Comparison of sparse biclustering algorithms for gene expression datasets

Kath Nicholls[1] and Chris Wallace[1,2]

[1]Cambridge Institute for Therapeutic Immunology and Infectious Disease, University of Cambridge, Cambridge, CB2 0AW, UK
[2]MRC Biostatistics Unit, Cambridge Institute of Public Health, Cambridge, CB2 0SR, UK

Abstract

Gene clustering and sample clustering are commonly used to find patterns in gene expression datasets. However, in heterogeneous samples (e.g. different tissues or disease states), genes may cluster differently. Biclustering algorithms aim to solve this issue by performing sample clustering and gene clustering simultaneously. Existing reviews of biclustering algorithms have yet to include a number of more recent algorithms and have based comparisons on simplistic simulated datasets without specific evaluation of biclusters in real datasets, using less robust metrics.

In this study we compared four classes of sparse biclustering algorithms on a range of simulated and real datasets. In particular we use a knockout mouse RNA-seq dataset to evaluate each algorithm's ability to simultaneously cluster genes and cluster samples across multiple tissues. We found that Bayesian algorithms with strict sparsity constraints had high accuracy on the simulated datasets and didn't require any post-processing, but were considerably slower than other algorithm classes. We assessed whether non-negative matrix factorisation algorithms can be repurposed for biclustering and found that, although the raw output was poor, after using a sparsity-inducing post-processing procedure we introduce, one such algorithm was one of the most highly ranked on real datasets. We also exhibit the limitations of biclustering algorithms by varying the complexity of simulated datasets. The algorithms generally struggled on simulated datasets with a large number of implanted factors, or with a large number of genes. In real datasets, the algorithms rarely returned clusters containing samples from multiple tissues, which highlights the need for further thought in the design and analysis of multi-tissue studies to avoid differences between tissues dominating the analysis.

Code to run the analysis is available at `https://github.com/nichollskc/biclust_comp`, including wrappers for each algorithm, implementations of evaluation metrics, and code to simulate datasets and perform pre- and post-processing. The full tables of results are available at `https://doi.org/10.5281/zenodo.4317556`

1 Introduction

Clustering can be used in two main ways to analyse gene expression datasets [1]. The first is to cluster the samples, finding groups of samples that have similar expression in all genes. This can be used, for example, to find subgroups of disease [2]. The second is to cluster the genes, finding groups of genes that have similar expression across all samples. Finding such groups of genes has many useful applications such as inferring function using guilt by association and inferring regulatory relationships [3].

Instead of clustering only samples or only genes, biclustering algorithms find groups of samples that have similar expression in some subset of the genes, effectively clustering both genes and samples simultaneously. Such a group is called a *bicluster* and we say that the bicluster consists of a set of samples and a set of genes. Biclustering has three main advantages over normal clustering. Firstly, it can discover meaningful groups that would not be detected using normal clustering; in complex datasets, many interesting groupings of genes will not hold across all samples. For example, we might expect genes to cluster differently in different cell types. Secondly, it provides a link between sets of genes and sample traits such as disease or sex. For example, if a biclustering algorithm returns a bicluster consisting of all the samples from patients with a given disease and a small set of genes then we can hypothesise that the set of genes might have biological importance for the disease. Finally, biclustering algorithms are additive, allowing the algorithm to learn biclusters corresponding to confounders, such as batch or sex, and adjust for these confounders whilst simultaneously extracting biologically interesting biclusters. In this study we have focused on identifying algorithms that should be able to identify sparse biclusters in a complex bulk RNA-seq dataset, such as one including samples from multiple cell types.

There exist previous reviews of biclustering algorithms [4–7], but we hope to improve on them in the following ways. First, we include new classes of algorithm yet to be considered in independent comparison studies. In particular, we include non-negative matrix factorisation algorithms, which we believe can be repurposed for biclustering, tensor factorisation algorithms, which aim to improve performance by sharing information across tissues, and two Bayesian algorithms which allow for a mixture of sparse and dense biclusters. Second, we use more robust metrics. Horta and Campello investigated metrics used to evaluate similarity between biclusterings, and found problems with many of the metrics used by previous comparison papers [8]. In this study we use one of the two metrics recommended by Horta and Campello, which was shown to satisfy all but one of their criteria. Third, we narrow the gap between real and simulated datasets. Previous reviews have often used unrealistically simplistic simulated datasets, such as using only $K = 1, 2, 3, 4, 5$ biclusters, leading to discrepancies between the conclusions they draw on simulated and on real datasets [6]. In this study we simulate datasets from a wider range of complexities, including datasets closer in complexity to real datasets than those included in previous reviews. A final key flaw of existing comparison studies is the lack of evaluation of biclustering ability on real datasets. In the absence of known structure in the real gene expression datasets used for evaluation, previous reviews have evaluated sample clustering ability and gene clustering ability separately. We carefully chose a knockout mouse RNA-seq dataset that allows linked analysis of sample clustering and gene clustering, thus allowing direct evaluation of biclustering on real datasets.

2 Methods

Here we discuss the algorithms compared, the datasets they are tested on and the evaluation metrics used to score their performance. Similar to previous reviews, we use a mixture of simulated and real datasets. Simulated data is important as it allows more precise evaluation of performance, since the true structure of the data is known. However, it is difficult to exactly mimic the noise and structure of real gene expression datasets, so it is also important to see whether the algorithms can handle the noise structure of real datasets.

2.1 Algorithms compared

We chose most promising algorithms from four classes of algorithm, focusing on sparse algorithms (Table 1).

We define a matrix $Y \in \mathbb{R}^{n \times p}$ where entry Y_{ij} gives the expression of gene j in sample i. This can either be the raw read count from an RNA-seq experiment, or a normalised count which has been adjusted for sample-specific effects such as library size, or gene-specific effects such as mean expression level. The typical approach to biclustering is to factorise this matrix as a product of two sparse matrices $X \in \mathbb{R}^{n\times K}$, which we call the sample loadings matrix, and $B \in \mathbb{R}^{p\times K}$, which we call the gene loadings matrix, with error matrix ε:

$$Y = XB^T + \varepsilon \tag{1}$$

The individual algorithms are described in detail in Section S1. Here we discuss why each algorithm was chosen for inclusion in this study and group the algorithms as *Popular*, *Adaptive*, *NMF* and *Tensor*.

2.1.1 Popular algorithms

We include two algorithms which have been included in previous reviews, which we use as a baseline to allow relative performance to be related to other comparison studies. FABIA [9] is a Bayesian algorithm using sparsity-inducing priors, included in a number of previous comparisons [3, 6, 7, 19]. Although our study focuses on sparse biclustering algorithms, we chose to also include Plaid [10, 11] even though it does not enforce sparsity, as it has often appeared as one of the better performing algorithms in other studies [6, 7] and its inclusion thus provides a helpful link to these studies.

2.1.2 Adaptive Bayesian algorithms

Like FABIA, BicMix [16, 17] and SSLB [18] use sparsity-inducing priors. The key difference with BicMix and SSLB is that they allow for both sparse and dense biclusters, and adapt the sparsity constraints to each bicluster. Neither has been included in previous comparisons but they have been compared against each other and against FABIA in the paper introducing SSLB, where both achieved much greater sparsity and accuracy than FABIA.

2.1.3 Non-negative matrix factorisation

Non-negative matrix factorisation (NMF) algorithms in general are not designed for biclustering, but since biclustering can be described as sparse matrix factorisation, NMF algorithms can recover biclusters if they use sufficiently strong sparsity constraints. We chose to include two examples of such algorithms: SNMF [13] and nsNMF [12]. The main advantage we expect these algorithms will have is speed, as they are computationally much simpler than many of the others included in this study.

2.1.4 Tensor algorithms

When applying an algorithm to data from multiple cell types, a natural extension to the two-dimensional algorithms presented so far is a three-dimensional algorithm which exploits similarity between corresponding samples in different cell types. We chose to include two algorithms which attempt this: SDA [15] and MultiCluster [14].

2.2 Algorithm parameters

The algorithms evaluated here, outlined in Table 1, have many parameters which can be tuned. Before running the full analysis, we conducted a parameter sweep (Section S2) to see if there were any parameter values that consistently improved the score relative to that when the default values were used. For most algorithm parameters, there was either no clear optimal value, or the default value was optimal. Thus for most algorithms we used the default parameters throughout this study. One key exception was BicMix, which has a parameter determining whether or not each gene gets transformed to a Gaussian distribution before the algorithm runs. Changing this parameter had a dramatic but inconsistent effect, so we decided to use two versions of BicMix: BicMix, using default behaviour of not transforming genes, and BicMix-Q, which does apply

Table 1: Summary of algorithms included in comparison. Algorithms are listed in groups: popular algorithms which have been included in previous reviews (*Popular*), non-negative matrix factorisations algorithms (*NMF*), tensor factorisation algorithms (*Tensor*) and Bayesian algorithms allowing for a mixture of sparse and dense biclusters, with strength of sparsity constraints adapting to the bicluster (*Adaptive*). With the exception of Plaid (Section S1.1), algorithms either factorise the gene expression matrix Y as $Y = XB^T + \varepsilon$ (matrix factorisation), where X is the samples loading matrix, and B is the gene loading matrix, or write it as a tensor product $Y = \sum_k a_k \otimes b_k \otimes z_k + \varepsilon$ (tensor product), where a_k gives the loadings for bicluster k for the individuals, b_k gives the loadings for genes and z_k gives the loadings for the tissues. Some algorithms use one language to implement the algorithm, and provide a 'wrapper' in another language. Where this occurs, the language given in the 'Version' column is the language used to interact with the algorithm (the wrapper), rather than the language that the implementation uses.

