$if and only if $\phi(j) = N$.

We then restrict the extension to the columns j with $v_j \neq D$. We greedily recruit a new row into the current bicluster by finding the longest path in a pseudo DAG constructed using the auxiliary seed and the new row in the same way as that used in the seed optimization subroutine. We repeat the procedure until the

current bicluster $B = \{I, J\}$ with min$\{|I|, |J|\}$ maximized, where $I$ and $J$ are the subsets of row indices and column indices of the data matrix, respectively. We call $B$ a core bicluster.

## 2.3 Optimum extension of a seed with noise

To deal with noise in a data matrix, we extend a core bicluster by adding new rows and columns to it with certain entries violating the trend-preserving pattern. Let $B = \{I, J\}$ be a core bicluster obtained by optimum extension without noise. For a column $j \notin J$, by $c_j$ we denote the proportion of rows in $I$ which are trend-preserved under the condition $J \cup \{j\}$. For a row $i \notin I$, by $c_i$ we denote the proportion of columns in the *i*-th row whose trend is the same as the rows in *I* under conditions *J*. We extend the core bicluster by adding all the new rows *i* with $c_i > \alpha$, and all the new columns j with $c_j > \alpha$ to the core bicluster. By default, $\alpha = 0.9$.

## 2.4 Bicluster scoring function

Given a $t \times s$ bicluster in an $m \times n$ data matrix, the probability that it is trend-preserved by chance is $(2/s!)^t$ [15, 18]. Based on experiments on both simulation and real data, we found that the significance of a trend-preserving $t \times s$ bicluster can be better measured using the function $(2nm/s!)^t$. Biologically, the more conditions under which the $s$ genes are co-expressed, the more highly correlated the genes are. Therefore, to identify broad biclusters using the row-based strategy, we define the score of a $t \times s$ bicluster as:

$$\mathrm{BS} = -\log\left(\frac{2nmt}{s!}\right)^t$$

The optimum seed obtained via the seed optimization subroutine should be genuine if rows $r_i$ and $r_j$ pass through a true bicluster, and fake otherwise. Ben-Dor et. al [8] developed a technique to estimate the minimum number of rows that a true bicluster should have based on requirements about the significance of the to-be-identified biclusters. Since each pair of rows in a true bicluster are equally capable of providing a genuine seed, it is sufficient to guarantee that at least one pair of rows in each true bicluster to have an opportunity to be selected as a seed by the subroutine of seed optimization. Therefore, we may divide the rows in data matrix into subsets such that each true bicluster has at least one pair of rows belonging to one subset, where the number of subsets can be determined using an estimation formula from [8], and thus it is sufficient to enumerate the pairs of rows in each subset separately. After seeding, we sort all the seed candidates in a decreasing order in length. Starting with the longest seed as the current bicluster, we iteratively extend it by calling the optimum extension subroutine without noises to get as two times many core biclusters $B = \{I, J\}$ as we want with min$\{|I|, |J|\}$ maximized. We then further extend the core biclusters using the optimum extension subroutine with noises until reaching the prespecified noise level. We finally output all biclusters whose scores ranked top *o* with the default value set to 50.

## 2.5 The ARBic algorithm

Given a preprocessed input data matrix $M_{m\times n}$, ARBic proceeds as follows.

If $n \leq N$ (N is set to 500 as default value), call RecBic; otherwise

Step 1. Partitioning: partition the $m$ rows in $M_{m\times n}$ into $k$ subsets, where $k$ is obtained based on the p value (by default, p=0.05) of biclusters to be identified (Supplementary Figure S3).

Step 2. Seeding: Find all the significant seeds by calling the seed optimization subroutine.

Step 3. Extening without noise: Find all the significant core biclusters by calling the optimum extension subroutine without noise.

Step 4. Extening with noise: find all the significant biclusters by calling the optimum extension subroutine with noise.

Step 5. Output top *o* biclusters according to their BS scores (by default, *o* = 50).

# 3. Results

To evaluate ARBic, we compared it with seven currently popular biclustering algorithms, including CPB[11], FABIA[22], ISA[23], OPSM[18], QUBIC2[19], EBIC[15] and UniBic[16] on various simulated datasets as well as real datasets. These algorithms were chosen because they have been proved to be the top algorithms for finding biclusters in previous studies. We ran all the algorithms with the parameters suggested in their publications on different types of simulated datasets with biclusters of different shapes and patterns, and on five real datasets collected from[27]. We only restricted our attention to the row-based strategy on the relatively broad datasets since otherwise RecBic[26] could be called to run on. RecBic performs almost perfectly when the data matrix is relatively narrow [26](much more rows than columns). However, it could be collapsed as the data matrix goes broader. Therefore, ARBic could be a powerful tool to identify both narrow and broad biclusters totally submerged in gene expression datasets.

# 3.1 Evaluation Criterion

# 3.1.1 Evaluation Criterion on simulated data

Since there is no golden benchmark real datasets to validate biclustering algorithms, it is a common practice to test them on simulated datasets using two metrics, recovery and relevance scores, introduced by Prelić et al. [20], based on the match scores (Jaccard coefficients) [20] between the predicted biclusters and genuine ones. Specifically, for two biclusters $b_1$ and $b_2$, the match score between them is defined as the ratio between the number of genes in their intersection and the number of genes in their union, i.e., $ms(b_1, b_2) = |b_1 \cap b_2| / |b_1 \cup b_2|$, which measures the similarity between the two biclusters. For two sets of biclusters $M_1$ and $M_2$, the match score between them is defined as [20]:

$$s\left(M_1, M_2\right) = \frac{1}{\left|M_1\right|} \sum_{b_1 \in M_1} \max_{b_2 \in M_2} ms(b_1, b_2),$$

which measures the average similarity between biclusters in $M_1$ and $M_2$. Let G and D be the sets of genuine and predicted biclusters, respectively, then we call s(G, D) and s(D, G) the recovery and relevance score, respectively.

# 3.1.2 Evaluation Criterion on real expression data

Since the genuine biclusters in the datasets are unknown, we evaluated each identified bicluster by each tool using the KEGG biological pathway enrichment method [28], with a Benjamini-Hochberg adjusted $p < 0.05$. A bicluster is enriched if and only if it is enriched for at least one pathway in the KEGG database. The enrichment score of a set of biclusters obtained by a tool is defined as the fraction of enriched biclusters in the set. For an enriched bicluster, the gene enrichment ratio [28] of the bicluster is defined as the ratio of the number of bicluster genes in pathways in the KEGG database to the total number of bicluster genes. The gene enrichment score of a set of biclusters obtained by a tool is defined as the average of the gene enrichment ratios of biclusters. The higher the gene enrichment score of a bicluster, the more portion of genes in the bicluster that are enriched for pathways, which means that the bicluster is more likely biologically meaningful. F1-score which balances these two scores is the harmonic mean of bicluster enrichment score and gene enrichment score.

# 3.2 Test on simulated datasets

# 3.2.1 Generation of artificial datasets

We first generated a background matrix of size 600×600. The elements of the background matrix were randomly generated by the standard normal distribution of N (0,1). We then randomly selected a predetermined number of submatrices of prespecified numbers of rows and columns, in the background matrix, and rearranged the elements of each submatrix such that all the rearranged submatrices are of prespecified properties. A trend-preserving bicluster was generated by fixing one row in the selected submatrix and rearranging the elements of other rows in the submatrix according to the trend of the fixed row. To better mimic the real data, we allow some elements in each row being equal, i.e., the number of different values in a row may be less than n, the number of columns of the background matrix. In our experiments, we generated three artificial datasets. 1) datasets implanted with complex trend-preserving biclusters; 2) datasets implanted with overlapping biclusters; 3) datasets implanted with noisy biclusters. Note that no noise levels are involved in datasets 1) and 2, and no overlaps between biclusters in datasets 1) and 3). We also evaluated the algorithms on the datasets implanted with six types of biclusters, separately (Supplementary Figure S7).

# 3.2.2 ARBic almost perfectly identifies trend-preserving biclusters

We tested the eight algorithms on an artificial data matrix of size 600×600 implanted with six trend-preserving biclusters of size 100×100, and repeated the experiment four times to reduce variation. As shown in Figure 2, ARBic substantially outperformed the other algorithms with both mean relevance and recovery scores of 0.99. CPB ranks the second-best with mean relevance and recovery scores of 0.9 and 0.6, respectively. UniBic hovers its relevance and recovery scores around 0.55 and 0.48, respectively. Although EBIC is comparative to ARBic in finding narrow biclusters[15, 26], it collapses as the number of columns increases because it needs more iterations to get rid of the local optimum solution. FABIA performs stably with both mean relevance and recovery scores of 0.4. ARBic also shows a leading advantage in data implanted with various biclustering patterns (Supplementary Figure S7).

### 3.2.3 ARBic almost perfectly identifies overlapping biclusters

Gene regulation is highly combinatorial and gene products can participate in multiple pathways, which means that overlaps between genes of biclusters in expression data matrices are widespread. We compared the algorithms for identifying overlapping biclusters on four artificial datasets implanted with six overlapping 100×100 biclusters in a 600×600 background matrix having four overlapping levels: 0×0 (no overlap), 30×30, 40×40 and 50×50, where overlapping level $s \times t$ represents two biclusters having s rows and t columns in common. To ensure the overlapping biclusters to be trend-preserved, we set the elements in the overlapping region to a constant value. As shown in Figure 3, ARBic still substantially outperforms the other algorithms for identifying overlapping biclusters. ARBic achieves almost perfect both mean relevance and recovery scores (0.99) on datasets with overlapping levels of 0×0, 30×30, 40×40, while its performance drops on the dataset with the overlapping level of 50×50, with both mean relevance and recovery scores of 0.85. QUBIC2 performs stably, but with poor scores. The relevance score of CPB decreases sharply as the overlapping level goes up, and both FABIA and EBIC perform poorly as the overlapping level goes up.

### 3.2.4 ARBic is robust to noise

We define the noise level in a bicluster to be the maximum of row noise levels in the bicluster, where the row noise level is the ratio between the number of elements whose removal will make the trend-preserving row to the consensus of the bicluster and the number of all elements in the row. We evaluated the impact of noise levels on the performance of the eight tools on datasets containing three biclusters of size 100×100 with different noise levels (0, 0.1, 0.2 and 0.3) in a background matrix of size 600×600 (see Methods). As shown in Figure 4a-d, ARBic consistently achieves the highest relevance and recovery scores on all the datasets, suggesting that it has the highest tolerance to noise among the eight tools. UniBic, the runner-up overall, shows its competitive results. Although FABIA scores in the middle of the packs, it is also highly resistant to noise.

## 3.3 Performance of eight tools on Real datasets

Since RecBic aimed to identify narrower bicluster has been demonstrated to be performed extremely well on data matrices with a relatively small number of columns/conditions, we compared ARBic with other

algorithms on five real datasets with a larger number (>500) of columns (details are in Supplementary Materils). These datasets include two from E. coli, two from yeast, and one from human tissues, which have been used in [27]( Supplementary Table S1). In order to minimize the deviation of selecting parameters of tools, we ran more than 20 times using different parameters each time for each method, and used the median score as the evaluation criterion. Since OPSM cannot get results in an acceptable time on the human dataset, both scores of OPSM on this data are set to 0.

As shown in Figure 5, ARBic reaches the highest median f1-score of 0.64 overall, and followed by QUBIC2 and UniBic with f1-scores of 0.34 and 0.35, respectively. On the E. coli Colombos and E. coli Dream5 datasets, ARBic substantially outperforms other algorithms. All algorithms sharply decreased in performance over the human dataset, while ARBic still keeps the best f1-score of 0.25. On the Yeast Dream5 dataset, UniBic achieves the best f1-score of 0.72, followed by ARBic with an f1-score of 0.68. EBIC, ISA and FABIA however all fluctuate widely across different datasets. As shown in Figure6, ARBic reaches the best genes enrichment score of 0.53 and the second highest bicluster enrichment score of 0.84 overall. Other algorithms have their own advantages. EBIC achieves the highest bicluster enrichment score of 0.91 overall, but its median genes enrichment score is merely 0.09. EBIC tends to find more genes, which makes the biclusters it finds easier to be enriched, while leading to a lot of false-positive genes. UniBic achieves a similar bicluster enrichment score to ARBic. QUBIC2 and UniBic are ranked suboptimum in gene enrichment scores. QUBIC2 achieves the second-best genes enrichment score of 0.25.