Class	**Name**	**Model**	**Sparsity**	**Version**	**References**
Popular	FABIA	Matrix factorisation	Laplacian prior	pyfabia::2016.8 *(Python)*[a]	[9]
	Plaid	Plaid		biclust::2.0.2 *(R)*[b]	[10, 11]
NMF	nsNMF	Matrix factorisation	smoothing matrix and non-negativity of X, B	nimfa::1.4.0 *(Python)*[c]	[12]
	SNMF	Matrix factorisation	non-negativity of gene loadings matrix B	nimfa::1.4.0 *(Python)*[d]	[13]
Tensor	MultiCluster	Tensor product	Tissue components z_k non-negative	MultiCluster 15-08-2018 *(MATLAB)*[e]	[14]
	SDA	Tensor product	Spike-and-slab prior on gene components b_k	SDA 02-05-2016[f]	[15]
Adaptive	BicMix	Matrix factorisation	Three Parameter Beta prior	BicMix 03-08-2019 *(R)*[g]	[16, 17]
	SSLB	Matrix factorisation	Mixture of Laplacian prior	SSLB 24-04-2020 *(R)*[h]	[18]

[a] `https://github.com/bioinf-jku/pyfabia/commit/6eaf5f87`
[b] `https://cran.r-project.org/web/packages/biclust/index.html`
[c] `http://nimfa.biolab.si/`
[d] `http://nimfa.biolab.si/`
[e] `https://github.com/Miaoyanwang/MultiCluster/commit/520b2542`
[f] `https://jmarchini.org/sda/`
[g] `https://github.com/chuangao/BicMix/commit/ba1cfdfc`
[h] `https://github.com/gemoran/SSLB/commit/115921f`

the Gaussian transformation before analysis. Full discussion of our investigation of parameter sensitivity are given in Section S2.

2.3 Simulated datasets

We simulated individual gene expression data for each gene as a sum across biclusters of negative binomial counts. Our base model for generating a gene expression dataset with p genes, m individuals and t tissues and with K potentially overlapping biclusters is illustrated in Figure 1 and described below:

1. For each bicluster $k = 1, \ldots, K$:
   (a) Select genes to include in bicluster k: first draw number of genes g_k uniformly from the set $\{\frac{p}{100}, \frac{p}{10}, \frac{p}{5}, \frac{p}{2}, p\}$ and then pick a random sample of g_k genes.
   (b) Select individuals to include in bicluster k: first draw number of individuals m_k uniformly from the set $\{\frac{m}{100}, \frac{m}{10}, \frac{m}{5}, \frac{m}{2}, m\}$ and then pick a random sample of m_k individuals.
   (c) Select tissues to include in bicluster k: first draw number of tissues t_k uniformly from the set $\{1, 2, \ldots, t\}$ and then pick a random sample of t_k tissues.
   (d) Sample bicluster-specific mean $\mu_k \sim \text{Gamma}(\alpha, \beta)$ using $\alpha = 2, \beta = \frac{1}{600}$.
   (e) Sample values in bicluster using negative binomial distribution with mean μ_k, shared parameter $p = 0.3$.
2. Add together values from all biclusters
3. Add background noise using negative binomial distribution

Formally the base model is:

$$\begin{aligned} Y_{ijl} &= \sum_k \delta_{ik}\gamma_{jk}\tau_{lk}E_{ijl}^{(k)} + B_{ijl} \\ E_{ijl}^{(k)} &\sim \text{NegBin}(n_k, p) \\ B_{ijl} &\sim \text{NegBin}(1, p) \\ n_k &= \frac{\mu_k p}{1-p} \\ \mu_k &\sim \text{Gamma}(\alpha, \beta) \end{aligned} \tag{2}$$

where δ_{ik}, γ_{jk} and τ_{lk} are binary indicators of membership of individual i, gene j and tissue l to bicluster k respectively, $E_{ijl}^{(k)}$ is the increase in expression of gene i in tissue l in individual i due to bicluster k and B_{ijl} is background noise. We chose to force the genes chosen in a bicluster to belong to a contiguous block rather than allowing genes from a bicluster to be scattered freely throughout the matrix, and did the same for the tissues and individuals chosen in a bicluster. This arrangement has little impact on the generality of the data but makes it easier to visualise the datasets.

We vary the size of the dataset, the number of biclusters (which also naturally changes the amount of overlap between biclusters) and the size of biclusters. We also introduce diversity by using different noise distributions. For Gaussian noise we use $E_{ijl}^{(k)} \sim \mathcal{N}(\mu_k, \sigma^2)$ and for noiseless datasets we use $E_{ijl}^{(k)} = \mu_k, B_{ijl} = 0$. A summary of the properties of all the simulated datasets is given in Table 2.

Previous reviews have also varied simulation parameters but have often used very small ranges such as $K = 1, 2, 3, 4, 5$ [6]. Real gene expression datasets are likely to be more complex than this, so we have used larger values of K: most of our simulated datasets have $K = 20$ but we consider values from $K = 5$ to $K = 400$.

The *Tensor* algorithms require an explicit breakdown of the samples into tissues. By listing the samples from each tissue in turn, with individuals in the same order within each tissue, we are able to use the *Tensor* algorithms on the same datasets as the remaining algorithms, allowing direct comparison between the classes of algorithm.

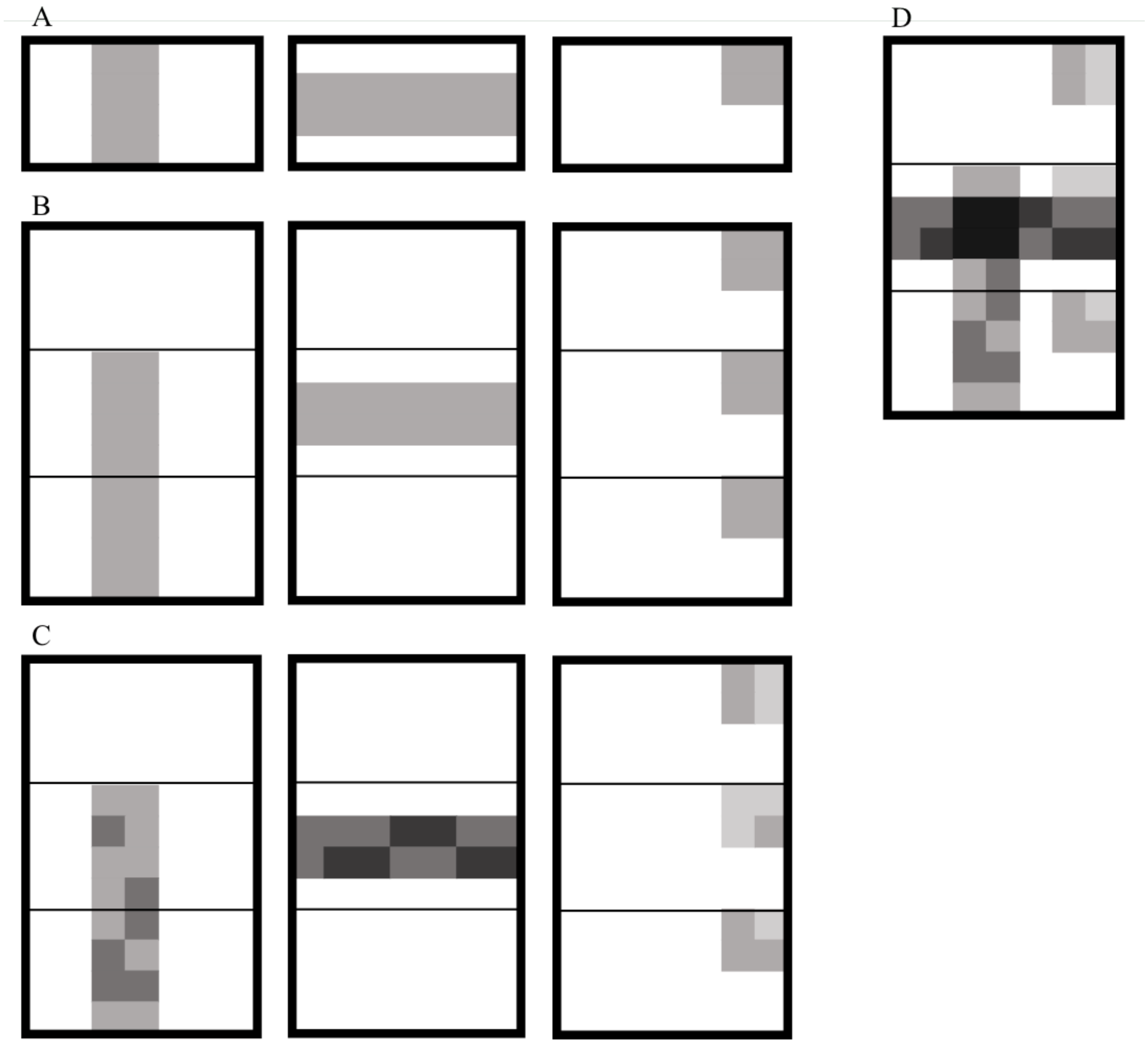

Figure 1: Illustration of process for simulating gene expression datasets with implanted biclusters. In this diagram (A) shows steps 1.a-1.b where membership for genes (columns) and individuals (rows) are sampled for 3 biclusters, (B) shows step 1.c for the 3 biclusters, where we extend the biclusters from size (m_k, g_k) to size $(m_k t_k, g_k)$ by sampling membership for tissues, (C) shows steps 1.d and 1.e where values for the bicluster members are sampled, with bicluster-specific means μ_k, (D) shows step 2 where the effects from the 3 biclusters are added together.

Table 2: Summary of simulated datasets. The attributes of the datasets are displayed in bold if they differ from the base dataset. N is the number of samples, T the number of tissues and G the number of genes in the dataset.

Name	N	T	G	Bicluster sizes	K	Noise
base	10	10	1000	mixed	20	NB(n_k, 0.3)
N50-T2	**50**	**2**	1000	mixed	20	NB(n_k, 0.3)
N10-T20	10	**20**	1000	mixed	20	NB(n_k, 0.3)
N100-T10	**100**	10	1000	mixed	20	NB(n_k, 0.3)
N500-T10	**500**	10	1000	mixed	20	NB(n_k, 0.3)
G100	10	10	**100**	mixed	20	NB(n_k, 0.3)
G5000	10	10	**5000**	mixed	20	NB(n_k, 0.3)
large-K20	**300**	**20**	**10000**	mixed	20	NB(n_k, 0.3)
negbin-medium	10	10	1000	mixed	20	NB(n_k, $\mathbf{0.1}$)
negbin-high	10	10	1000	mixed	20	NB(n_k, $\mathbf{0.01}$)
Gaussian	10	10	1000	mixed	20	$\boldsymbol{\mathcal{N}(\mu_k, 20^2)}$
Gaussian-medium	10	10	1000	mixed	20	$\boldsymbol{\mathcal{N}(\mu_k, 100^2)}$
Gaussian-high	10	10	1000	mixed	20	$\boldsymbol{\mathcal{N}(\mu_k, 300^2)}$
noiseless	10	10	1000	mixed	20	**No noise**
sparse	10	10	1000	**small**	20	NB(n_k, 0.3)
dense	10	10	1000	**large**	20	NB(n_k, 0.3)
sparse-square	10	10	1000	**small, square**	20	NB(n_k, 0.3)
dense-square	10	10	1000	**large, square**	20	NB(n_k, 0.3)
K5	10	10	1000	mixed	**5**	NB(n_k, 0.3)
K10	10	10	1000	mixed	**10**	NB(n_k, 0.3)
K50	10	10	1000	mixed	**50**	NB(n_k, 0.3)
K70	10	10	1000	mixed	**70**	NB(n_k, 0.3)
large-K100	**300**	**20**	**10000**	mixed	**100**	NB(n_k, 0.3)
large-K400	**300**	**20**	**10000**	mixed	**400**	NB(n_k, 0.3)

2.4 Real datasets

A key limitation of the existing reviews of biclustering algorithms is their inability to assess *simultaneous* clustering of samples and genes on real datasets, due to the absence of known biclusters in the data. In order to have predictable bicluster structure in a real dataset, we chose to use a knockout mouse dataset [20, 21]. We proposed that a successful algorithm would recover, for each of the 106 knockout genes, a bicluster containing the roughly 20 samples where the gene was knocked out and enriched for genes that share a pathway with the knocked-out gene. Thus this dataset allows us to have some sense of its true bicluster structure.

2.4.1 IMPC dataset

We use the RNA-seq dataset available on ArrayExpress under accession number E-MTAB-5131, part of the International Mouse Phenotyping Consortium (IMPC) [20, 21]. It consists of 106 knockout genotypes, from each of which are available roughly 3 replicates in each of up to 7 tissues. There are also samples from wild-type mice.

To make the study feasible for multiple algorithms in terms of computational time, we chose to restrict to a subset of genes. We restricted to the 4444 genes which share a Reactome pathway with at least one of the 106 knockout genes, found by searching the Reactome pathways [22, 23] using Mouse Mine [24]. We apply three different normalisation methods to the data: (1) library size adjustment using DESeq's median of ratios normalisation method, (2) the log transform $x \rightarrow \log(x+1)$, which is commonly used in analysis of gene expression data and (3) Gaussian quantile normalisation so that each gene has approximate $N(0,1)$ distribution.

2.4.2 Tensor structure

The *Tensor* algorithms require the dataset to have three dimensions i.e. m individuals, t tissues, p genes rather than just $n = m \times t$ samples and p genes. We chose the 3 tissues with the most samples (liver, lung and cardiac ventricle), and the 64 genotypes with at least one sample in each of these tissues. Unfortunately we were unable to find information detailing which samples came from which specific mouse replicate so couldn't simply include a row for each individual, a column for each gene and a layer for each tissue. Instead we pooled the samples from each genotype for each tissue individually by taking the mean of the replicates. Thus we had $m = 64, t = 3$ with a total of $n = 192$ samples.

This type of dataset, which we call the *tensor* dataset, can be used by all the algorithms, whereas the *non-tensor* dataset, which simply uses all $n = 1143$ samples, can't be used by the *Tensor* algorithms.

2.5 Evaluation metrics

We use a range of metrics to evaluate performance of the biclustering algorithms (Table 3), which are described fully in Section S3. In particular, we made use of an extensive study of biclustering accuracy metrics [8] to choose the *clustering error* (CE) metric [25, 8] to evaluate biclustering accuracy. This was shown to satisfy all but one of the desirable properties defined by Horta and Campello, which is a great improvement on the consensus score and recovery and relevance scores commonly used to evaluate biclustering similarity.

Most metrics used by previous reviews, including the *clustering error* metric that we intend to use for evaluation of performance on simulated datasets, cannot be used on real datasets, as they require knowledge of the entire biclustering structure of the dataset. We introduce two metrics that can be used even when nothing is known about the structure of the dataset: Normalised Reconstruction Error (NRE) and Mean Biclustering Redundancy (MBR).

3 Results

3.1 Post-processing

After looking at the raw output, we decided that we would first need to apply some post-processing steps in order to allow meaningful comparison of the algorithms. The *Tensor* algorithms, *NMF* algorithms and FABIA returned many biclusters containing all genes and all samples (Figure S11 and Figure S12). We found that removing elements in the matrices below a certain threshold, a process we call *thresholding*, helped to reveal the biclusters within the noisy raw output. The exact process is described in Section S4. Without thresholding, the biclusters returned by FABIA, SDA and the *NMF* algorithms were highly redundant but this redundancy was reduced by thresholding with a threshold of 0.01 (Figure 2). The optimal threshold is similar for most algorithms, both on simulated datasets (Figure S13) and real datasets (Figure S14), and is largely independent of the metric used to select the threshold. In particular, we note that we could have chosen a suitable threshold using only measures available for real datasets, such as Mean Bicluster Redundancy (Figure 2) and Normalised Reconstruction Error (Figure S15).

It is worth highlighting that Plaid and the *Adaptive* algorithms did not require this post-processing step, but in the interests of avoiding bias and unnecessary complications in the analysis we apply the same post-processing steps for every algorithm. The one exception is Plaid, whose implementation returns only the binary membership variables so thresholding cannot be

Table 3: Metrics used to evaluate the biclustering algorithms, which are described more fully in Section S3. For each metric we note what information about the dataset it requires, the best and worst scores theoretically possible and briefly describe what it measures.

Name	**Requires**	**Best**	**Worst**	**Measures**
Clustering Error (CE)	Entire bicluster structure	1	0	Biclustering accuracy
Normalised Reconstruction Error (NRE)	-	0	1	Reconstruction error
Mean Biclustering Redundancy (MBR)	-	0	1	Similarity between biclusters within a run
Similarity between runs	Multiple runs	1	0	Similarity between runs
Sample clustering	Sample clusters	1	0	Sample clustering
Pathway enrichment	Pathway database	1	0	Gene clustering
Relevant pathway enrichment	Known link between pathways and sample clusters	1	0	Biclustering

applied. The fact that these algorithms perform well without need for post-processing is a key advantage in terms of ease of use.

3.2 Choice of K_{init}

Overall, algorithms were poor at accurately recovering the right number of biclusters (Figure S19), with only FABIA and BicMix-Q showing any positive correlation between the true K and recovered K (BicMix-Q had correlation of 0.836 between true K and recovered K). Ideally we would simply use a large value of K_{init} for all algorithms, as this is what we would do in practice on a real dataset with unknown structure. However, only Plaid and the *Adaptive* algorithms have shown that they would effectively learn the number of biclusters to include. The remaining algorithms consistently returned the same number of biclusters as they started with, so didn't 'learn' K at all (Figure 3). The *Adaptive* algorithms achieve better performance when started with an overestimate of the number of biclusters (Figure S20). Thus, we use $K_{init} = K$, the true number of biclusters for all algorithms, except for the *Adaptive* algorithms for which we use a slight overestimate of K_{init} ($K_{init} = K + 10$, except for when $K = 20$, when we use $K_{init} = 25$.). Note that our way of choosing K_{init} is dependent on knowing the true number of biclusters, so it gives the algorithms an advantage they would not have on real datasets. However, it allows us to compare the 'ideal' behaviour of each algorithm. For the real datasets we use $K_{init} = 50, 200$ for all algorithms.

3.3 Results on simulated datasets

The results are summarised in Table 4. Figure 4 shows the biclustering accuracy of the algorithms across all the simulated datasets. The *Adaptive* algorithms performed best, with SSLB having the best overall accuracy on simulated datasets (0.336). SNMF has the best accuracy of the non-Bayesian algorithms (0.239).

The accuracy of the algorithms generally decreases as the size of the dataset increases (Figure S21), and as the number of biclusters increases (Figure S22). We had expected algorithms to perform better when there were fewer biclusters, which is the case for SSLB and the *NMF* algorithms. However, FABIA, BicMix and BicMix-Q have poor accuracy on the datasets with small number of biclusters. For the very largest datasets (*large-K100* and *large-K400*) many algorithms took a long time to run and only MultiCluster and nsNMF completed runs within 12 hours when using K_{init} close to 400 (Table S5 shows failure counts across all runs and Table S3 shows failure counts restricted to the value of K_{init} chosen for analysis).

Changing the sparsity of the biclusters in the simulated datasets had a large effect on accuracy. We had expected that on the datasets with only very sparse biclusters, the *Adaptive* algorithms would have best accuracy as they have the strongest sparsity constraints but in fact the *NMF* algorithms performed best on these datasets (Figure S23). We looked at how the recovery of true biclusters was affected by the sparsity of the bicluster and found that most algorithms achieved better recovery scores for denser biclusters (Figure 5).

The algorithms were in general, however, fairly robust to noise, with little difference in performance between datasets using Negative Binomial noise, Gaussian noise and no noise and only Plaid showing significant decrease in accuracy as noise was increased (Figure S24).