# 4. Complexity Estimation

The number of seeds enumerated by the ARBic is upper bounded by $O(m^2 / k)$. The time to construct a DAG from a seed and the time to find a longest path of a DAG is $O\left(q^2 n^2\right)$. The time consumed by extending the seed into a bicluster is upper bounded roughly by $O\left(q^2 n^2 m^2\right)$. Therefore, the complexity of ARBic is $O\left(q^2 m^2 n^2\right)$, where q is the percentage of the condition that the gene expression level is up-(down-) regulated to the total conditions. Therefore, the ARBic is quite efficient in practice because the parameter *q* is usually set to smaller. As shown in Figure7, ARBic can get advanced results shown in Fig. 6 within an acceptable time.

# 5. Discussion

Ideally, a biclustering algorithm should be able to identify both narrow and broad biclusters in a data matrix. To our best knowledge, ARBic is the first such an algorithm with such capability. ARBic achieves this goal by integrating our earlier RecBic algorithm targeting narrow biclusters, and new graph-theoretic algorithm targeting broad biclusters. Unlike RecBic with time complexity of $O(mn^3)$, which cannot run on the even intermediately broad data matrix, the new algorithm with time complexity of $O(q^2 m^2 n^2)$, is able

to run on quite broad data matrix. The algorithm finds a bicluster based on a seed associated with the longest path on a DAG. Therefore, the seed is globally optimized, the first of its kind, to our best knowledge, which accurately uses graphs to model trend-preserving biclusters, so that ARBic compensates for the drawback of UniBic that locates a seed by identifying the longest common subsequence between two sequences. Moreover, using a new objective function, the algorithm is able to more accurately discriminate the true positives from false negatives. Comparing the algorithm with other seven salient biclustering algorithms, we showed that our algorithm substantially and consistently outperforms all these algorithms in identifying various kinds of biclusters implanted in simulated datasets as well as functional biclusters in real gene expression datasets. Our algorithm can not only get more functionally related modules, but also have fewer false positive co-expression genes in the modules. Therefore, ARBic can be very useful to analyze gene expression data and find the co-regulated gene and underlying gene regulatory networks.

As the cost of sequencing decreases, there will be more and more single-cell gene expression data generated with higher column dimensions. For example, researchers can easily obtain scRNA-Seq data from tens of thousands of cells using 10x Genomics platforms. Although the ARBic has been highly efficient, it will consume a lot of time when dealing with data of this scale. However, ARBic is easily parallelizable, we are currently undertaking this effort to make it more useful in practice.

# 6. Conclusions

Biclustering has been widely used technology in analyzing gene expression data as it provides flexibility to identify co-expressed genes under some but not necessarily all conditions/samples, which the traditional clustering methods lack. However, all the existing biclustering algorithms were designed to identify a special kind of biclusters, broader or narrow but both. In this study, we developed a novel bicluster algorithm ARBic to discover trend-preserving biclusters, be it narrow or broad, in any type of data matrices. ARBic first identifies globally optimized seeds, and then grows them into full-sized biclusters using a greedy strategy. The ARBic was developed by integrating column-based and row-based strategies into the whole procedure. The column-based strategy borrowed from ReBic[26] prefers to narrow bicluters, and the row-based strategy newly designed in this article prefers to broader ones. Mathematically, ARBic globally optimizes the solutions in each iteration. To our best knowledge, ARBic is the first to identify a bicluster with a global optimum in each iteration. Comparing ARBic with seven existing state-of-the-art biclustering algorithms, we shown that RecBic substantially and consistently outperformed all of them in identifying various kinds of biclusters in simulated datasets as well as functionally enriched biclusters in real gene expression datasets. ARBic can be a useful tool for analyzing gene expression data and elucidating transcriptional regulation networks.

# 7. Declarations

Abbreviations:

Not applicable.

Ethics approval and consent to participate:

Not applicable.

Consent for publication:

Not applicable.

Availability of data and material:

Code: https://github.com/holyzews/ARBic.

Data: https://doi.org/10.5281/zenodo.5121018.

Competing interests:

The authors declare that there are no competing interests.

Funding:

This work has been supported by National Science Foundation of China with codes 11931008 and 61771009, and by Ministry of Science and Technology of China with code 2020YFA0712400.

Authors' contributions:

GL conceived and designed the experiments. XL performed the experiments and analyzed the data. GL and ZS wrote the paper.

# References

1. Reuter, J.A., D.V. Spacek, and M.P. Snyder, *High-throughput sequencing technologies.* Mol Cell, 2015. **58**(4): p. 586-97.

2. Hwang, B., J.H. Lee, and D. Bang, *Single-cell RNA sequencing technologies and bioinformatics pipelines.* Experimental and Molecular Medicine, 2018. **50**(8): p. 1-14.

3. Van Dam, S., et al., *Gene co-expression analysis for functional classification and gene–disease predictions.* Briefings in Bioinformatics, 2017. **19**(4): p. 575-592.

4. Morgan, J.N. and J.A. Sonquist, *Problems in the Analysis of Survey Data, and a Proposal.* Journal of the American Statistical Association, 1963. **58**(302): p. 415-434.

5. Hartigan, J.A., *Direct Clustering of a Data Matrix.* Journal of the American Statistical Association, 1972. **67**(337): p. 123-129.

6. Cheng, Y. and G.M. Church. *Biclustering of Expression Data*. in *intelligent systems in molecular biology*. 2000.

7. Bryan, K. and P. Cunningham. *Bottom-up biclustering of expression data*. in *2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*. 2006. IEEE.

8. Carmonasaez, P., et al., *Biclustering of gene expression data by non-smooth non-negative matrix factorization.* BMC Bioinformatics, 2006. **7**(1): p. 78-78.

9. Gu, J. and J.S. Liu, *Bayesian biclustering of gene expression data.* BMC Genomics, 2008. **9**(1): p. 1-10.

10. Henriques, R.T., F.L. Ferreira, and S.C. Madeira, *BicPAMS: software for biological data analysis with pattern-based biclustering.* BMC Bioinformatics, 2017. **18**(1): p. 82-82.

11. Bozdag, D., J.D. Parvin, and U.V. Catalyurek. *A Biclustering Method to Discover Co-regulated Genes Using Diverse Gene Expression Datasets*. in *international conference on bioinformatics*. 2009.

12. Kung, S., M. Mak, and I. Tagkopoulos, *SYMMETRIC AND ASYMMETRIC MULTI-MODALITY BICLUSTERING ANALYSIS FOR MICROARRAY DATA MATRIX.* Journal of Bioinformatics and Computational Biology, 2006. **4**(02): p. 275-298.

13. Li, H., et al., *A GENERAL FRAMEWORK FOR BICLUSTERING GENE EXPRESSION DATA.* Journal of Bioinformatics and Computational Biology, 2006. **4**(4): p. 911-993.

14. Reiss, D.J., N.S. Baliga, and R. Bonneau, *Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks.* BMC Bioinformatics, 2006. **7**(1): p. 280-280.

15. Orzechowski, P., et al., *EBIC: an evolutionary-based parallel biclustering algorithm for pattern discovery.* Bioinformatics, 2018. **34**(21): p. 3719-3726.

16. Wang, Z., et al., *UniBic: Sequential row-based biclustering algorithm for analysis of gene expression data.* Scientific Reports, 2016. **6**(1): p. 23466-23466.

17. Eren, K., et al., *A comparative analysis of biclustering algorithms for gene expression data.* Briefings in Bioinformatics, 2013. **14**(3): p. 279-292.

18. Bendor, A., et al., *Discovering local structure in gene expression data: the order-preserving submatrix problem.* Journal of Computational Biology, 2003. **10**: p. 373-384.

19. Xie, J., et al., *QUBIC2: A novel and robust biclustering algorithm for analyses and interpretation of large-scale RNA-Seq data.* Bioinformatics, 2019. **36**(4): p. 1143-1149.

20. Madeira, S.C. and A.L. Oliveira, *Biclustering Algorithms for Biological Data Analysis: A Survey.* IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2004. **1**(1): p. 24-45.

21. Lazzeroni, L. and A. Owen, *Plaid models for gene expression data.* Statistica sinica, 2002: p. 61-86.

22. Hochreiter, S., et al., *FABIA: factor analysis for bicluster acquisition.* Bioinformatics, 2010. **26**(12): p. 1520-1527.

23. Bergmann, S., J. Ihmels, and N. Barkai, *Iterative signature algorithm for the analysis of large-scale gene expression data.* Physical Review E, 2003. **67**(3): p. 031902.

24. Murali, T.M. and S. Kasif. *Extracting conserved gene expression motifs from gene expression data*. in *pacific symposium on biocomputing*. 2002.

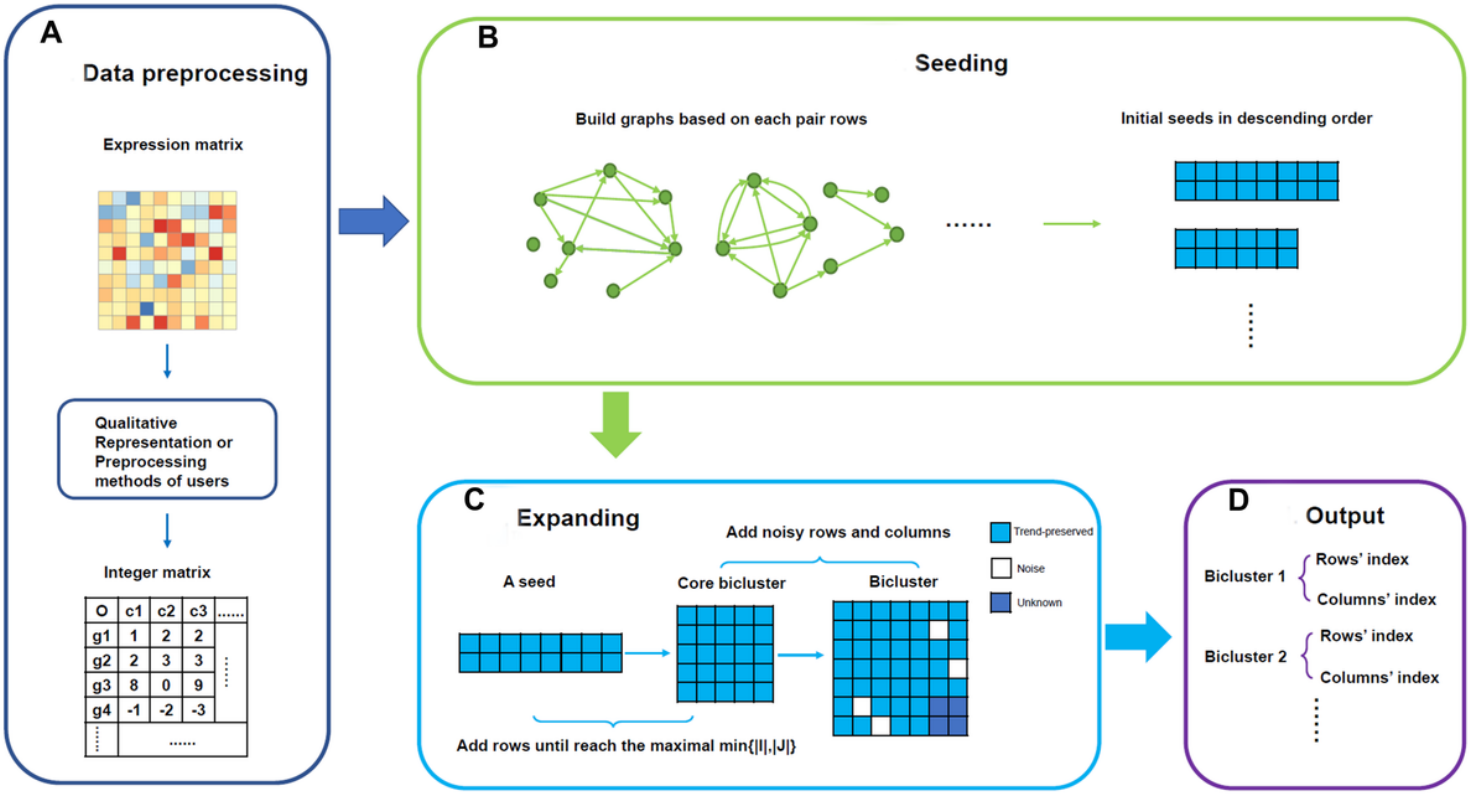
25. Li, G., et al., *QUBIC: a qualitative biclustering algorithm for analyses of gene expression data.* Nucleic Acids Research, 2009. **37**(15).