3.4 Results on real datasets

The algorithms performed well at finding biclusters corresponding to tissues, with many achieving near perfect performance (Table 4, Figure S25). The algorithms were less effective at finding biclusters corresponding to genotypes, with FABIA and SSLB the top two algorithms (Figure S26). This poor clustering of samples from the same genotype might be due to the fact that the algorithms did not return many biclusters containing samples from multiple tissue types (Figure S27), suggesting that between-tissue differences are dominating over between-genotype differences.

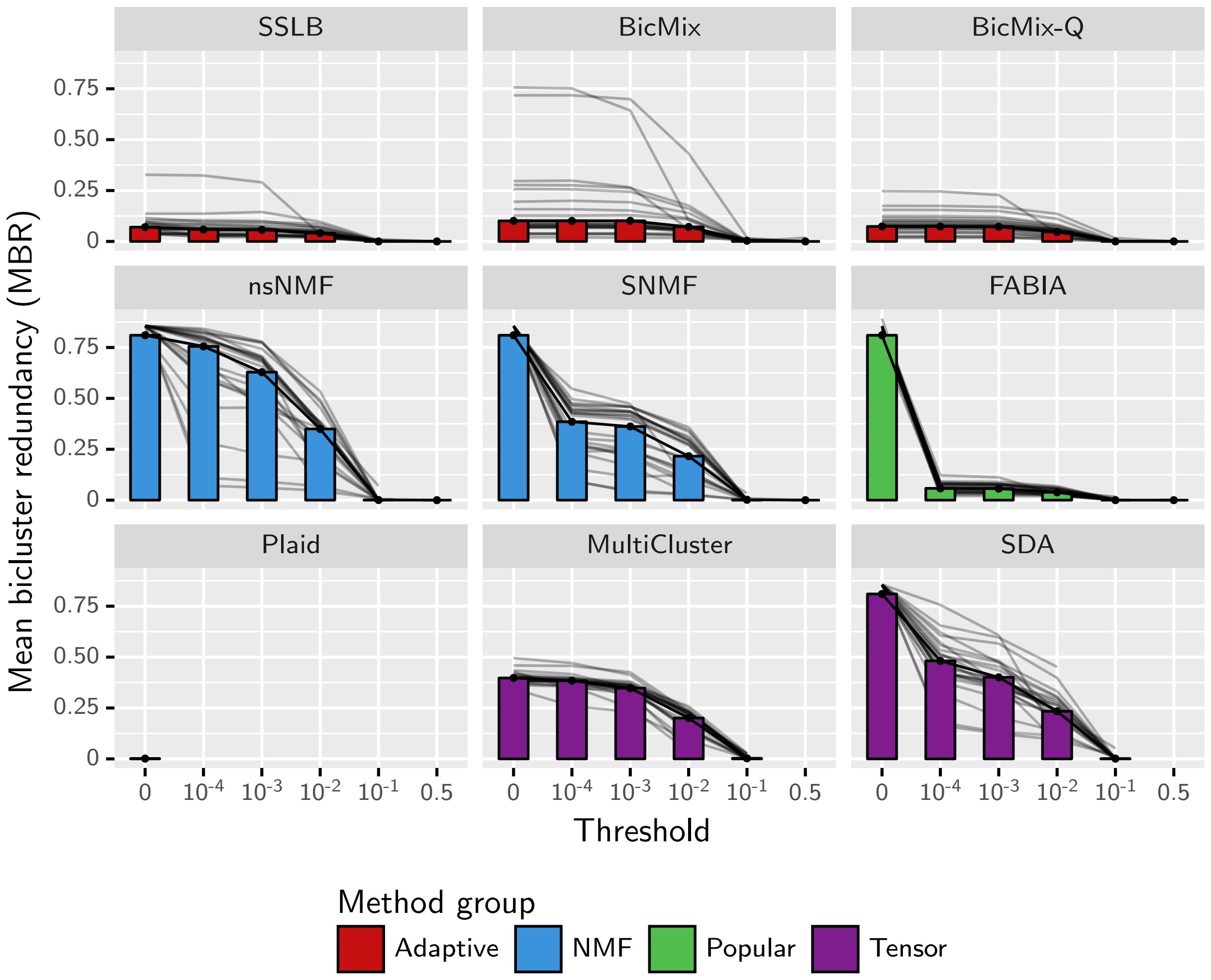

Figure 2: Mean bicluster redundancy (MBR) within a run plotted against the threshold for inclusion in bicluster, for simulated datasets. MBR (Section S3.3) is in the range $[0, 1]$, and a lower value is preferred as it suggests that the algorithm is not returning biclusters which are very similar to each other. The thresholding process is described in Section S4. The median of this measure across all simulated datasets and all values of K_{init} is shown by the bars. The grey lines show the median for each dataset type. Note that without thresholding (threshold 0) the biclusters within each run by FABIA, SDA and nsNMF are almost all identical.

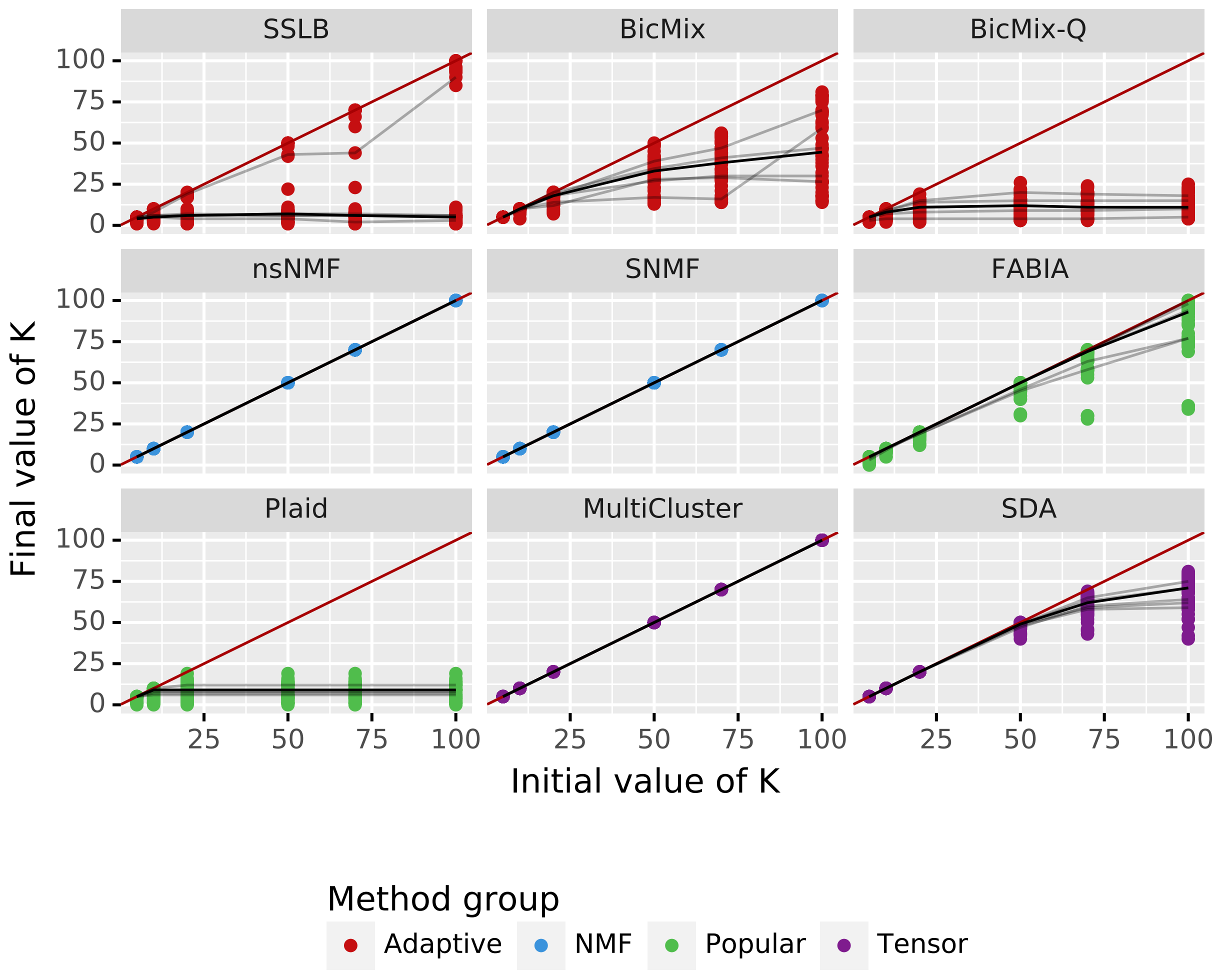

Figure 3: Robustness to choice of K_{init}, shown by final value of K (after post-processing) plotted against initial value of K. There is a point for each of the 3 seeds for each dataset of type *K5*, *K10*, *base*, *K50* and *K70*. The red line shows the limit, where the final value of K is the same as the initial value of K. The black line joins the median $\hat{K}$ across all datasets, and the gray lines join the medians for each individual dataset. The ideal behaviour is for the line to be flat once K_{init} exceeds the true K, showing that the algorithm converges to the same value of K regardless of the initial value given, as long as K_{init} is sufficiently large. True K varies from $K = 5$ to $K = 70$. MultiCluster and *NMF* algorithms always returned the same number of factors as they started with.