26. Liu, X., et al., *RecBic: a fast and accurate algorithm recognizing trend-preserving biclusters.* Bioinformatics, 2020. **36**(20): p. 5054-5060.

27. Saelens, W., R. Cannoodt, and Y. Saeys, *A comprehensive evaluation of module detection methods for gene expression data.* Nature Communications, 2018. **9**(1): p. 1090.

28. Yu, G., et al., *clusterProfiler: an R package for comparing biological themes among gene clusters.* Omics-a Journal of Integrative Biology, 2012. **16**(5): p. 284-287.

# Figures




## Figure 1

Workflow of ARBic

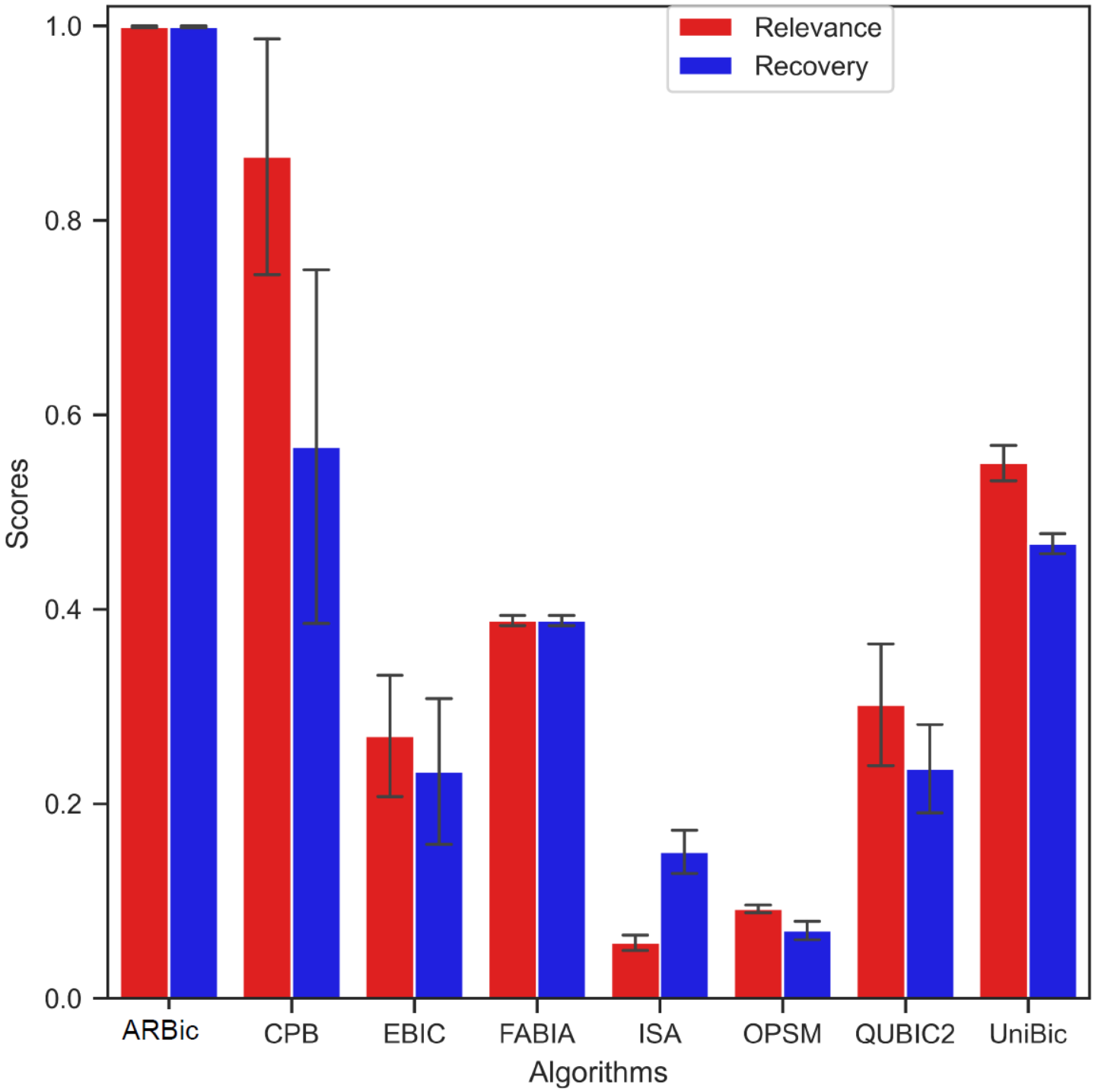
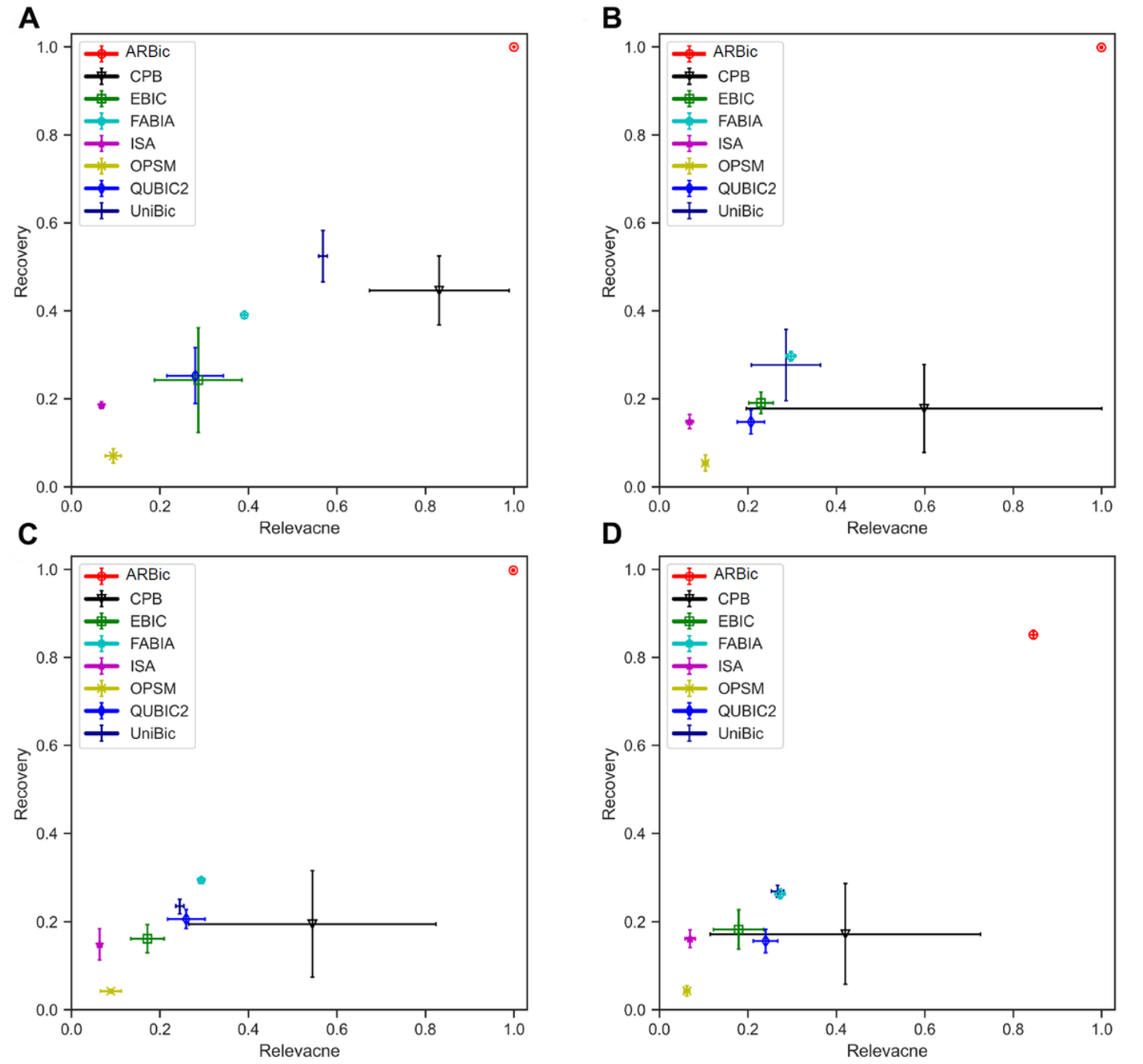