Table 4: Summary of results, with the best score for each measure underlined and in bold and scores close to the best score underlined and in italics. Unless otherwise stated, the measures are given for all runs using K_{init} as described in Section 3.2 and after the standard thresholding has been applied. Runs that failed (Table S3 and Table S4) are discarded in the analysis. *Tensor* algorithms could not be run on non-tensor datasets so these entries are marked 'N/A'. (A) Clustering Error (CE) across all simulated datasets. (B) Reconstruction error (NRE) across all simulated datasets. Algorithms which did not return the raw values required to calculate this measure are marked with '*'. (C-D) Average similarity between recovered biclusters (MBR) before (*raw*) and after (*thresholded*) the standard thresholding has been applied. (E) Correlation between K_{init} and CE, best score is 0, indicating that score was unaffected by K_{init}. (F) Correlation between K_{init} and $K_{\text{recovered}}$, best score is 0, indicating that the algorithm converged to the same number of biclusters regardless of K_{init}. (G) Correlation between $K_{\text{recovered}}$ and K_{true} when $K_{init} = 100$, an overestimate of K_{true}. The best score is 1. Entries marked '*' correspond to algorithms that returned $K_{\text{recovered}} = K_{init}$ for all runs, for which correlation could not be calculated. (H) Similarity between all ten runs on each IMPC dataset. (I-L) Performance on IMPC datasets with tensor structure with $K_{init} = 50$, using measures described in Section S3. (M-P) Same as H-K but for IMPC datasets with non-tensor structure. Note that the *Tensor* algorithms were not run on the non-tensor datasets. (Q-R) Time to run in seconds on the datasets of type *K5*, *K10*, *base*, *K50* and *K70*, with $K_{init} = 20$ and $K_{init} = 100$ respectively. (S) Time to run in seconds for the largest simulated dataset *large-K400*, when the algorithms were run with a small number of biclusters ($K_{init} = 20$ for all algorithms except SSLB, BicMix, BicMix-Q which used $K_{init} = 25$). Plaid failed to complete any runs on this largest dataset and is marked as '*'. (T-U) for the largest IMPC datasets, where the algorithms used a large number of biclusters ($K_{init} = 200$). Note that the *Tensor* algorithms were not run on the non-tensor datasets.

		Adaptive			***NMF***		***Popular***		***Tensor***	
		SSLB	BicMix	BicMix-Q	nsNMF	SNMF	FABIA	Plaid	MultiCluster	SDA
	Simulated datasets									
(A)	Biclustering accuracy (CE)	**0.366**	*0.3*	0.261	0.165	0.239	0.21	0.145	0.173	0.173
(B)	Reconstruction error (NRE)	**0.0683**	**0.0739**	*	0.137	0.18	*	*	*	*0.0939*
	Ease of use									
(C)	Redundancy (MBR - thresholded)	0.0618	0.134	0.0692	**0.34**	*0.239*	0.0441	0.00321	0.192	*0.259*
(D)	Redundancy (MBR raw)	0.129	0.249	0.124	**0.914**	**0.912**	**0.901**	0.00321	0.439	**0.913**
(E)	Robustness to K_{init} (CE)	**-0.0651**	*-0.269*	**-0.008**	-0.774	-0.748	*-0.16*	**0.0202**	-0.66	-0.392
(F)	Robustness to K_{init} (Recovered K)	*0.321*	0.782	*0.445*	1	1	0.962	**0.201**	1	0.955
(G)	Recovery of K_{true}	-0.484	0.0289	**0.836**	*	*	0.432	0.0483	*	0.0421
(H)	Similarity between runs	0.283	0.342	0.405	0.521	0.606	0.248	0.602	**1**	0.394
	Tensor IMPC datasets									
(I)	Tissue clustering (tensor)	0.822	0.754	0.579	*0.934*	0.752	0.658	0.666	**0.994**	**0.966**
(J)	Genotype clustering (tensor)	*0.167*	0.123	0.129	0.045	0.0406	**0.253**	0.0493	0.0538	0.0507
(K)	Gene clustering (tensor)	0.771	0.68	0.408	**0.997**	0.647	0.834	**0.977**	*0.94*	0.801
(L)	Relevant pathway clustering (tensor)	0.255	0.12	0.0981	**0.53**	0.3	0.242	0.328	0.35	0.24
	Non-tensor IMPC datasets									
(M)	Tissue clustering (non-tensor)	**0.951**	0.708	0.521	0.671	*0.871*	0.787	0.641	N/A	N/A
(N)	Genotype clustering (non-tensor)	*0.088*	0.0648	0.0521	0.0355	0.0483	**0.122**	0.0406	N/A	N/A
(O)	Gene clustering (non-tensor)	*0.956*	0.762	0.691	**1**	0.915	*0.969*	**1**	N/A	N/A
(P)	Relevant pathway clustering (non-tensor)	0.36	0.221	0.177	**0.483**	*0.445*	0.325	*0.433*	N/A	N/A
	Time									
(Q)	Time ($K_{\text{init}} = 20$)	10.1	52.1	12.6	**1.49**	11.9	5.41	4.75	14.8	45.9
(R)	Time ($K_{\text{init}} = 100$)	101	666.7	*22.7*	**1.93**	513.6	39	*7.3*	*17.3*	612.8
(S)	Time (large-K400 dataset)	6801.9	11250.6	29587.7	*263*	**146.4**	1459.4	*	*696.1*	4746.9
(T)	Time (tensor IMPC datasets)	3904.3	354.2	837.9	**6.85**	29107.7	749	*90.7*	*40.7*	3330.3
(U)	Time (non-tensor IMPC datasets)	19579.8	8631.4	10134.4	**25.7**	27194.5	5119.7	*383.1*	N/A	N/A

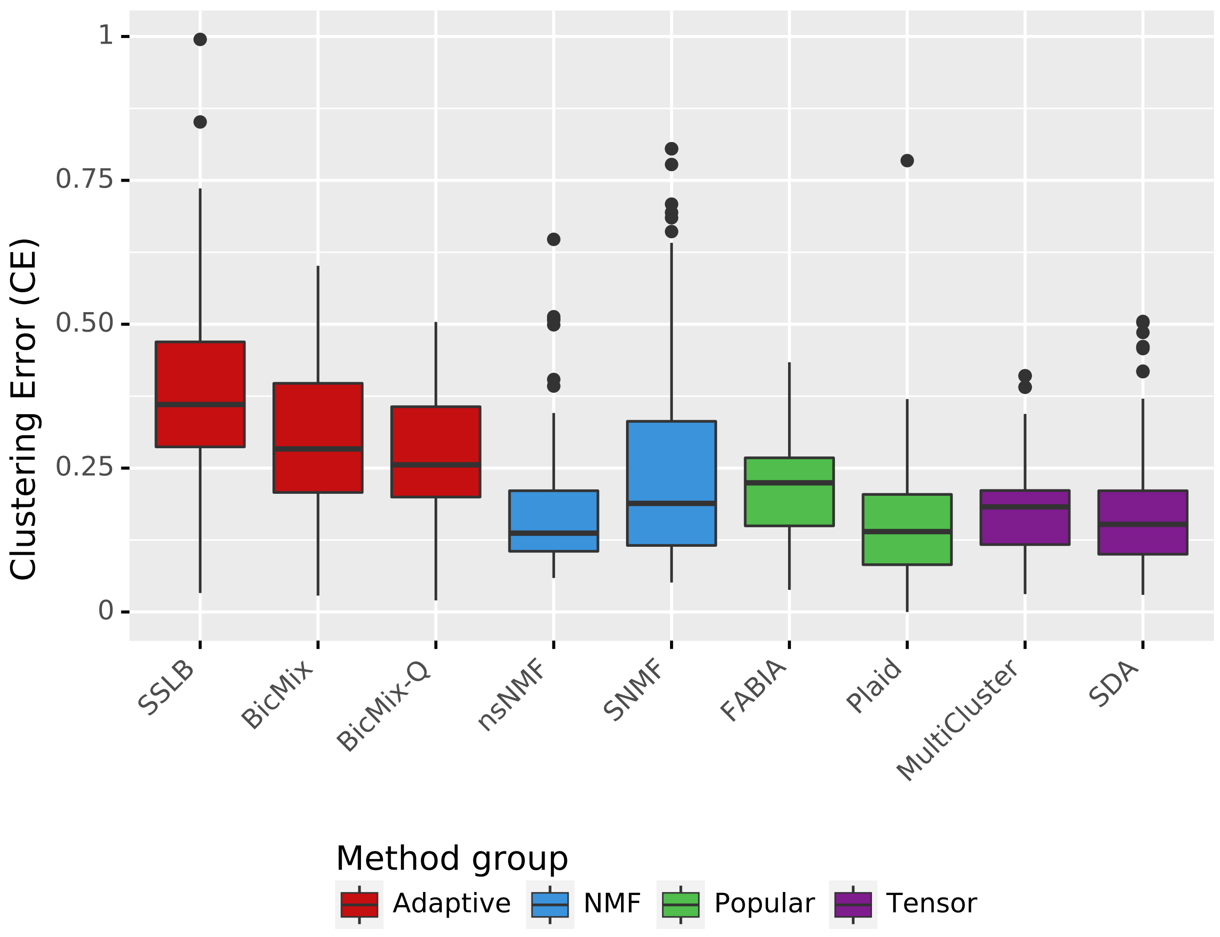

Figure 4: Clustering error (CE) across all simulated datasets. The score is in the range $[0, 1]$ with larger values preferred. Datasets used are described in Table 2. K_{init} is as described in Section 3.2 and standard thresholding has been applied. Runs that failed (Table S3) are discarded in the analysis.