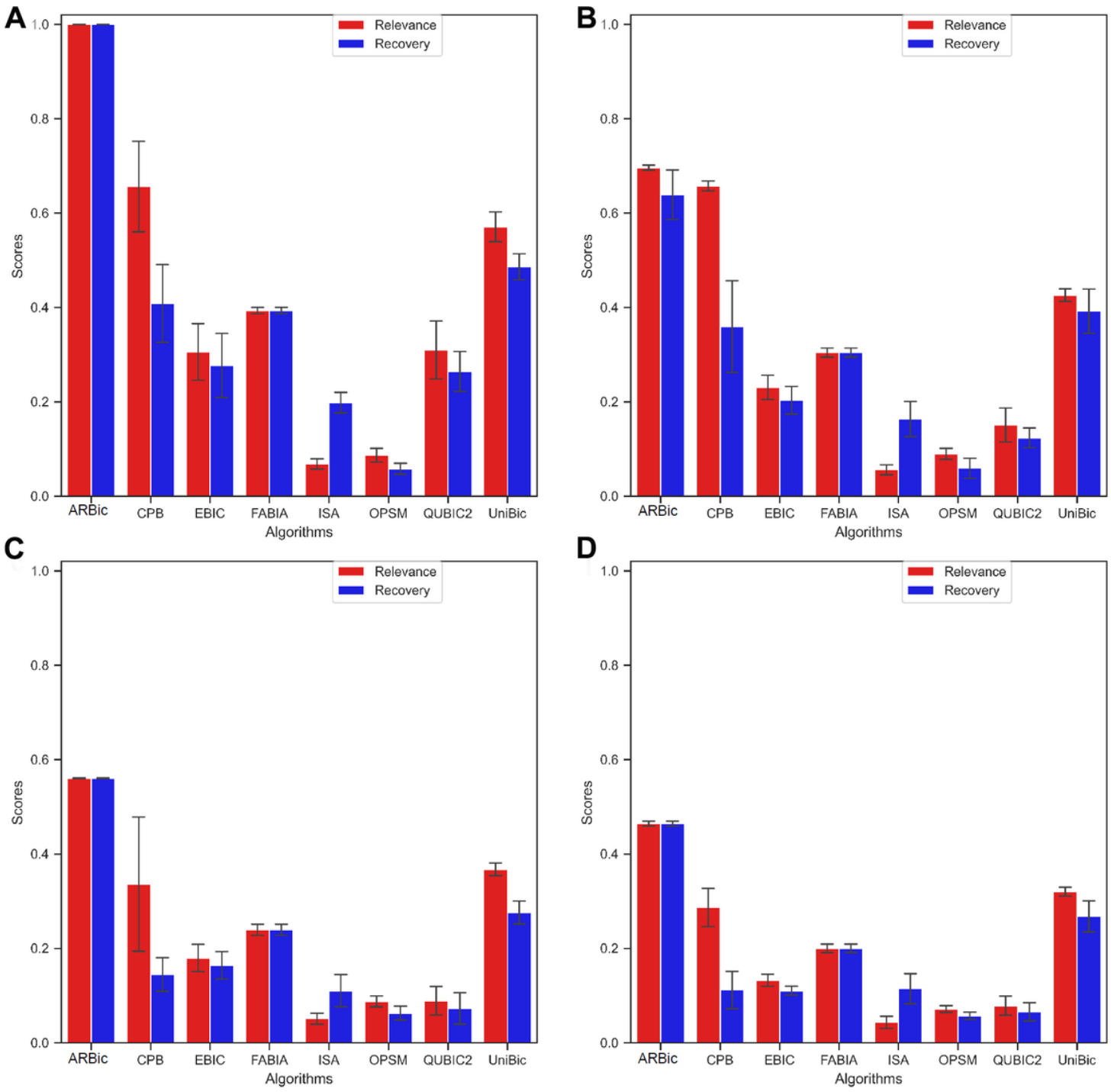

**Figure 2**

Comprasion of the performce of eight algroritthms on datasets implanted with six trend-preserving biclusters measured by relevance and recovery scores. The error bars indicate standard error deviations.