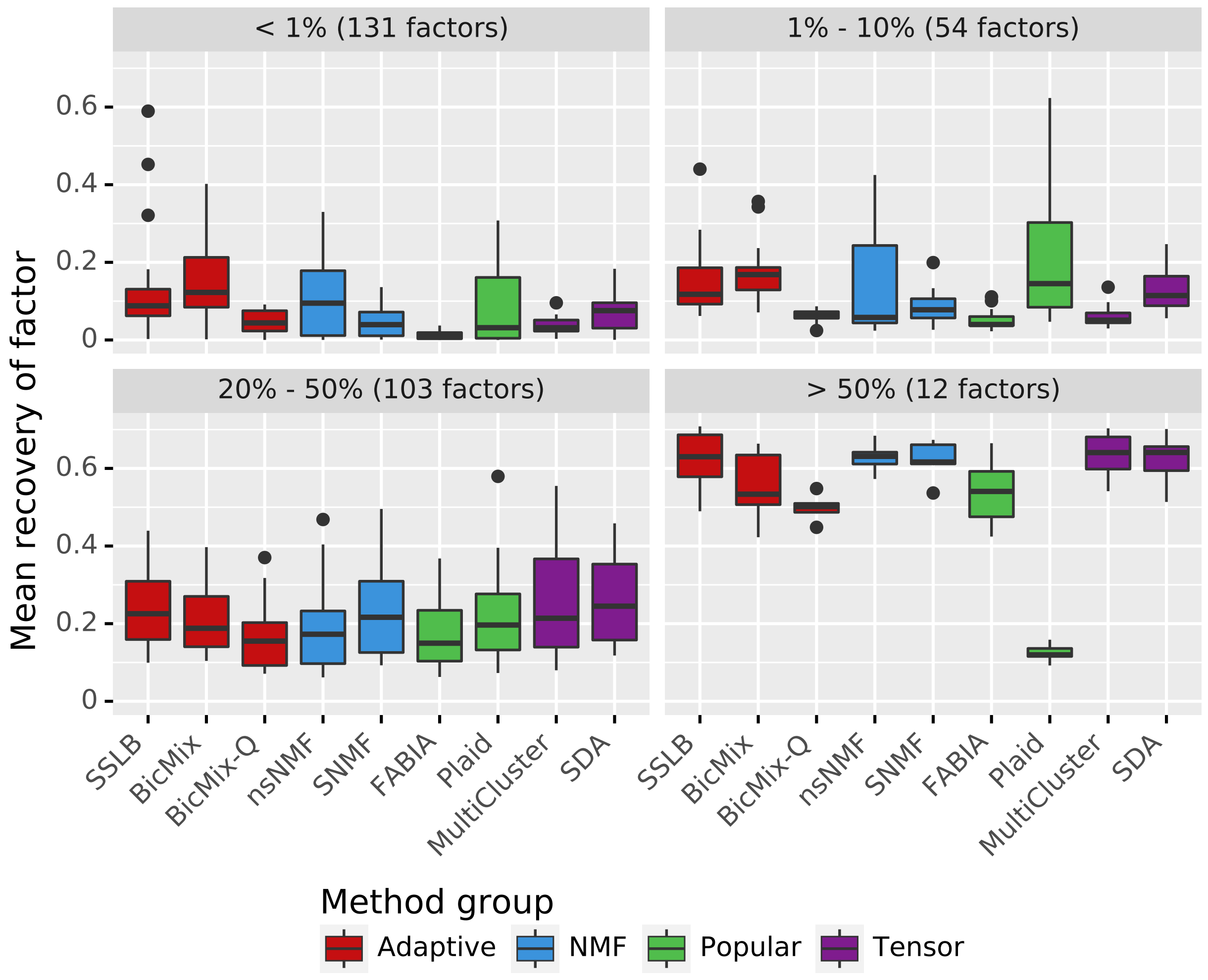

Figure 5: Mean recovery of true biclusters, grouped by size of true biclusters (fraction of total matrix area taken up by true bicluster). We restrict to the datasets *base*, *sparse*, *dense*, *sparse-square*, *dense-square*. For each true bicluster in these datasets and for each algorithm we find the recovered bicluster achieving maximum Jaccard index with the true bicluster. We call this the recovery score for that true bicluster and that algorithm, which is a measure of how well the algorithm has recovered a particular true bicluster. This plot shows the spread of recovery scores for each algorithm, grouped by the proportion of the total area of the dataset taken up by the true bicluster. Recovery scores are generally better for denser biclusters, though Plaid has notably lower recovery scores for the densest biclusters compared to other algorithms.

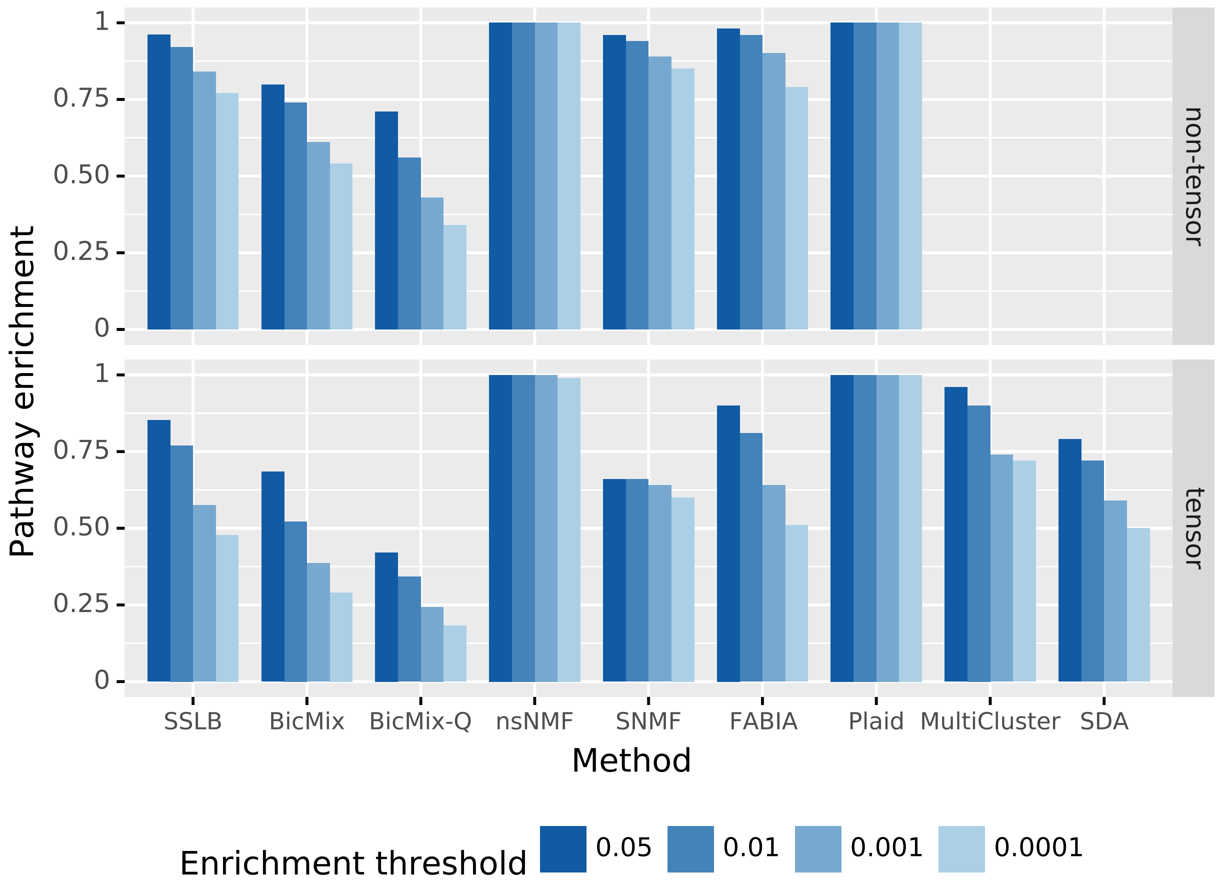

Figure 6: Gene clustering ability, measured by the mean proportion of recovered biclusters which are enriched for at least one pathway. Enrichment is measured using the one-tailed hypergeometric test adjusted for multiple testing using the Benjamini-Yekutieli correction, using a range of thresholds. The median of this measure is shown for each algorithm, split into tensor and non-tensor datasets. Runs that failed (Table S4) are discarded in the analysis.

Many algorithms also achieved good clustering of genes, as measured by enrichment of biclusters for Reactome pathways (Figure 6 and Figure S28), with FABIA, SSLB, nsNMF, MultiCluster and Plaid achieving near perfect scores on multiple versions of the dataset. However, this performance should be considered alongside the fact that Plaid returned on average only 4 biclusters and that nsNMF returned factors with high similarity to each other (Figure S14) and thus many of nsNMF's factors may be enriched for the same small set of Reactome pathways. For example, in one run 145 of the 200 factors recovered by nsNMF were enriched for the 'Metabolism' pathway ($q < 0.05$) compared to only 39 of the 188 factors recovered by an SSLB run on the same version of the IMPC dataset.

The unifying test on IMPC data is the biclustering ability, measured as the proportion of knockout genotypes for which the bicluster best recovering the samples is enriched for at least one pathway containing the knocked-out gene (Figure 7). To achieve a high score, an algorithm needs to (1) cluster samples well by genotype, (2) cluster genes well by pathway and (3) return biclusters where there is a link between the samples selected and the genes selected. The *NMF* algorithms, Plaid and SSLB did best according to this metric, achieving enrichment of relevant pathways for approximately 45% of the knockout genotypes in multiple versions of the IMPC dataset. Most algorithms had worse performance when using $K_{init} = 200$ (Figure S29), particularly SNMF which failed on 34 of the 40 runs using $K_{init} = 200$ (Table S6). Plaid was the only method to fail on a higher percentage of all runs (54 out of 120) but had similar failure rates when using $K_{init} = 50$ and $K_{init} = 200$.

Strikingly, the reconstruction error (NRE) on the IMPC datasets is much worse (higher) than on the simulated datasets (Figure S30), except for nsNMF. This demonstrates the additional complexity in the real datasets compared to the simulated datasets.

3.5 Robustness

With the exception of MultiCluster, the algorithms compared here are stochastic and thus may produce different results each time they are run. If a similar set of biclusters is recovered by repeated runs of an algorithm, this can give confidence that the bicluster decomposition is meaningful. For each algorithm in turn we considered pairs of runs on the same dataset which used the same value of K_{init} and calculated the similarity between each pair using CE (Clustering Error). As MultiCluster is deterministic, it achieves a perfect score of 1 in this test. Of the remaining algorithms, Plaid and the *NMF* algorithms are the only ones to have a median similarity score between runs of over 0.5 (Figure 8).

3.6 Computational time

It is important to evalute the time taken for the algorithms to run, and how this scales with the size and complexity of the dataset, as this can restrict the datasets that an algorithm is able to process. The slowest algorithm on the IMPC datasets was SNMF, which took 8 hours to run on the tensor log-transformed dataset, compared to the 7 seconds taken by nsNMF (Table 4). Figure S31 shows runtime with small value of K_{init}, Figure S32 and Figure S33 show, for simulated datasets and IMPC datasets respectively, that runtime for the *Adaptive* algorithms, SDA, SNMF and FABIA changed drastically with K_{init}.

4 Discussion

On simulated datasets *Adaptive* algorithms had the best overall performance. We investigated the limitations of biclustering algorithms by varying dataset complexity. All algorithms were relatively unaffected by increasing noise in simulated datasets, but performance decreased when dataset size and number of biclusters were increased. Dense biclusters were generally recovered better than sparse biclusters. From a biological perspective, we expect the dense biclusters to correspond to confounding variables such as sex and age and the sparse biclusters to be more biologically interesting, so this behaviour is not ideal.

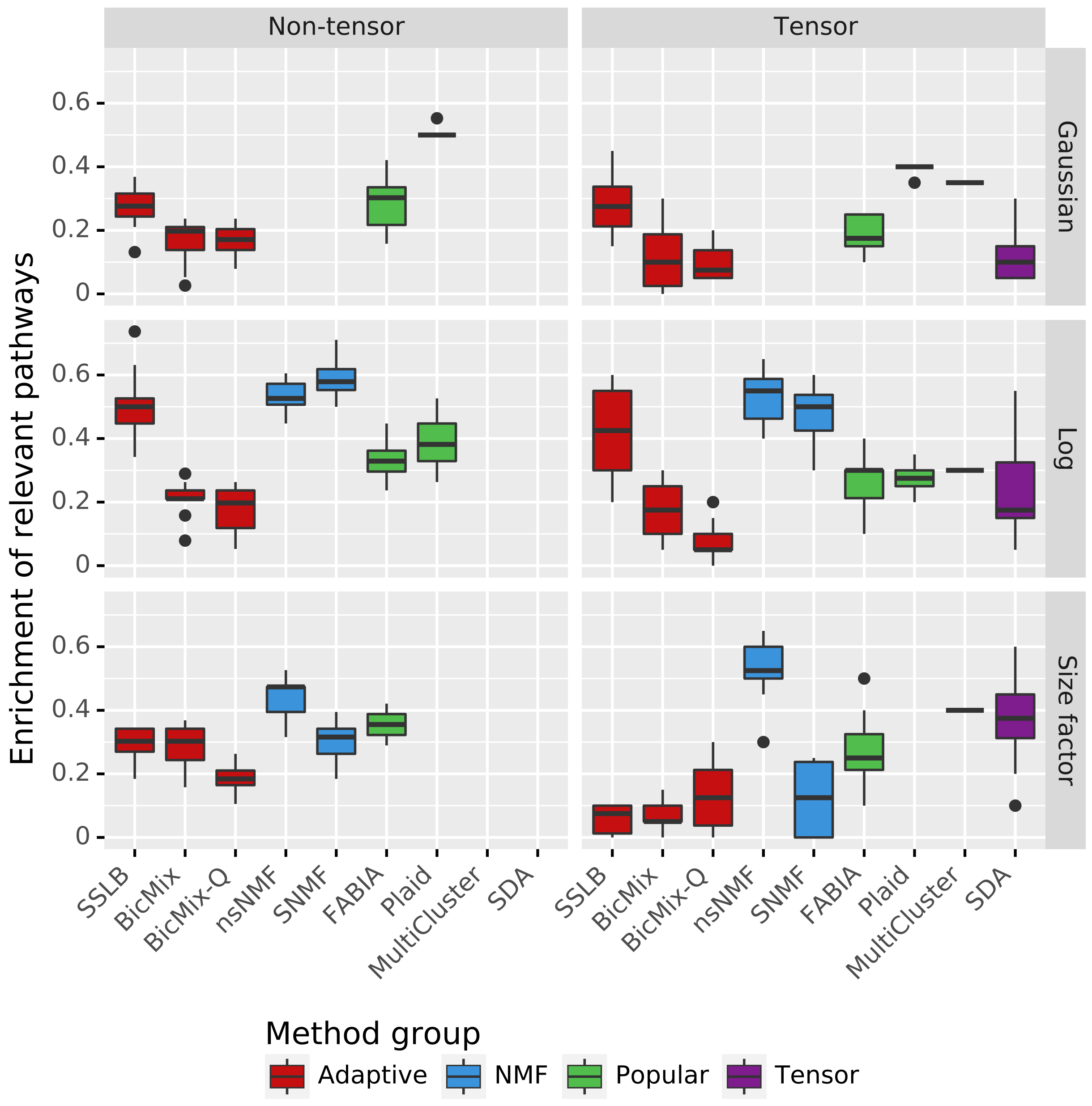

Figure 7: Biclustering ability on IMPC datasets, measured by the mean proportion of knocked-out genes for which the bicluster best matching the samples where the gene was knocked out is enriched for at least one pathway containing the knocked-out gene. Enrichment is measured using the one-tailed hypergeometric test adjusted for multiple testing using the Benjamini-Yekutieli correction, using a threshold for significance of 0.05. Standard thresholding is applied and $K_{init} = 50$. Results for $K_{init} = 200$ are in S29. Note that *Tensor* algorithms couldn't be run on the non-tensor datasets, *NMF* algorithms couldn't run on datasets which used quantile normalisation and Plaid failed to run on the dataset which used DESeq's size factor normalisation (Table S4).

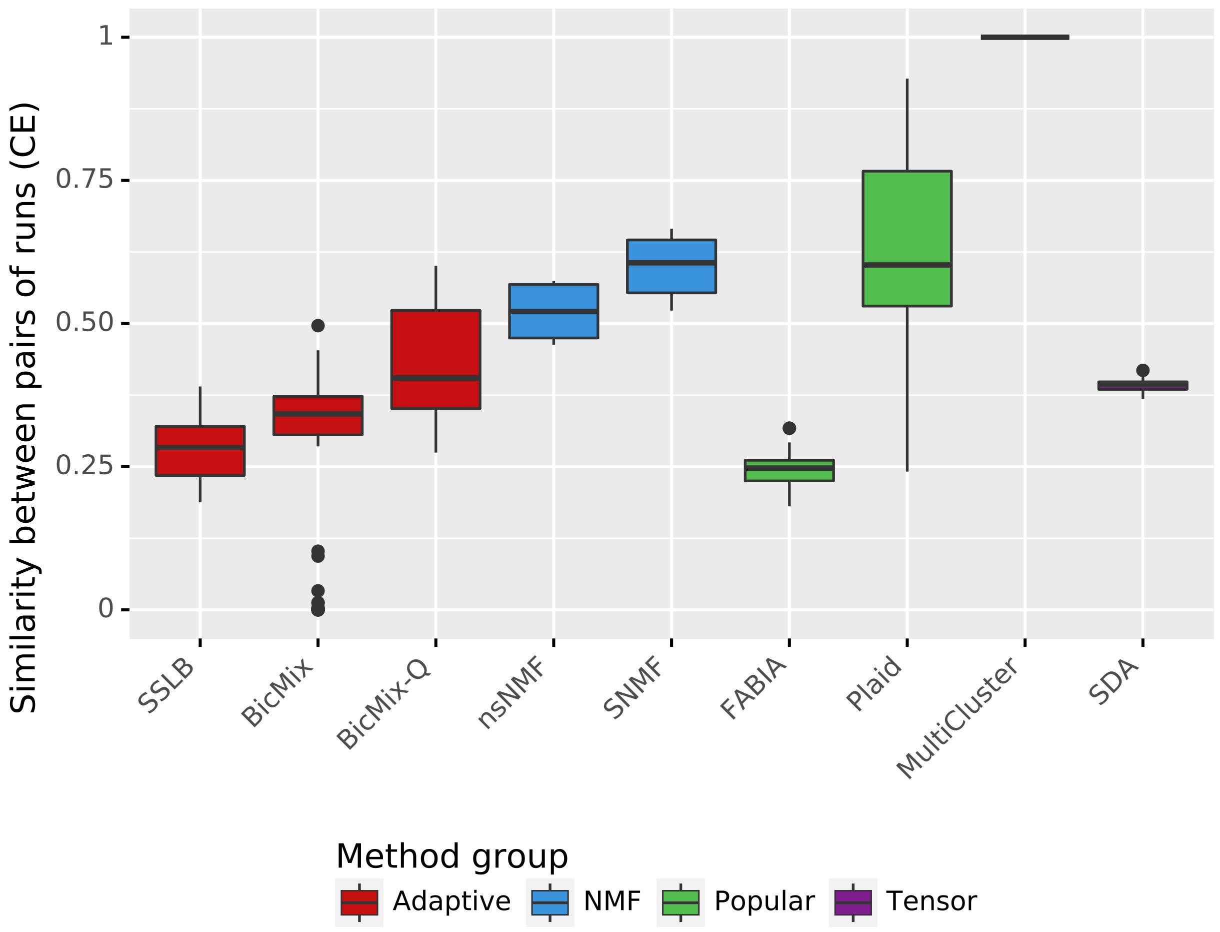

Figure 8: Robustness of biclusters recovered by the algorithms. For each IMPC dataset we calculate similarity scores using clustering error between each pair of runs from a given algorithm, restricted to runs which used the same K_{init}. This boxplot shows the spread of similarity scores achieved by each algorithm. Higher scores are preferred, as they indicate that the algorithm recovers similar biclusters on each re-run. MultiCluster is deterministic, so achieves maximal score of 1. Plaid and the *NMF* algorithms all achieve relatively high scores.

Like previous studies, we found that algorithms achieved good enrichment of biclusters for gene pathways in real datasets. However, all algorithms struggled to cluster samples from different tissues, highlighting the difficulty of borrowing information across tissue types. We carefully chose a knockout mouse dataset to allow evaluation of biclustering on real datasets, a task which has eluded previous studies, and found that *NMF* algorithms, SSLB and Plaid performed best at recovering biclusters.

In terms of ease of use, *Adaptive* algorithms and Plaid are the only algorithms well suited to use without tuning of K_{init}, and also didn't require post-processing. *NMF* algorithms and Plaid had the most robust results, with different runs having on average a similarity of 0.5, as measured by clustering error. Plaid and nsNMF were the fastest, with nsNMF running on the largest IMPC dataset in 7 seconds, compared to the 8 hours taken by the slowest method (SNMF). We found that most algorithms performed well with their default parameters, with few parameter values that showed consistent and significant improvement over the default values during our parameter sweep.

Overall, many algorithms performed better than the *Popular* algorithms which had performed best in previous reviews, showing the need for continued comparison studies as biclustering algorithms develop further. *NMF* algorithms had poor raw output but nsNMF was one of the top-ranking methods after using the sparsity-inducing thresholding procedure we introduce. *Tensor* algorithms did not perform better than other algorithm types, despite both real and simulated datasets having tensor structure. *Adaptive* algorithms performed particularly well on the simulated datasets, and SSLB also had good performance on the real datasets.

Key points

- We introduce a promising thresholding procedure to enhance sparsity of the returned biclusters, essential for FABIA, SDA, and *NMF* algorithms which otherwise returned only biclusters containing every gene and every sample.
- We introduce the MBR metric for redundancy within a run, and NRE metric for measuring reconstruction error, which can be used even when the true structure of the dataset is not known.
- We have shown the potential for re-purposing of *NMF* algorithms to the task of biclustering. The nsNMF algorithm was orders of magnitude faster than the more complex algorithms, and had good performance, particularly on the real datasets.
- For datasets with unknown structure we recommend SSLB. If a fast algorithm is needed and the number of biclusters is known, or if metrics are available to aid the choice of K_{init}, then we recommend nsNMF.
- Normalisation method used for real datasets had a large impact on the performance of algorithms. The algorithms performed best on log-transformed data.

Funding

CW is supported by the Wellcome Trust (WT107881) and the Medical Research Council (MC UU 00002/4). KN is supported by the Wellcome Trust (220024/Z/19/Z). For the purpose of open access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, November 2004. ISSN 2326-3865. doi: 10.1109/TKDE.2004.68.

[2] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286(5439):531–537, October 1999. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.286.5439.531.

[3] Wouter Saelens, Robrecht Cannoodt, and Yvan Saeys. A comprehensive evaluation of module detection methods for gene expression data. *Nature Communications*, 9(1):1–12, March 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-03424-4.

[4] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, May 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl060.

[5] Doruk Bozdağ, Ashwin S. Kumar, and Umit V. Catalyurek. Comparative Analysis of Biclustering Algorithms. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, BCB '10, pages 265–274, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0438-2. doi: 10.1145/1854776.1854814.

[6] Victor A. Padilha and Ricardo J. G. B. Campello. A systematic comparative evaluation of biclustering techniques. *BMC Bioinformatics*, 18(1):1–25, December 2017. ISSN 1471-2105. doi: 10.1186/s12859-017-1487-1.

[7] K. Eren, M. Deveci, O. Kucuktunc, and U. V. Catalyurek. A comparative analysis of biclustering algorithms for gene expression data. *Briefings in Bioinformatics*, 14(3):279–292, May 2013. ISSN 1467-5463, 1477-4054. doi: 10.1093/bib/bbs032.

[8] Danilo Horta and Ricardo J.G.B. Campello. Similarity Measures for Comparing Biclusterings. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(5): 942–954, September 2014. ISSN 1545-5963. doi: 10.1109/TCBB.2014.2325016.

[9] Sepp Hochreiter, Ulrich Bodenhofer, Martin Heusel, Andreas Mayr, Andreas Mitterecker, Adetayo Kasim, Tatsiana Khamiakova, Suzy Van Sanden, Dan Lin, Willem Talloen, Luc Bijnens, Hinrich W. H. Göhlmann, Ziv Shkedy, and Djork-Arné Clevert. FABIA: Factor analysis for bicluster acquisition. *Bioinformatics*, 26(12):1520–1527, June 2010. ISSN 1460-2059, 1367-4803. doi: 10.1093/bioinformatics/btq227.

[10] Laura Lazzeroni and Art Owen. PLAID MODELS FOR GENE EXPRESSION DATA. *Statistica Sinica*, 12(1):61–86, 2002. ISSN 1017-0405.

[11] Heather Turner, Trevor Bailey, and Wojtek Krzanowski. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational Statistics & Data Analysis*, 48(2):235–254, February 2005. ISSN 0167-9473. doi: 10.1016/j.csda.2004.02.003.

[12] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmann, and R.D. Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsNMF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):403–415, March 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.60.

[13] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23 (12):1495–1502, June 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm134.

[14] Miaoyan Wang, Jonathan Fischer, and Yun S. Song. Three-way clustering of multi-tissue multi-individual gene expression data using semi-nonnegative tensor decomposition. *The*

Annals of Applied Statistics, 13(2):1103–1127, June 2019. ISSN 1932-6157. doi: 10.1214/18-AOAS1228.

[15] Victoria Hore, Ana Viñuela, Alfonso Buil, Julian Knight, Mark I McCarthy, Kerrin Small, and Jonathan Marchini. Tensor decomposition for multi-tissue gene expression experiments. *Nature genetics*, 48(9):1094–1100, September 2016. ISSN 1061-4036. doi: 10.1038/ng.3624.

[16] Chuan Gao, Ian C. McDowell, Shiwen Zhao, Christopher D. Brown, and Barbara E. Engelhardt. Context Specific and Differential Gene Co-expression Networks via Bayesian Biclustering. *PLOS Computational Biology*, 12(7):e1004791, July 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004791.

[17] Chuan Gao, Shiwen Zhao, Ian C. McDowell, Christopher D. Brown, and Barbara E. Engelhardt. Differential gene co-expression networks via Bayesian biclustering models. *arXiv:1411.1997 [q-bio, stat]*, November 2014.

[18] Gemma E Moran, Veronika Ročková, and Edward I George. Spike-and-Slab Lasso Biclustering. page 43.

[19] Quan Gu and Kirill Veselkov. Bi-clustering of metabolic data using matrix factorization tools. *Methods*, 151:12–20, December 2018. ISSN 1046-2023. doi: 10.1016/j.ymeth.2018.02.004.

[20] David B. West, Eric K. Engelhard, Michael Adkisson, A. J. Nava, Julia V. Kirov, Andreanna Cipollone, Brandon Willis, Jared Rapp, Pieter J. de Jong, and Kent C. Lloyd. Transcriptome Analysis of Targeted Mouse Mutations Reveals the Topography of Local Changes in Gene Expression. *PLOS Genetics*, 12(2):e1005691, February 2016. ISSN 1553-7404. doi: 10.1371/journal.pgen.1005691.

[21] Gautier Koscielny, Gagarine Yaikhom, Vivek Iyer, Terrence F. Meehan, Hugh Morgan, Julian Atienza-Herrero, Andrew Blake, Chao-Kung Chen, Richard Easty, Armida Di Fenza, Tanja Fiegel, Mark Grifiths, Alan Horne, Natasha A. Karp, Natalja Kurbatova, Jeremy C. Mason, Peter Matthews, Darren J. Oakley, Asfand Qazi, Jack Regnart, Ahmad Retha, Luis A. Santos, Duncan J. Sneddon, Jonathan Warren, Henrik Westerberg, Robert J. Wilson, David G. Melvin, Damian Smedley, Steve D. M. Brown, Paul Flicek, William C. Skarnes, Ann-Marie Mallon, and Helen Parkinson. The International Mouse Phenotyping Consortium Web Portal, a unified point of access for knockout mice and related phenotyping data. *Nucleic Acids Research*, 42(Database issue):D802–D809, January 2014. ISSN 0305-1048. doi: 10.1093/nar/gkt977.

[22] Antonio Fabregat, Konstantinos Sidiropoulos, Guilherme Viteri, Oscar Forner, Pablo Marin-Garcia, Vicente Arnau, Peter D'Eustachio, Lincoln Stein, and Henning Hermjakob. Reactome pathway analysis: A high-performance in-memory approach. *BMC bioinformatics*, 18(1):142, March 2017. ISSN 1471-2105. doi: 10.1186/s12859-017-1559-2.

[23] Bijay Jassal, Lisa Matthews, Guilherme Viteri, Chuqiao Gong, Pascual Lorente, Antonio Fabregat, Konstantinos Sidiropoulos, Justin Cook, Marc Gillespie, Robin Haw, Fred Loney, Bruce May, Marija Milacic, Karen Rothfels, Cristoffer Sevilla, Veronica Shamovsky, Solomon Shorser, Thawfeek Varusai, Joel Weiser, Guanming Wu, Lincoln Stein, Henning Hermjakob, and Peter D'Eustachio. The reactome pathway knowledgebase. *Nucleic Acids Research*, 48(D1):D498–D503, August 2020. ISSN 1362-4962. doi: 10.1093/nar/gkz1031.

[24] H. Motenko, S. B. Neuhauser, M. O'Keefe, and J. E. Richardson. MouseMine: A new data warehouse for MGI. *Mammalian Genome: Official Journal of the International Mammalian Genome Society*, 26(7-8):325–330, August 2015. ISSN 1432-1777. doi: 10.1007/s00335-015-9573-z.

[25] Anne Patrikainen and Marina Meila. Comparing Subspace Clusterings. *IEEE Transactions on Knowledge and Data Engineering*, 18(7):902–916, July 2006. ISSN 1041-4347. doi: 10.1109/TKDE.2006.106.