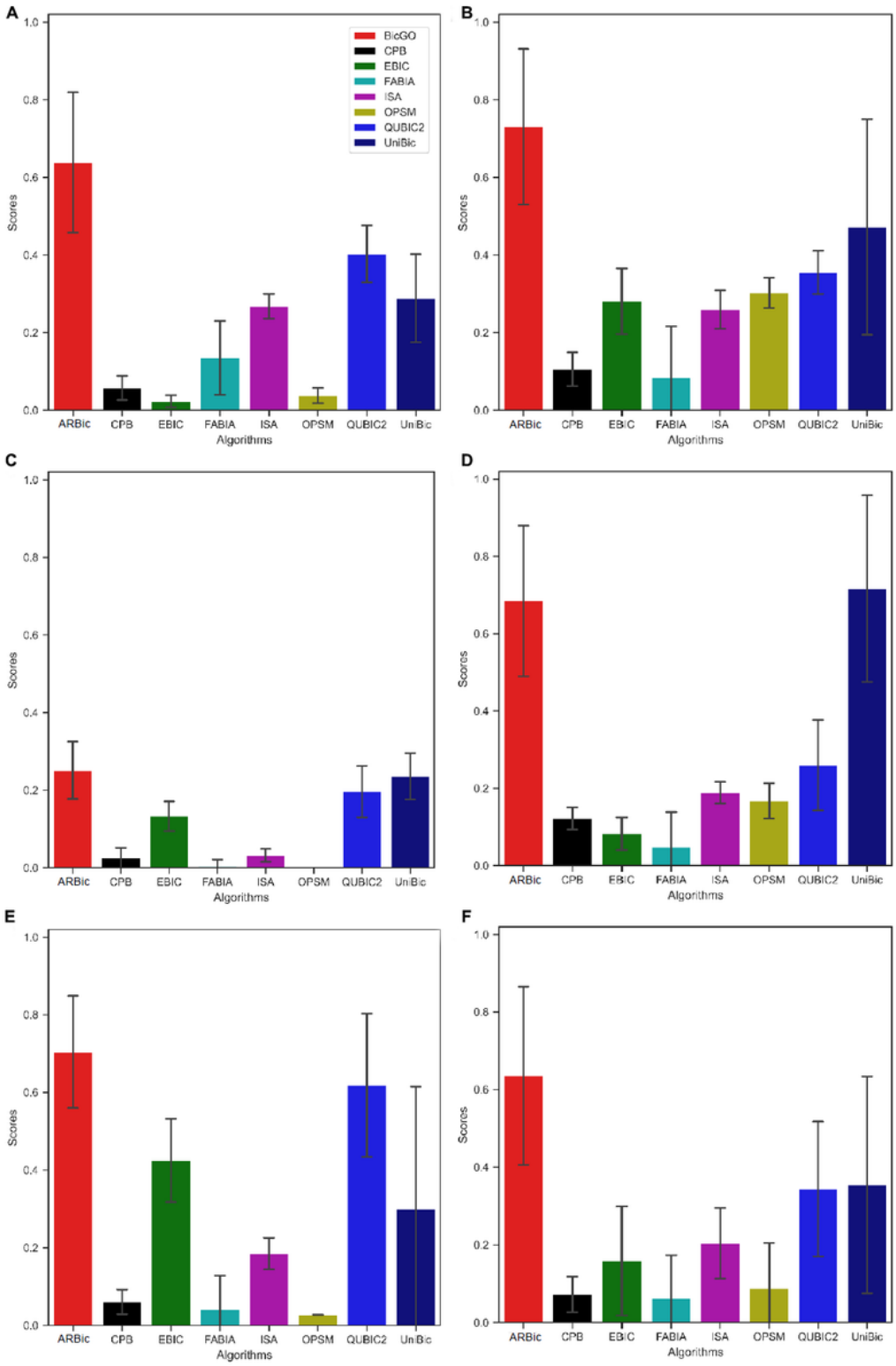

## Figure 3

Relationship between relevance and recovery scores for the eight algorithms on datasets of different overlap levels. A) Results on the dataset with non-overlapping biclusters. B) Results on the dataset with the biclusters with an overlapping level of 30×30. C) Results on the dataset with the biclusters with an overlapping level of 40×40. D) Results on the dataset with the biclusters with an overlapping level of 50×50.



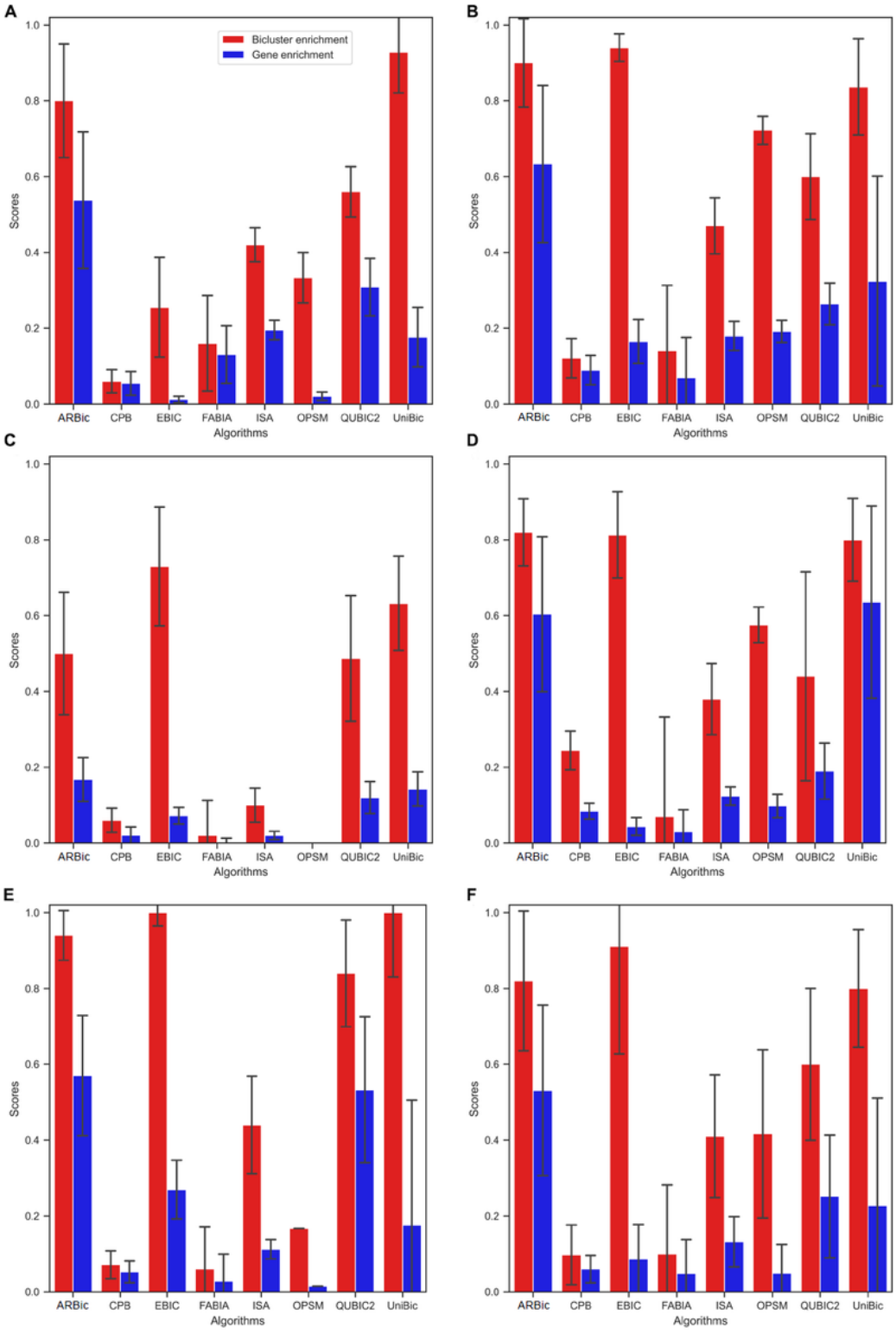

## Figure 4

Comprasion of the performce of eight algroritthms on datasets of different noise levels measured by relevance and recovery scores. A) Results on the dataset with non-noise biclusters. B) Results on the dataset with the biclusters noise level of 0.1. C) Results on the dataset with the biclusters noise level of 0.2. D) Results on the dataset with the biclusters noise level of 0.3. The error bars indicate standard error deviations.




## Figure 5

Comparison of the performance of the eight algorithms on five real datasets measured by the F1-score. A) E. coli colombos dataset; B) E. coli Dream5 dataset; C) Human seek gpl5175 dataset; D) Yeast Dream5 dataset; E) Yeast GPL2529 dataset; F) The avavage on the five datasets.




## Figure 6

Comparison of the performance of eight algorithms on five real datasets measured by bicluster enrichment and gene enrichment scores. A) Yeast GPL2529 dataset; B) Yeast Dream5 dataset; C) Human seek GPL5175 dataset; D) E. coli colombos dataset; E) E. coli Dream5 dataset; F) Average on the five datasets.

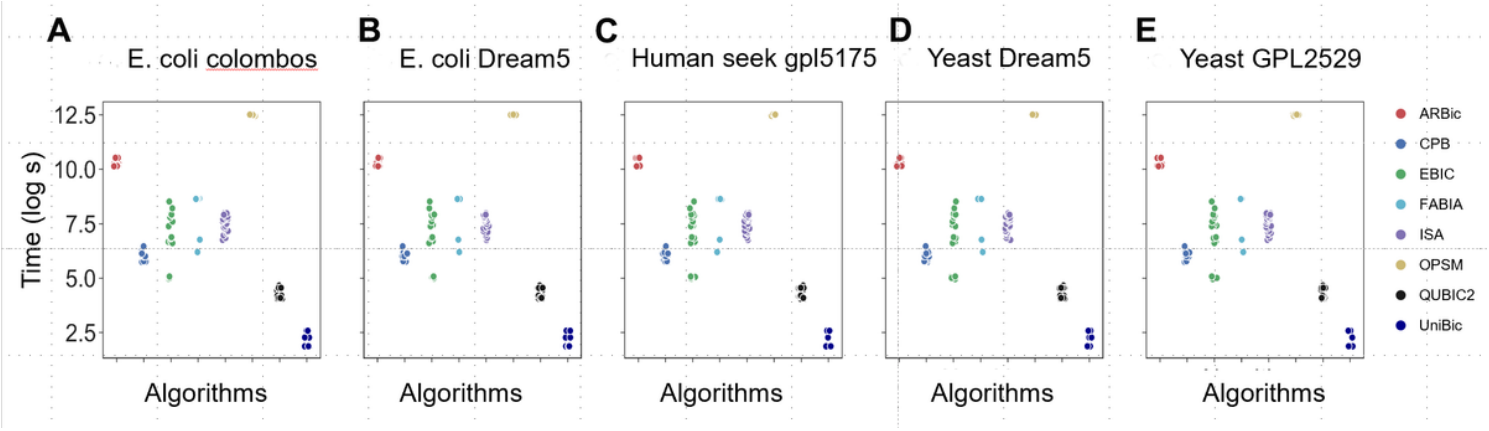



**Figure 7**

Comparison of the running time of eight algorithms on five real datasets.

# Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplymentaryMaterialsGLi105.docx