

Supplementary Information for: A loop-counting method for
covariate-corrected low-rank biclustering of gene-expression and
genome-wide association study data

Aaditya V. Rangan Caroline C. McGrouther John Kelsoe Nicholas Schork Eli Stahl
Qian Zhu Arjun Krishnan Vicky Yao Olga Troyanskaya Seda Bilaloglu
Preeti Raghavan Sarah Bergen Anders Jureus Mikael Landen
Bipolar Disorders Working Group of the Psychiatric Genomics Consortium

Contents

| | | |
|-----------|--|------------|
| 1 | Introduction | 4 |
| 1.1 | Background | 4 |
| 1.2 | Additional information for Example-1: Gene expression analysis | 6 |
| 1.3 | Additional information for Example-2: Gene expression analysis | 10 |
| 1.4 | Additional Information for Example-3: Genome-Wide-Association-Study (GWAS) | 17 |
| 2 | Simple case: D only | 20 |
| 3 | Calculating the scores: | 29 |
| 3.1 | Updating the scores using a low-rank update: | 30 |
| 4 | Interpreting the scores: | 30 |
| 5 | Generalization to noisy biclusters | 31 |
| 5.1 | Constructing a low-rank bicluster B : | 35 |
| 5.2 | Generating a sparse low-rank B | 37 |
| 6 | Application to the Planted-Bicluster problem: | 37 |
| 7 | More detailed discussion of the algorithm: | 38 |
| 7.1 | Why binarize? Advantages and disadvantages | 40 |
| 7.2 | Why use loops? Advantages and disadvantages | 44 |
| 7.3 | Why iterate? | 48 |
| 7.4 | Comparison with some other biclustering methods | 48 |
| 8 | Correcting for Controls: D and X | 56 |
| 9 | Correcting for Categorical Covariates: | 59 |
| 9.1 | Interpretation when $I_{\text{cat}} = 2$ and $I_{\text{req}} = 2$ | 62 |
| 10 | Correcting for Continuous Covariates: | 63 |
| 11 | Correcting for Sparsity | 72 |
| 12 | Putting it all together | 75 |
| 13 | Notes regarding computation | 81 |
| 14 | Notes regarding application | 85 |
| 14.1 | Interpreting the output: | 85 |
| 14.2 | Determining significance: | 90 |
| 14.3 | Delineating a bicluster: | 91 |
| 14.4 | Finding secondary biclusters: | 91 |
| 15 | Additional Applications | 96 |
| 15.1 | Accounting for genetic-controls: | 96 |
| 15.2 | Searching for ‘rank-0’ (i.e., differentially-expressed) biclusters: | 98 |
| 15.3 | Searching for triclusters: | 106 |
| 16 | Various properties of multivariate gaussians | 114 |
| 16.1 | one-dimensional gaussian: | 114 |
| 16.2 | convolution of two one-dimensional gaussians: | 116 |
| 16.3 | two-dimensional gaussian: | 116 |
| 16.4 | isotropic two-dimensional gaussian: | 116 |
| 16.5 | restriction of two-dimensional gaussian to quadrants: | 116 |
| 16.6 | approximation of orthonormal \hat{u}, \hat{v} : | 117 |
| 16.7 | convolution of two two-dimensional gaussians: | 117 |
| 17 | Asymptotic formula for $g_{1,\varepsilon,m}$: | 118 |

| | |
|--|------------|
| 18 Low-noise limit for $B^\top B$: | 118 |
| 19 Information-theoretic phase-transitions for rank-1 planted-bicluster problem | 120 |
| 20 Does binarization destroy angular information? | 121 |
| 21 Analysis of loop-scores | 121 |
| 22 Comparison with a simple spectral-biclustering method | 124 |
| 22.1 Size-threshold: | 126 |
| 22.2 Noise-threshold: | 127 |
| 22.3 Behavior near the detection-threshold | 128 |
| 22.4 A remark regarding message-passing algorithms | 131 |
| 22.5 Accounting for controls: | 132 |

1 Introduction

This document contains supplementary information for the manuscript entitled *A loop-counting method for covariate-corrected low-rank biclustering of gene-expression and genome-wide association study data*. This supplementary information describes our method in detail, presenting both analysis and numerical experiments where appropriate. Occasionally we will reference citations within the bibliography of the main text. We will denote these citations with an \mathcal{M} before the citation number.

1.1 Background

Many applications in data-analysis involve ‘low-rank biclustering’; that is, searching through a large data-matrix to find submatrices which have a low numerical-rank (i.e., for which the rows and columns are strongly correlated – see Fig 1). To give an example from genomics, one might imagine a data-matrix involving several genetic-measurements taken across many patients. In this context a ‘bicluster’ would correspond to a subset of genetic-measurements that are correlated across a subset of the patients. While some biclusters might include many genes, or extend across most (or all) of the patients, it is also possible for biclusters to include only a small subset of genes and extend across only a small subset of patients. Detecting biclusters such as these provides a first step towards unraveling the physiological mechanisms underlying the heterogeneity within a patient population. With this picture in mind, a natural question is: given a large data-array, how can one quickly detect any such low-rank biclusters? In this document we address this issue, expanding on a biclustering algorithm originally introduced in [M39].

In its most basic form our algorithm is very simple: Given a large $M \times N$ data-matrix D , we ‘binarize’ the data by sending each entry of D to either $+1$ or -1 , depending on its sign. Then we form row-scores $Z_{\text{ROW}} \in \mathbb{R}^M$ by taking the diagonal entries of DD^TDD^T , and column-scores $Z_{\text{COL}} \in \mathbb{R}^N$ by taking the diagonal entries of D^TDD^TD . We then eliminate the rows and columns of D for which Z_{ROW} and Z_{COL} are small, and repeat the entire process. Eventually, after repeating this process multiple times, we will eliminate almost all the rows and columns of D . If there were indeed a low-rank bicluster hiding within D , then (assuming certain criteria are satisfied) our algorithm will usually find it; retaining the rows and columns of the bicluster until the end.

This algorithm works due to the following observation regarding high dimensional space: a random planar projection of an eccentric gaussian-distribution is typically concentrated in non-adjacent quadrants. This fact implies that 2×2 submatrices of D (referred to as ‘loops’ in the following sections) contain substantial information about biclusters within D ; the rows and columns of any low-rank bicluster will correspond to large values of Z_{ROW} and Z_{COL} , and will thus be retained as the algorithm proceeds. In the sections below we’ll analyze our algorithm a little more carefully, and explain under what conditions it is expected to work.

Our ultimate goal is to provide a working code which can be applied to real problems in data-science. To this end, we introduce several modifications to the original algorithm of [M39]; these modifications allow us to account for many considerations which commonly arise in practice. For example, within the context of genomics, the $M \times N$ data-matrix above may correspond to N different measurements each taken across M patients. Within this paradigm, it is often important to correct for:

1. Cases-versus-Controls: Some patients may suffer from a certain disease (i.e., ‘cases’) while others do not. By correcting for controls we can search for correlation patterns that are limited to the case-population. These case-specific patterns may be useful for clinical diagnosis or for revealing disease mechanisms.
2. Categorical- and continuous-covariates: Often patients come from different studies, or are measured with different machines. Each patient may also be associated with a vector of continuous-covariates (e.g., a vector of mds-components correlated with genetic ancestry). It is often critical to correct for the influence of these covariates when looking for significant patterns.
3. Sparsity: In certain circumstances (e.g., when dealing with genotyped data) the data-matrix can be sparse. Moreover, different columns of the data-matrix can have different sparsity-coefficients (e.g., different minor-allele-frequencies). It is typically important to take this sparsity into account when determining which patterns are significant and which are not.

Before we describe our algorithm in detail we’ll briefly present three examples drawn from genomics. Example-1b and Example-3 below correspond to Example-A and Example-B from the main-text, respectively. Collectively, these examples highlight some of the practical considerations necessary for a biclustering algorithm to be useful.

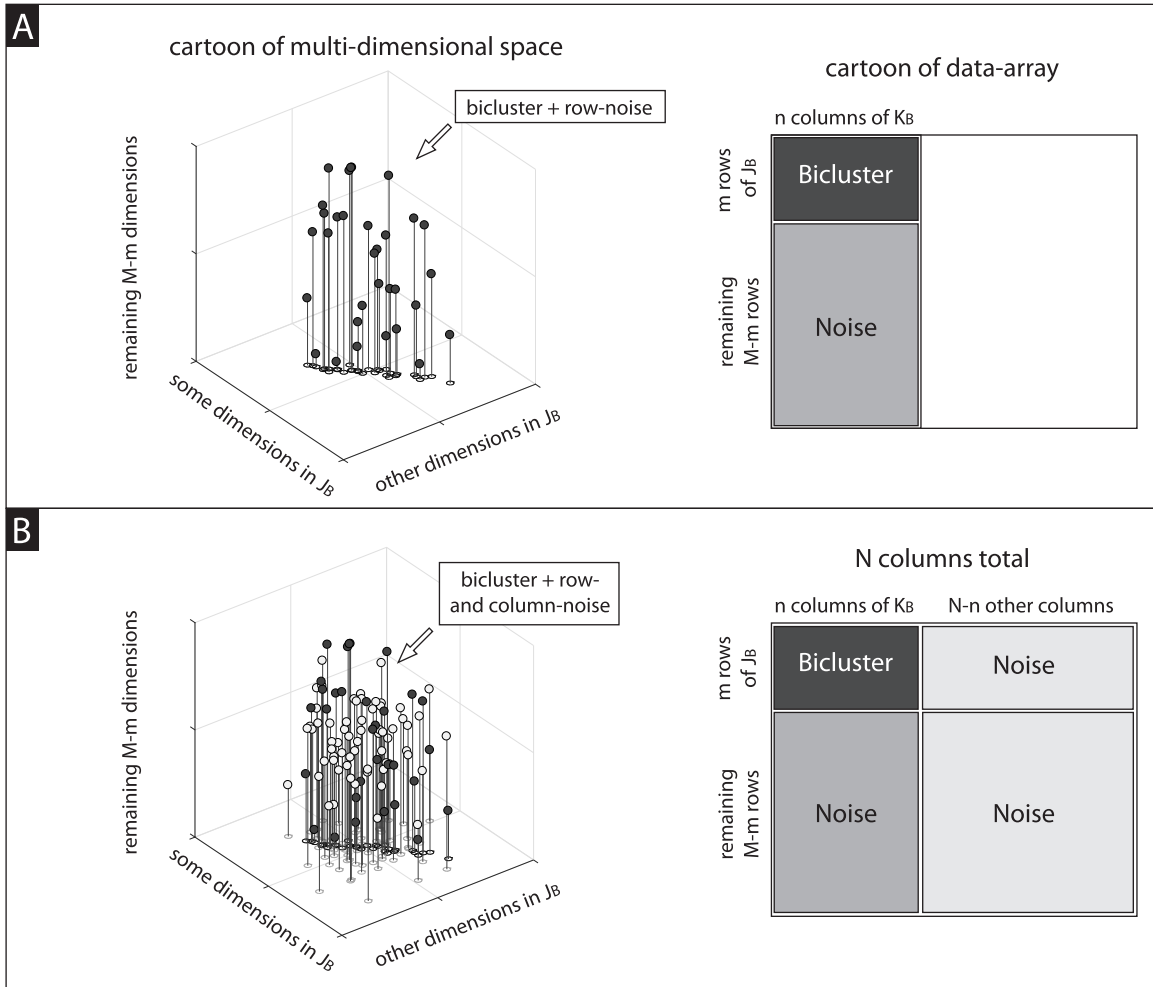


Figure 1: A cartoon illustrating the geometry underlying a ‘low-rank bicluster’. In panel-A we imagine a situation involving an $M \times n$ data-set. This data-set contains m correlated rows (which will later form a bicluster in panel-B), as well as $M - m$ noisy rows. We’ll assume that the m correlated rows are described by the subset J_B . On the left side of panel-A we’ve plotted each of the columns of this data-set as though they were points in \mathbb{R}^M (see dark dots). For this illustration we’ve plotted the m -dimensions of J_B in the xy -plane, and used the z -axis to represent the remaining $M - m$ dimensions. Note that, due to the noisy rows, the data-points will not be confined to a very low-dimensional space. In order to reveal the low-rank structure within the data, it is necessary to first restrict our attention to the m rows of the bicluster. In other words, we need to project the data-points onto the subset of dimensions described by J_B – which is in this case the xy -plane. Only after such a projection will the data lie in a low-dimensional space (note that the shadows of the data-points in the xy -plane align). In panel-B we illustrate an $M \times N$ data-set containing the data shown in panel-A, as well as an additional $N - n$ noisy columns. The correlated $m \times n$ submatrix within this data-set is described by subsets J_B and K_B , and is an example of a low-rank bicluster. On the right side of panel-B we’ve plotted each of the columns of this $M \times N$ data-set as though they were points in \mathbb{R}^M . The points from the columns in K_B are plotted as dark dots (identical to panel-A), whereas the noisy columns drawn from outside K_B are plotted as white dots. After projecting all the dots onto the m -dimensions of J_B (i.e., onto the xy -plane), only the dark shadows from K_B align. The remaining shadows from the noisy columns are scattered about the xy -plane, and do not exhibit any particular correlation. The low-rank bicluster within the data-set has two defining subsets: first, the axes of the projection (i.e., the rows within J_B), and second, the identity of the dark dots whose shadows align (i.e., the columns of K_B). Detecting the bicluster requires the identification of both these subsets. The algorithm presented in this document strives to detect these kinds of biclusters within large data-sets – identifying the subsets J_B and K_B as best it can.

1.2 Additional information for Example-1: Gene expression analysis

Our first example is taken from the GSE48091 data-set available from the gene-expression-omnibus¹ uploaded in 2015. The original data-set comprises 28499 gene-expression measurements (referred to later as ‘genes’ for simplicity) collected across 623 patients diagnosed with primary breast cancer.

Within this data-set some of these patients were ‘cases’ that developed distant metastatic disease, whereas others were ‘controls’ that did not. This data-set was prepared specifically so that the control-population could be compared to the case-population: the control-population was randomly matched to the case-population by adjuvant therapy, age and calendar period at diagnosis [1]. The original data-set also had a nested case-control design, with several levels to the case-control hierarchy: most of the patients (506/623) were assigned to the base-level of this hierarchy, while a minority (117/623) were assigned to deeper levels of the hierarchy. For this example we will limit our analysis to the patients at the base-level: i.e., we’ll focus on the $M_D = 340$ base-level ‘cases’ (who developed distant metastatic disease) and $M_X = 166$ base-level ‘controls’ (who did not). This restriction will allow us to perform a straightforward comparison between cases and controls.

Similar to the original research for which this data-set was created, our ultimate objective will be to find signals that are specific to those case-patients that developed distant metastatic disease. A natural first step towards answering this question would be to look for genes which are differentially-expressed with respect to the case- and control-populations. There are indeed many such genes. In fact, 11711 of the original 28499 genes are significantly differentially-expressed (with a p-value less than 0.05). Many of these differentially-expressed genes have already been considered by the original paper (see [1]).

Example-1a: biclustering the strongly differentially-expressed genes

To begin with, we’ll analyze these $N = 11711$ strongly differentially-expressed genes. This data is shown in Fig 2, with each column (i.e., gene) normalized to have median 0 across the patient-population.

Our goal will be to find a bicluster within the case-population. To achieve this goal we can run our control-corrected loop-counting algorithm (described in section 8). Alternatively, because we know a-priori that the bicluster we are searching for involves differentially-expressed genes, we can instead run our control-corrected half-loop algorithm (described in section 15.2).

Regardless of which method we choose, the results in both cases will be very similar. This is because the data-set shown in Fig 2 contains a very large case-specific bicluster. The exact delineation of this bicluster (i.e., which genes and patients are inside versus outside) will depend on the specific methodology used (see section 14.3). Nevertheless, both algorithms will return essentially the same result (i.e., the detected bicluster will be very similar, with a large overlap in terms of patients and genes).

Shown in Fig 3 is an illustration of this large bicluster – comprising $m = 65$ of the $M_D = 340$ cases and $n = 3984$ of the $N = 11711$ genes, detected using our half-loop algorithm. For this example we set the internal parameter γ in our algorithm to ≤ 0.05 (see Fig 32 later on for justification). If we had used our loop-counting algorithm instead of our half-loop algorithm, then the detected bicluster would have been almost the same (i.e., exhibiting a very large overlap with the one shown).

The bicluster shown in Fig 3 consists of genes that, taken individually, are either significantly over-expressed or under-expressed – relative to the control population. We will later refer to this kind of bicluster as a ‘rank-0’ bicluster, because the rows/columns of the bicluster cluster together near a single point in high-dimensional space (see section 15.2). To illustrate the stereotyped differential-expression pattern within this bicluster, we replot the bicluster at the top of Fig 4, and below we plot the control data – reorganized to reveal the differential-expression. While there are some control-patients (towards the bottom of the picture) that exhibit similar expression-levels to the case-patients from the bicluster, the majority do not.

This statement can be quantified as follows: Let’s define $v \in \mathbb{R}^n$ to be the dominant right-principal-compont of this bicluster, and let c_j be the pearson’s-correlation between the j^{th} -patient in the bicluster and v (recall that the pearson’s correlation is a measure of correlation that is normalized to lie within -1 and $+1$). If we use the value c_j as a measure of ‘alignment’, most of the rows in the bicluster are aligned (with the stereotyped pattern) at a value of 75% or more. By contrast, most of the rows in the control-matrix are aligned with a value of -50% to $+10\%$; only 1 of the 166 control-patients has an alignment greater than 75%. This statement can be further quantified as follows: The distribution of alignments c_j for the patients in the bicluster is significantly different than the distribution of alignments for the controls; the AUC (i.e., area under the receiver-operator-characteristic curve) for these two distributions is $> 99\%$, meaning that there is a $> 99\%$ chance that a randomly drawn patient from the bicluster will have a higher alignment than a randomly drawn control. Note that this AUC only reveals that the gene-expression pattern is significantly different between patients within and outside this bicluster. This AUC by itself does not imply that this bicluster is statistically significant or biologically relevant. Put another way, this AUC only implies a high prediction accuracy when discriminating cases *within* the bicluster from the controls; this AUC does not necessarily translate into high case/control prediction accuracy overall.

¹found at <http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE48091>

N = 11711 ‘genes’

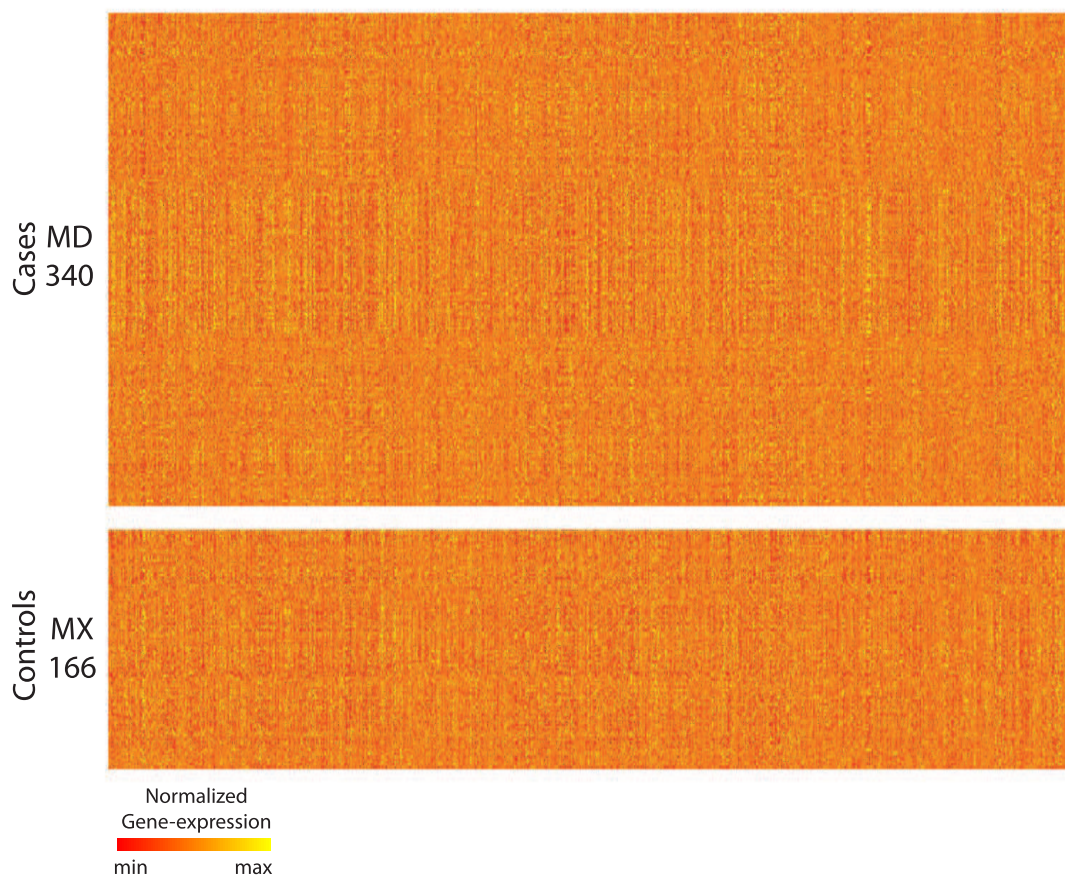


Figure 2: This figure illustrates the GSE48091 gene-expression data-set used in Example-1a. Each row corresponds to a patient, and each column to a ‘gene’ (i.e., gene-expression measurement): the color of each pixel codes for the intensity of a particular measurement of a particular patient (see colorbar to the bottom). $M_D = 340$ of these patients are cases, the other $M_X = 166$ are controls; we group the former into the case-matrix ‘ D ’, and the latter into the control-matrix ‘ X ’. The original data-set had 28499 genes; here we focus only on the $N = 11711$ genes that are each differentially-expressed with respect to the case- and control-populations at a significance level of $p < 0.05$.

Based on the results above, we are lead to ask: How statistically-significant is this bicluster, and is it biologically relevant? As described later on in section 14.2 and Figs 61 and 62, this bicluster has a P-value of $\lesssim 0.002$, which we obtain by comparing it against the distribution of biclusters obtained under a suitable ‘label-shuffled’ null-hypothesis (i.e., formed from shuffling the case-vs-control labels). This level of statistical significance implies that this signal is ‘real’, and suggests that many of the genes implicated in this bicluster may be important for distant metastatic disease.

If this bicluster were indeed biologically-relevant, then we would expect many of these genes to serve similar functions or affect the same pathway. This is certainly the case: We used ‘Seek’ [M42] to perform a gene-enrichment analysis on these $n = 3984$ genes. Using the ‘go_bp_iaa’ ontology, this gene-enrichment analysis revealed a significant enrichment for processes related to mitosis ($p=2e-10$), chromosome-segregation ($p=3e-9$), cell division ($4e-8$), DNA-dependent-DNA-replication ($p=4e-5$), spindle-organization ($p=2e-6$), microtubule organization ($p=1e-4$) and many more. A full list can be found in the attached worksheet ‘S1 Data’. Each page of this worksheet lists the enrichment results using one of the 11 different gene-ontology databases available within the ‘Seek’ software.

Example-1b: biclustering the remaining genes

Now we’ll look at the remaining $N = 28449 - 11711 = 16738$ genes. By construction, these genes are not significantly differentially-expressed with respect to the case- and control-populations; such genes are often ignored by many conventional analyses. These remaining genes are shown in Fig 5, where again each column (i.e., gene) is normalized to have median 0 across the patient-population.

Our goal will be to find a bicluster within the case-population. Because we don’t expect any of the genes in this population to be differentially-expressed, we will look for a bicluster that is ‘low-rank’. That is to say, we’ll look for a subset of genes that are co-expressed (i.e., correlated) across a subset of the patients. It is important to note, however,

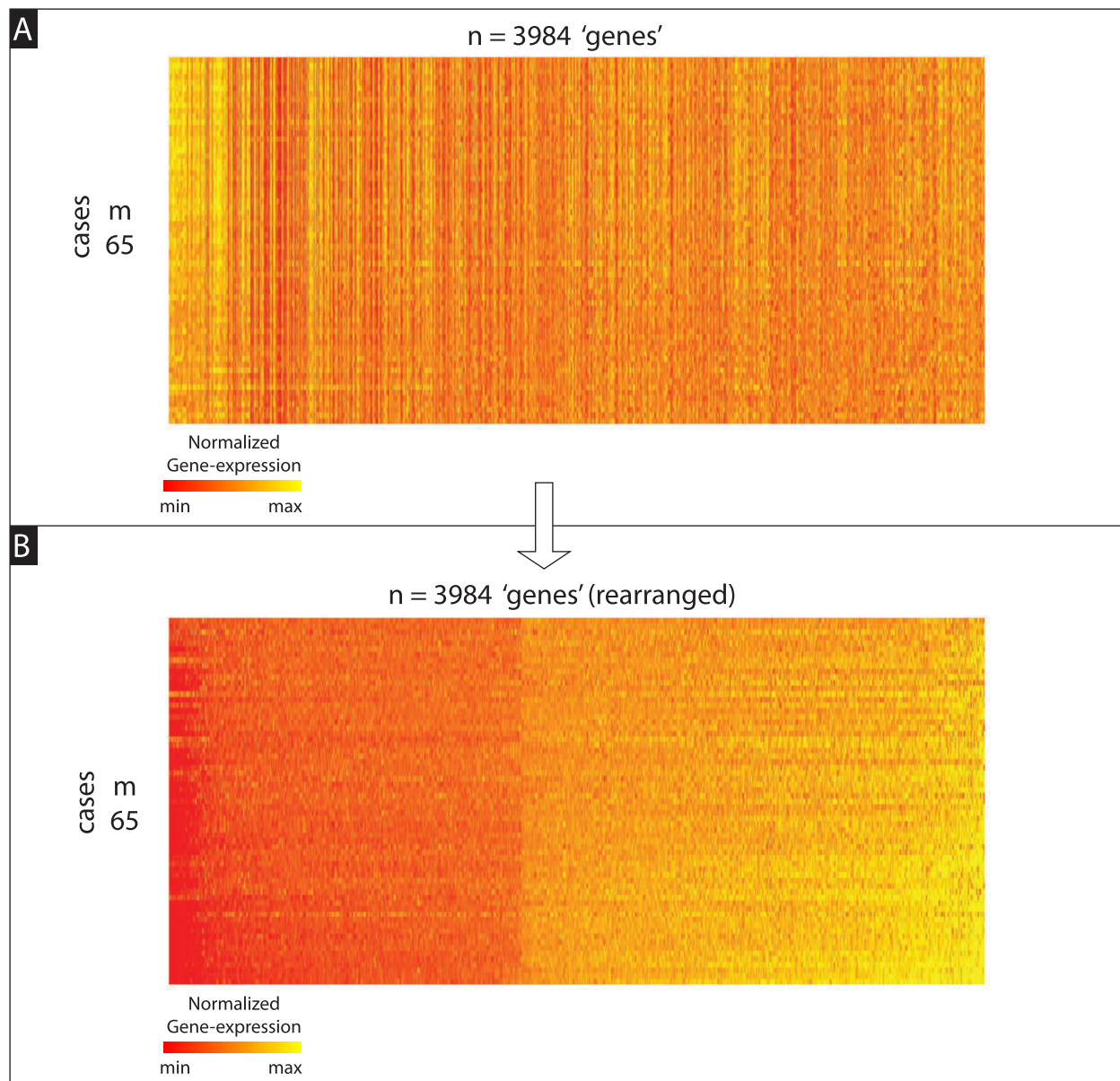


Figure 3: Both panels illustrate the same submatrix (i.e., bicluster) drawn from the full case-matrix shown at the top of Fig 2. This bicluster was found using our control-corrected biclustering algorithm (described in sections 15.2). In Panel-A we represent this bicluster using the row- and column-ordering given by the output of our algorithm. This ordering has certain advantages, which we'll discuss later on, but does not make the differential-expression pattern particularly clear to the eye. Thus, to show this differential-expression more clearly, we present the bicluster again in Panel-B, except this time with the rows and columns rearranged so that the coefficients of the first principal-component-vector change monotonically. As can be seen, there is a striking pattern of differential-expression across the 3984 genes for the 65 cases shown.

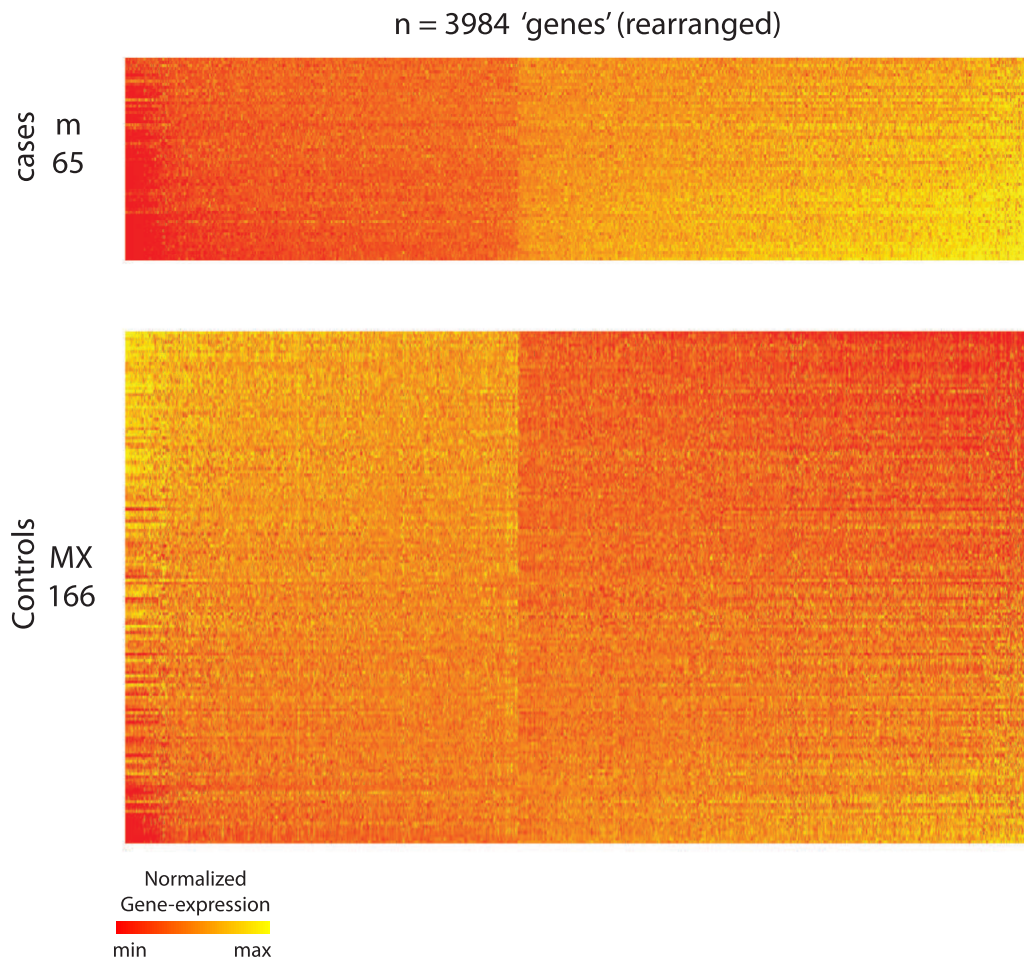


Figure 4: This shows the bicluster of Fig 3B on top, and the rest of the controls from Fig 2 on the bottom. The control-patients have been rearranged in order of their correlation with the co-expression pattern of the bicluster. Even though one of the controls (i.e., $\sim 1/166$) exhibit a coexpression pattern comparable to that expressed by the bicluster, the vast majority do not.

that many genes are correlated across the entire population — including both the cases and controls. In other words, this data-set includes a few ‘low-rank clusters’ which are not specific to either the cases or the controls. Due to these non-specific correlations, the largest low-rank submatrices within this kind of gene-expression data are typically not informative of case-control status — these submatrices will simply include all the genes that are expressed in a coordinated fashion across the entire population.

Our objective will be somewhat more specific. We’ll search this data-set for case-specific biclusters – namely subsets of genes that are structured in some way across a significantly large subset of the case-patients, while not being similarly structured across the control-population. To achieve this goal we will run our control-corrected loop-counting algorithm (described in section 8). As usual, for this (and all subsequent examples), we set the internal parameter $\gamma \leq 0.05$.

Shown in Fig 6 is an illustration of a large bicluster – comprising $m = 45$ of the $M_D = 340$ cases and $n = 793$ of the $N = 16738$ genes, detected using our loop-counting algorithm. This bicluster consists of genes that, taken individually, are neither significantly over-expressed nor under-expressed – relative to the control population. Instead, these 793 genes are significantly co-expressed (i.e., either strongly correlated or anti-correlated) across a significant fraction of the case population (in this case $45/340 \sim 13\%$ of the cases), without being as significantly co-expressed across a comparable fraction of the control population. While a little difficult to see in Fig 6A, we’ve rearranged the bicluster in Fig 6B to reveal its co-expression pattern; each patient in the bicluster is either strongly correlated or anti-correlated with this stereotyped pattern. We will later refer to this kind of bicluster as a ‘low-rank’ bicluster; in this case the rank is approximately 1.

This statement can be quantified as follows: Similar to Example-1a, we’ll define $v \in \mathbb{R}^n$ to be the dominant right-principal-component of this bicluster, and let c_j be the pearson’s-correlation between the j^{th} -patient in the bicluster and v . This time, however, we’ll use the absolute-value $|c_j|$ as a measure of ‘alignment’; we take absolute-values to include both strong positive- and negative-correlations in our definition of co-expression. When using this definition of alignment, most of the rows in the bicluster are aligned (with the stereotyped pattern) at a value of 90% or more. By contrast, most of the rows in the control-matrix are aligned with a value of 0% to 50%; only 3 of the 166 control-patients have an alignment greater than 90%.

To illustrate the stereotyped co-expression pattern within this bicluster, we replot the bicluster at the top of Fig 7, and below we plot the control data – reorganized to reveal the co-expression. While there are some control-patients (towards the top and bottom of the control-matrix) that exhibit similar alignment to the case-patients from the bicluster, the majority do not. This statement can be further quantified: The distribution of alignments $\|c_j\|$ for the patients in the bicluster is significantly different than the distribution of alignments for the controls; the AUC for these two distributions is $> 98\%$. As before, this AUC only reveals that the gene-expression pattern is significantly different between patients within and outside this bicluster, and does not imply that this bicluster is statistically significant or biologically relevant.

Using the same methodology as before, we can ask whether or not this bicluster is statistically-significant, and whether or not it is biologically relevant. As shown in Figs 57 and 58 in section 14.2, this bicluster has a P-value of $\lesssim 0.008$ (which, as before, we obtained by comparing this bicluster against the distribution of biclusters obtained under a ‘label-shuffled’ null-hypothesis). Using the ‘go_bp_iea’ ontology in ‘Seek’ to perform gene-enrichment-analysis on the $n = 793$ genes in this bicluster, we find significant enrichment for mitosis ($p=2e-9$), DNA-replication ($3e-8$), chromosome segregation ($p=2e-5$) and many more; including several pathways that are likely to play a role in the development of cancer. A full list can be found in the attached worksheet ‘S2 Data’. Each page of this worksheet lists the enrichment results using one of the 11 different gene-ontology databases available within the ‘Seek’ software.

We remark that the $n = 793$ gene-expression measurements for the bicluster shown in Fig 6A are completely distinct from the $n = 3984$ differentially-expressed gene-expression-measurements shown in Fig 3A. The patient-subsets are also largely distinct; in fact, the $m = 45$ patients shown in Fig 6A and the $m = 65$ patients shown in Fig 3A have an overlap of only 4, significantly less than one would expect by chance ($p < 0.04$).

1.3 Additional information for Example-2: Gene expression analysis

Our second example is taken from the GSE17536 data-set available from the gene-expression-omnibus² uploaded in 2009. See the supplementary-tutorial ‘S1 Source Code’ for the matlab source code, as well as a full description of how this data-set was pre- and post-processed. This tutorial can be used to reproduce the results in this section (using the ‘n2x’ normalization convention), and the source code can be used to bicluster many other gene-expression data sets as well (including the GSE48091 set shown previously).

The subset of data that we use comprises $N = 17942$ gene-expression measurements (i.e., ‘genes’) collected across 175 patients, each diagnosed with colorectal-cancer. Of these patients, $M_D = 55$ patients have already died, with colorectal-cancer determined to be the significant cause-of-death. The remaining $M_X = 120$ patients either have not yet died (as of 2009), or have died of other causes.

Similar to the original research from which this data is drawn, we’ll try to find signals that are related to mortality [3, 2]. With this objective in mind, we’ll use the cause-of-death to divide our patient population into cases and controls, respectively. While using the cause-of-death as a case-control classification is far from ideal [4], we’ll proceed under the

²found at <http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE17536>

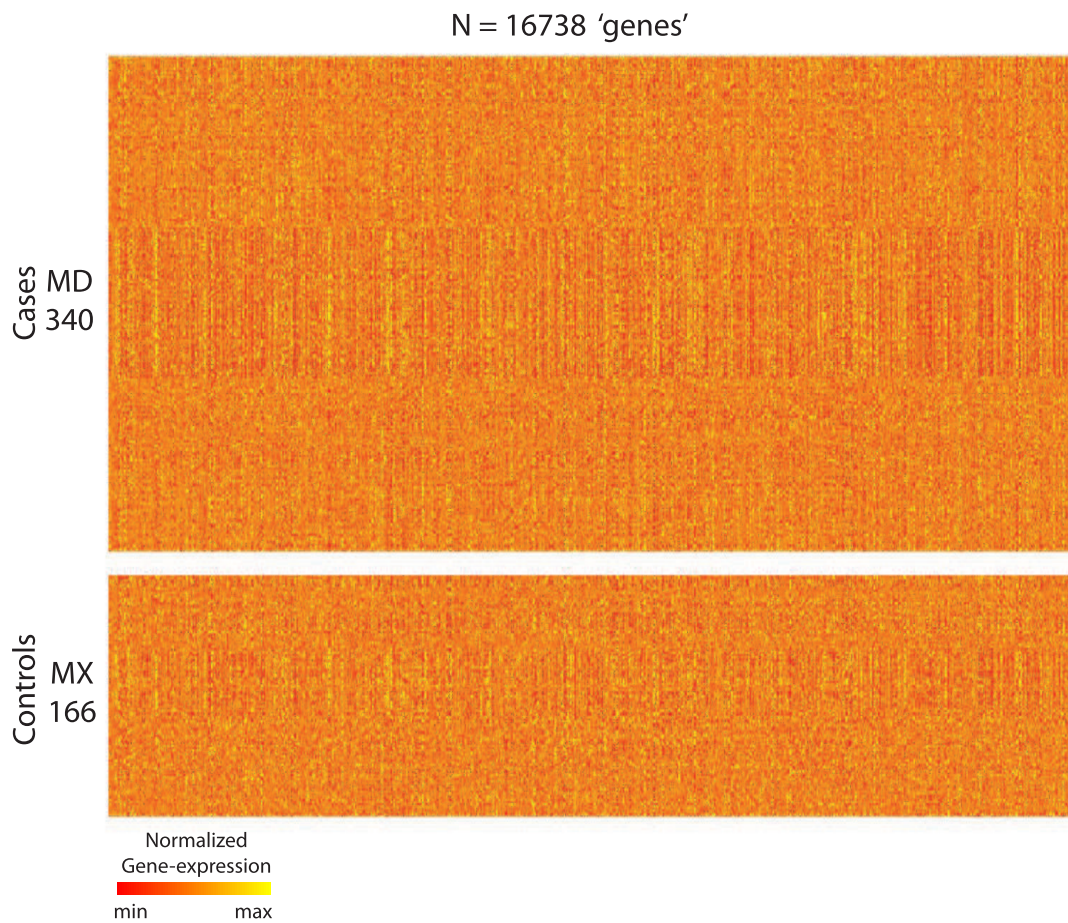


Figure 5: This figure illustrates the GSE48091 gene-expression data-set used in Example-1b (see Example-A in the main-text). The format is similar to Fig 2, except that this time we illustrate the $N = 16738$ genes that are *not* significantly differentially-expressed with respect to the case- and control-populations (i.e., we only include those genes which were excluded in Fig 2).

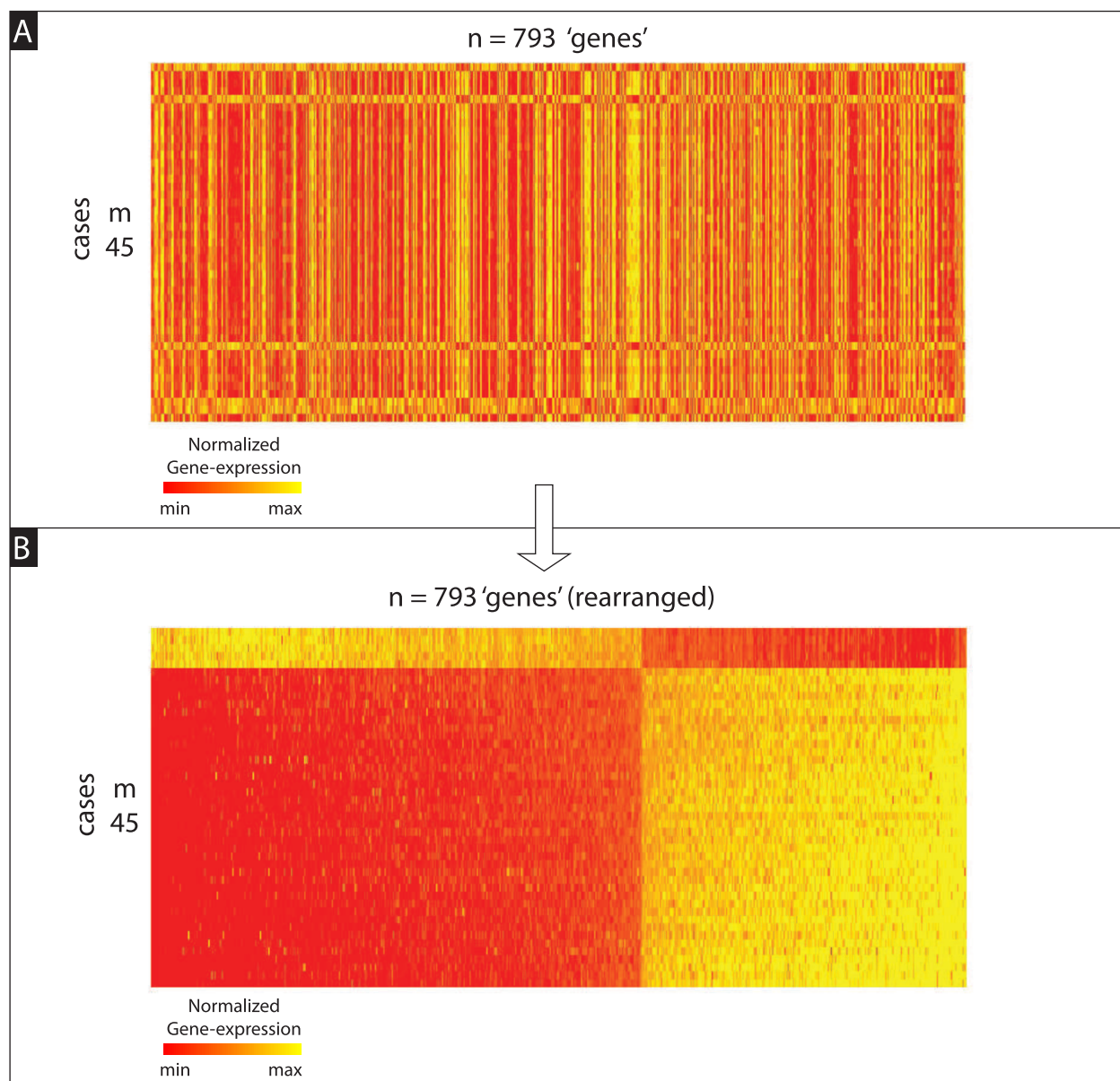


Figure 6: Both panels illustrate the same submatrix (i.e., bicluster) drawn from the full case-matrix shown at the top of Fig 5. This bicluster was found using our control-corrected biclustering algorithm (described in sections 8). The format is similar to Fig 3, except that this time the bicluster includes fewer genes and patients. Nevertheless, there is a striking pattern of differential-expression across the 793 genes for the 45 cases shown. As we discuss in the text, this bicluster is largely distinct from the bicluster shown in Fig 3; the genes are completely distinct, and the patient-overlap is significantly lower than that expected by chance.

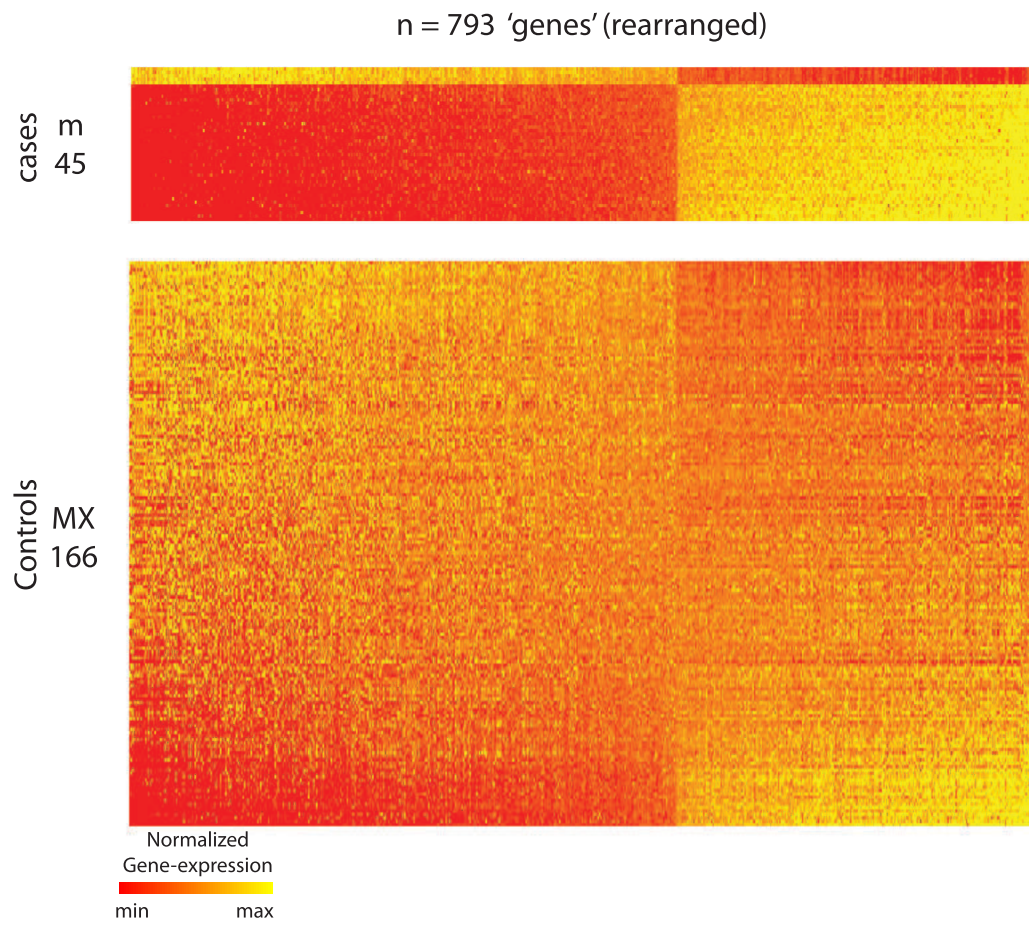


Figure 7: This shows the bicluster of Fig 6B on top, and the rest of the controls from Fig 5 on the bottom. The control-patients have been rearranged in order of their correlation with the co-expression pattern of the bicluster. Even though a few of the controls (i.e., $\sim 3/166$) exhibit a coexpression pattern comparable to that expressed by the bicluster, the vast majority do not.

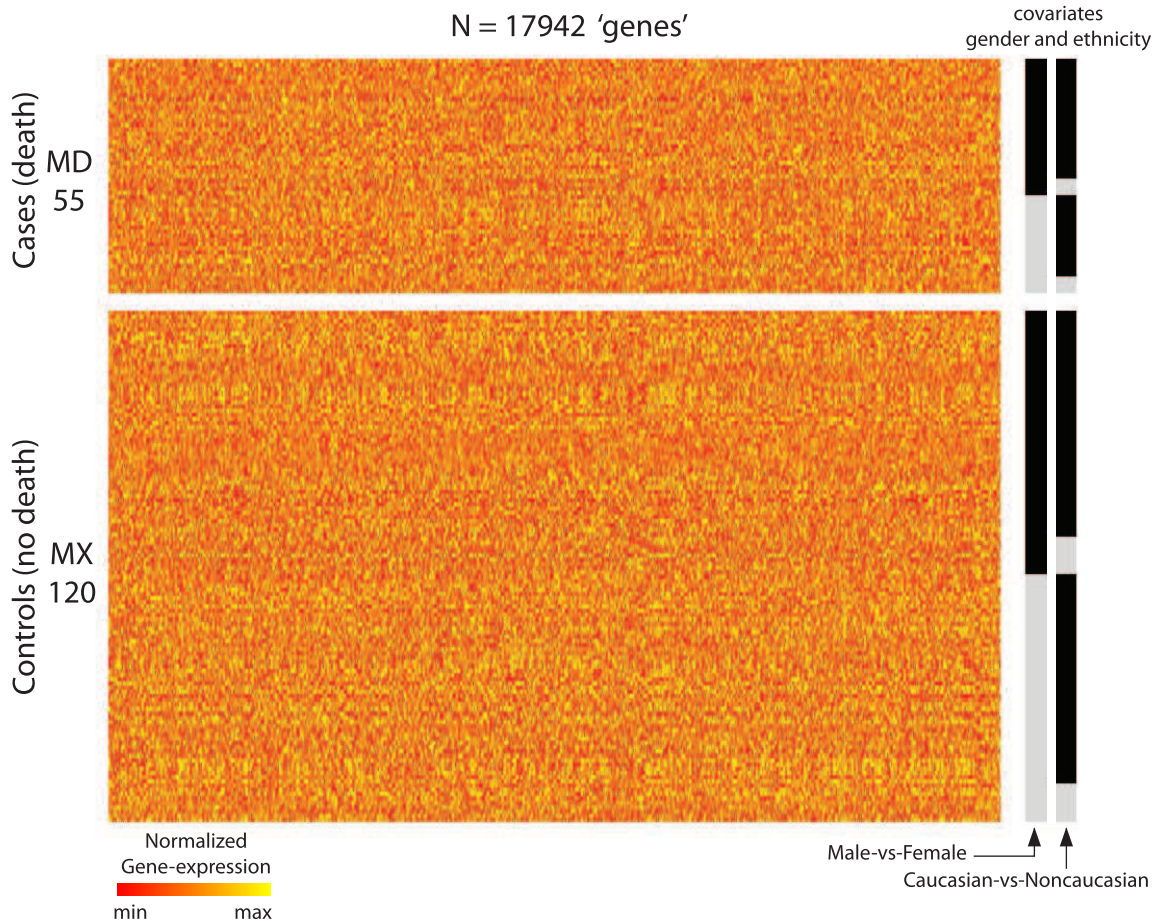


Figure 8: This figure illustrates the GSE17536 gene-expression data-set used in Example-2. The format is similar to Fig 5. For this data set only $M_D = 55$ of the patients are cases, the other $M_X = 120$ are controls; we group the former into the case-matrix ‘ D ’, and the latter into the control-matrix ‘ X ’. The covariates are shown to the far right, (i.e., grey vs black). Given two binary categories, there are a total of $I_{\text{cat}} = 4$ covariate-categories in total (ranging from caucasian-male to non-caucasian-female). These covariate-categories can be used to further divide the case- and control-populations.

assumption that the cause-of-death is at least somewhat correlated with the severity of cancer. Our hope is that the subset of case-patients (that died of cancer) will be enriched for patients that exhibit disease-related co-expression across some subset of their genes.

The data-set is illustrated in Fig 8. Note that, aside from their gene-expression data, each patient is also endowed with a variety of other characteristics, including their gender and ethnicity, both of which we’ll use as covariates in our analysis.

As in section 1.2, we’ll search this data-set for case-specific biclusters. These case-specific biclusters will pinpoint genes useful for diagnosis and discrimination between case and control status. While conducting this search, we’ll also try and correct for the covariates. That is, we’ll try and focus our efforts on biclusters which include a reasonable mixture of genders and ethnicities.

Fig 9 shows a large bicluster – comprising $m = 14$ of the $M_D = 55$ cases and $n = 966$ of the $N = 17942$ genes – that was hidden within the case-matrix and discovered using a covariate-corrected version of our algorithm. This bicluster consists of genes that, taken individually, are neither significantly over-expressed nor under-expressed – relative to the control population. Instead, these 966 genes are significantly co-expressed (i.e., either strongly correlated or anti-correlated) across a significant fraction of the case population (in this case $14/55 \sim 25\%$ of the cases), without being as significantly co-expressed across a comparable fraction of the control population. Similar to the bicluster from Example-1b from section 1.2, this bicluster is a ‘low-rank’ bicluster of rank approximately 1.

To illustrate that this stereotyped co-expression pattern is indeed case-specific (i.e., not comparably shared across the controls), we replot the bicluster at the top of Fig 10 and below we plot the control data – reorganized in an attempt to reveal co-expression patterns. As one can see, while there are certainly some control patients that exhibit strong correlation or anti-correlation with the stereotyped gene-expression pattern of the bicluster, the majority are not so strongly aligned. Using our definition of $|c_j|$ as alignment (from Example-1b in section 1.2), we see that almost all the rows in the bicluster

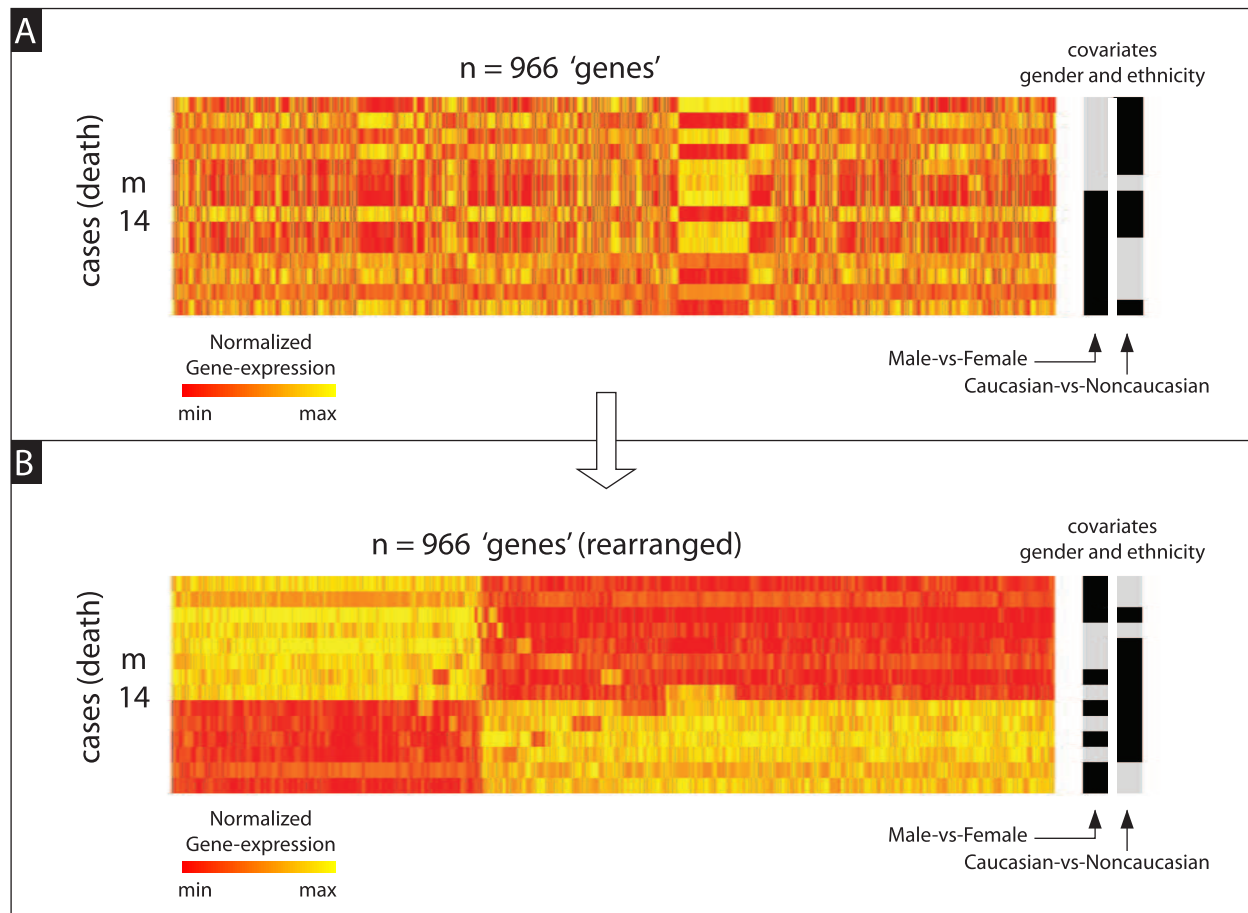


Figure 9: Both panels illustrate the same submatrix (i.e., bicluster) drawn from the full case-matrix shown at the top of Fig 8. This bicluster was found using our covariate-corrected biclustering algorithm (described in sections 8 and 9). In Panel-A we represent this bicluster using the row- and column-ordering given by the output of our algorithm. This ordering has certain advantages, which we'll discuss later on, but does not make the co-expression pattern particularly clear to the eye. Thus, to show this co-expression more clearly, we present the bicluster again in Panel-B, except this time with the rows and columns rearranged so that the coefficients of the first principal-component-vector change monotonically. As can be seen, there is a striking pattern of correlation across the 966 genes for the 14 cases shown.

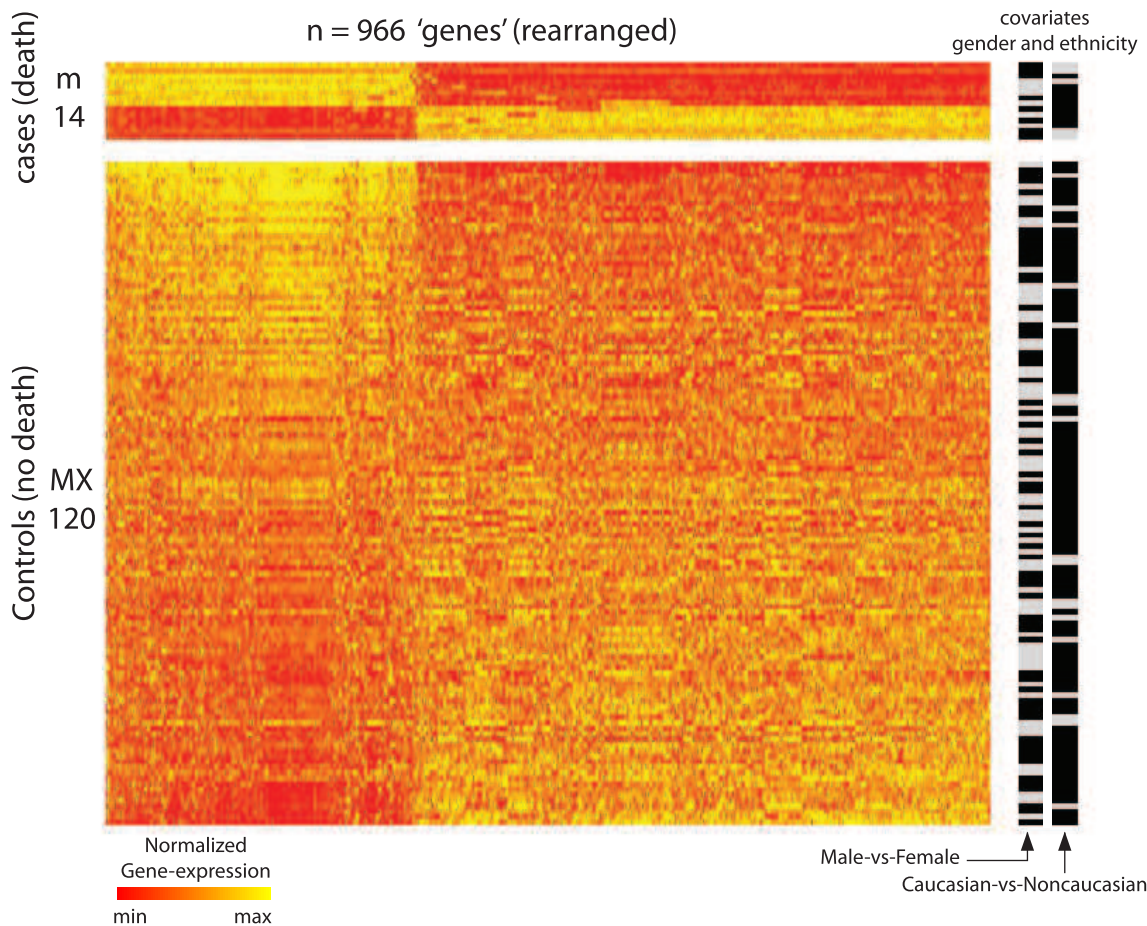


Figure 10: This shows the bicluster of Fig 9B on top, and the rest of the controls on the bottom. The control-patients have been rearranged in order of their correlation with the co-expression pattern of the bicluster. Even though some of the controls (i.e., $\sim 5/120$) exhibit a coexpression pattern comparable to that expressed by the bicluster, the vast majority do not.

are aligned at a value of 70% or more, whereas only 5 of the 120 controls (i.e., significantly less than 14/55) have an alignment that is greater than 70%; most only exhibit an alignment around 20% – 30%. The distribution of alignments $|c_j|$ for the patients in the bicluster is significantly different than the distribution of alignments for the controls; the AUC for these two distributions is 97%. As before, this AUC only reveals that the gene-expression pattern is significantly different between patients within and outside this bicluster, and does not imply that this bicluster is statistically significant or biologically relevant.

At this point we remark on the covariates. Gender and ethnicity are both covariates that strongly influence the expression levels of many genes throughout the body. It is therefore rather typical for gene-expression data-sets to contain large biclusters driven solely by these covariates. If we were to run our biclustering algorithm without taking these covariates into account, it is likely that we would reveal one of these large covariate-driven biclusters. To account for this we conducted this example using a covariate-corrected version of our algorithm. The covariate-corrected algorithm attempts to find biclusters that are well balanced across the covariate-categories. The algorithm was successful in this case: as seen in Fig 9, the 14 patients that exhibit this co-expression are not all male, nor all female, nor all of one ethnicity. This balance is reassuring, and hints that the signal we are seeing is not driven solely by gender or ethnicity.

Finally, we can ask: How statistically-significant is this bicluster, and is it biologically relevant? As described later on in section 14.2 and Figs 59 and 60, this bicluster has a P-value of ~ 0.027 , which we obtain by comparing it against the distribution of biclusters obtained under an appropriate 'label-shuffled' null-hypothesis (i.e., formed from shuffling the case-vs-control labels, while respecting covariate categories). This level of statistical significance implies that this signal is 'real', and suggests that many of the genes implicated in this bicluster may be important for colorectal cancer.

Using the 'go_bp_iea' ontology in 'Seek' to perform gene-enrichment analysis on the $n = 966$ genes within the bicluster, we find significant enrichment for processes related to mitosis ($p=9e-17$), chromosome-segregation ($p=9e-12$), DNA-dependent-DNA-replication ($p=9e-9$), microtubule organization ($p=2e-8$), spindle-organization ($p=6e-8$), RNA-splicing ($p=1e-5$), mitotic recombination ($p=5e-4$), and many more, including many pathways that are likely to play a role in the

development of cancer. A full list can be found in the attached worksheet ‘S3 Data’. Each page of this worksheet lists the enrichment results using one of the 11 different gene-ontology databases available within the ‘Seek’ software.

To further probe the potential relevance of this bicluster, we can check to see if the $n = 966$ genes in the bicluster overlap significantly with ‘recognized’ genes that are already understood to play a role in colon-cancer. For this example we’ll generate this latter list by first choosing three of the most well documented genes which influence colon-cancer: MLH1, MSH2 and MSH6 [5]. We then use ‘Seek’ once again to generate a list of genes that are commonly co-expressed alongside these three cancer-related genes. We then define our list of ‘recognized’ genes to be all the genes which have a combined co-expression value (with MLH1, MSH2 and MSH6) of at least 0.75. The list of recognized-genes generated this way comprises 1659 of our original 17941 genes. After generating this list, we can check to see if the $n = 966$ genes in our bicluster significantly overlap with the list of recognized-genes. Indeed they do: the intersection is 174, which is significantly higher than chance ($p=1e-20$).

Given the analysis above, it comes as no surprise that the list of recognized-genes also enriches strongly for the same pathways we saw within our bicluster (i.e., DNA-replication, DNA-repair, mitosis, etc.). While this similarity is encouraging (and further suggests that our bicluster might be biologically relevant), it begs the following question: Could we possibly search for a bicluster within this data set that was distinct from the genes that are already well-recognized? Such a search might unveil previously undiscovered sets of genes that also play a role the pathology of colon cancer for certain subsets of patients. We perform this analysis later on in section 15.1, using this same list of recognized-genes as genetic-controls.

Before we move on to the next example, we add one final comment on the advantages of biclustering. As we mentioned above, biclustering allows us to search for subsets of genes that are coexpressed across only a subset of the patients. Because the biclustering process is capable of ignoring ‘noisy’ patients that do not participate in the coexpression pattern, it is possible for our algorithm to find gene-subsets that are very tightly related. More traditional algorithms that try to cluster genes (across the entire case-patient population) may not be as successful in highlighting relevant gene-subsets. For example, we can contrast our biclustering approach with a standard logistic-regression analysis (performed on a gene-by-gene basis across the full set of case- and control-patients after rank-normalizing the data). To compare the logistic-regression results with our biclustering, we consider the top $n = 966$ genes with the most significant regression-coefficients (i.e., the $n = 966$ ‘top-hits’). These top-hits do not display many significant enrichments for disease-related processes. In fact, the only significant enrichment we found involved a Cytosolic DNA sensing pathway ($p=5.6e-4$) (using the kegg ontology). Moreover, these top-hits did not significantly overlap with the list of recognized-genes (intersection 91, p -value ~ 0.5). In summary, for this example it seems that a standard logistic-regression analysis does not identify the same kinds of disease-related signals as our biclustering approach.

1.4 Additional Information for Example-3: Genome-Wide-Association-Study (GWAS)

Our third example is a subset of a Genome-Wide-Association-Study used with permission from the Bipolar Disorders Working Group of the Psychiatric Genomics Consortium (PGC-BIP) [6]³. This data-set includes $N = 276768$ alleles genotyped across 16577 patients of european ancestry. These patients are drawn from the following studies (i.e., cohorts): `bip_gain_eur_sr-qc`, `bip_dub1_eur_sr-qc`, `bip_top7_eur_sr-qc`, `bip_swa2_eur_sr-qc`, `bip_fat2_eur_sr-qc`, `bip_wtcc_eur_sr-qc`, `bip_edi1_eur_sr-qc`, `bip_uclo_eur_sr-qc`, `bip_stp1_eur_sr-qc` and `bip_st2c_eur_sr-qc` described within the supplementary information of [7] and the supplementary note of [8].

The patients themselves fell into two phenotypic categories: 9752 are neurotypical, whereas the remaining 6825 exhibit a particular psychiatric disorder. For this example we’ll try and find a signal within the neurotypical patients that is not shared by those with the disorder; We’ll use the phenotypic information to divide the patients into $M_D = 9752$ cases and $M_X = 6825$ controls. Note that, for this example, our nomenclature is non-standard; neurotypical patients are typically referred to as ‘controls’ and not cases. The reason we deviate from this standard is because, below, we will try to find a bicluster within the neurotypical patients that does not extend to include the remaining patients. In order to remain consistent with our notation and equations in the rest of this manuscript, we will refer to these neurotypical patients as cases, and we will store their information in the case-matrix D .

To select our set of alleles, we considered all the fully-genotyped alleles available for the studies listed above, limiting ourselves to those with a minor-allele-frequency of at least 10%. We used all the alleles that fell within this bound; We did not use imputation or correct for linkage-disequilibrium. In addition to their genotyped data, each patient is also associated with an $N_T = 2$ -dimensional vector of ‘mds-components’ that serve as a continuous-covariate. In this case the continuous-covariate plays the role of a proxy for each patient’s genetic ancestry.

Our objective in this situation is similar to Example-1b and Example-2: We would like to search for case-specific biclusters involving subsets of alleles that are structured in some way across a significantly large subset of the case-patients, while not being similarly structured across the control-population. In addition, we’d like to ensure that the

³Due to data-usage agreements, only cursory information regarding this data-set will be provided here. A more detailed description of the data-set, as well as the structures we’ve found within it, will be provided in a later publication.

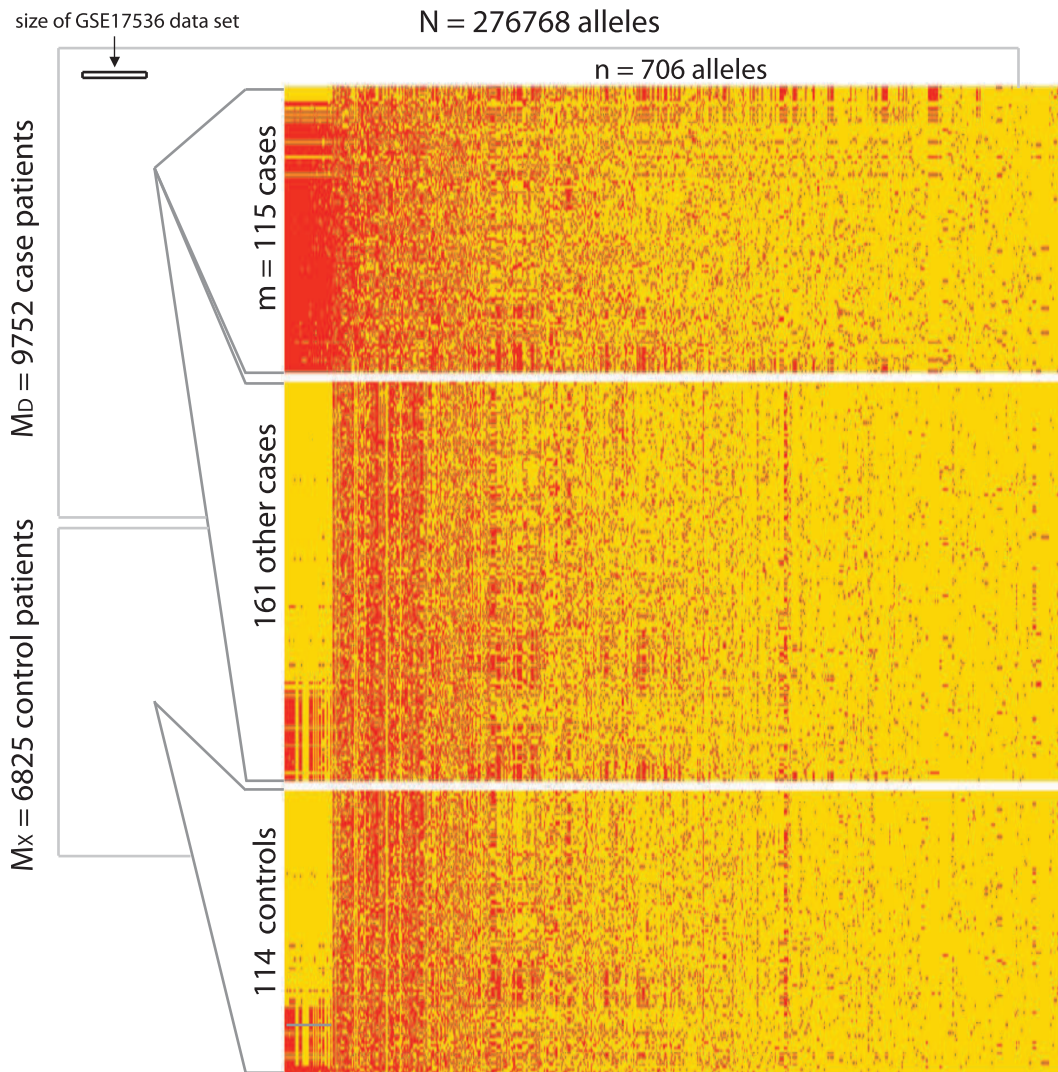


Figure 11: In this figure we illustrate the genome-wide association-study (i.e., GWAS) data-set discussed in Example-3 (see Example-B in the main text). This data-set involves 16577 patients, each genotyped across 276768 genetic base-pair-locations (i.e., alleles). Many of these patients have a particular psychological disorder, while the remainder do not. We use this phenotype to separate the patients into $M_D = 9752$ cases and $M_X = 6825$ controls. The size of this GWAS data-set is indicated in the background of this picture, and dwarfs the the size of the gene-expression data-set used in Example-2 (inset for comparison). At the top of the foreground we illustrate an $m = 115$ by $n = 706$ submatrix found within the case-patients. This submatrix is a low-rank bicluster, and the alleles are strongly correlated across these particular case-patients. The order of the patients and alleles within this submatrix has been chosen to emphasize this correlation. For comparison, we pull out a few other randomly-chosen case-patients and control-patients, and present their associated submatrices (defined using the same 706 alleles) further down.

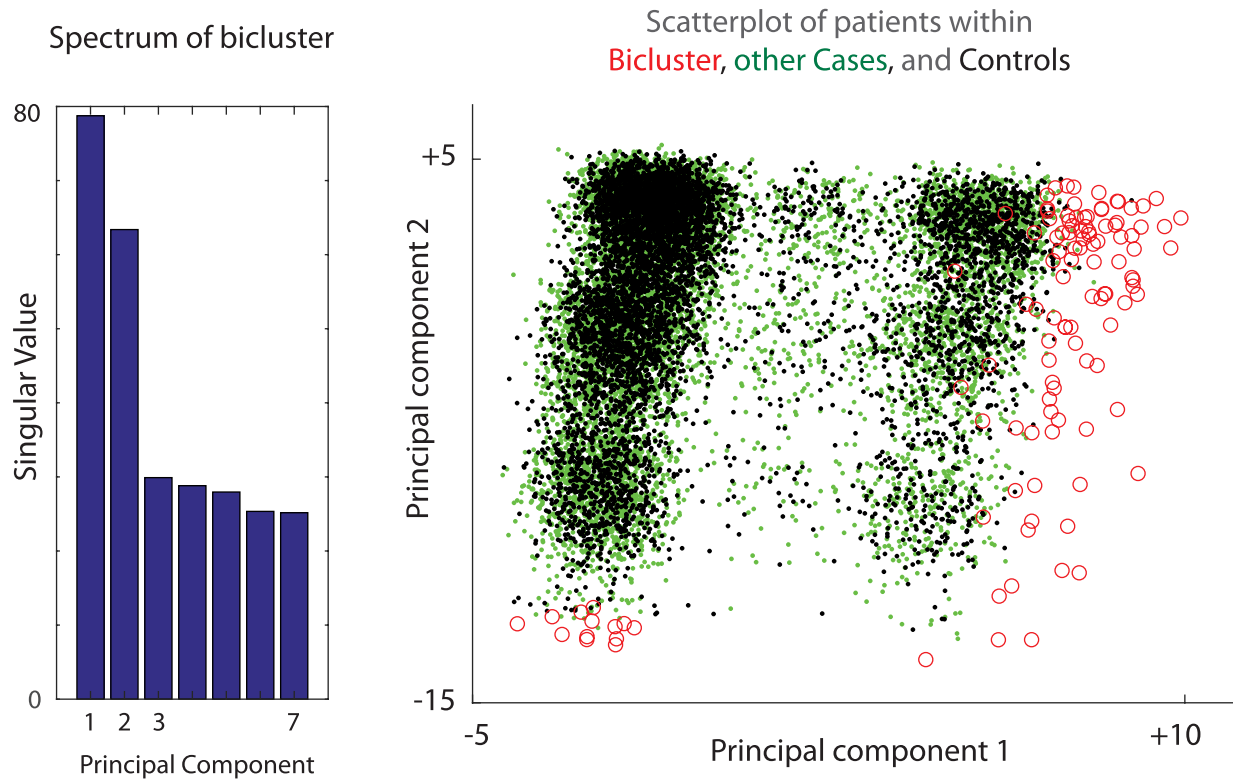


Figure 12: On the left we show the first 7 singular-values of the bicluster shown in Fig 11. Note that the first two singular-values are much larger than the rest; this bicluster is effectively rank-2. On the right we show a scatterplot of the patients in the data-set, as projected onto the first two dominant principal-components of the bicluster. The patients within the bicluster are shown with red circles. The remaining case-patients are shown with green dots, whereas the control-patients are shown with black-dots. Note that the patients within the bicluster mostly lie outside the distribution defined by the other patients, but that the bicluster-patients cannot easily be separated from the rest after projecting onto a straight-line (i.e., the bicluster is rank-2, but not rank-1).

biclusters we find are well-distributed with regards to the continuous-covariate (i.e., we don't want to focus on a subset of patients that all have the same ancestry).

Fig 11 illustrates the size of this data-set, as well as one case-specific low-rank bicluster which we discovered. For this example we used a version of our algorithm that corrects for continuous-covariates; specifically, we use the '2-sided' covariate-correction described in section 10. As can be seen from Fig 11, the pattern shown within the bicluster is rather different than the pattern exhibited by the typical control. Indeed, as described later on in section 14.2, this bicluster has a P-value of $\ll 0.05$.

What may not be obvious from visual inspection is that this bicluster is essentially 'rank-2'; i.e., the dominant two principal components of this bicluster are large compared to the rest. Put another way, the patients within this bicluster exhibit a second-order correlation across the subset of alleles in the bicluster; a correlation not exhibited by the population at large. We illustrate this second-order structure in Fig 12. As can be seen in Fig 12B, the distribution of patients within the bicluster is markedly different from the distribution of the other patients. Even though this bicluster is essentially rank-2, a substantial amount of the variance is still captured by the first principal-component. Indeed, if we calculate the distribution of alignments across individuals in this bicluster (similar to Example-1b and Example-2) and compare them to the distribution of alignments across the controls, we obtain an AUC of $> 99.75\%$. Just as in our previous examples, we should interpret this AUC as evidence that the allele-pattern exhibited by the patients in the bicluster is indeed significantly different than the allele-pattern exhibited by the remaining patients. That is to say, this AUC only implies a high prediction accuracy when discriminating cases *within* the bicluster from the controls; this AUC does not translate into high case/control prediction accuracy overall.

We remark that the above observations alone do not prove that the bicluster is biologically significant! The patterns we observe within this bicluster could very well be due to a covariate that we did not correct for, or some other artifact of the data-set. One way to substantiate the biological relevance of this bicluster would be to perform a replication study on an independent data-set. Ideally, the projection of a fresh set of patients onto the first two principal-components of the bicluster discovered above should reveal a similarly distinct set of distributions, with a similar proportion of the case-patients falling far from the other patients and lying near the red-circles shown in Fig 12B.

Nevertheless, given the significance-level and strength of the signal within this bicluster, we might expect many of the alleles from the bicluster to affect the same physiological functions. Using ‘Seek’ once again, we find that the 124 genes involved in these 706 genetic-loci are enriched for many pathways. Using the ‘go_mf_iea’ and ‘kegg’ ontologies, we find enrichment for: phosphate-ion transmembrane transport ($p=1.6e-4$), metal-ion transmembrane transport ($p=5.7e-4$), calcium-ion binding ($p=1.2e-3$), GTPase-activation ($p=2.2e-3$), ion-gated channel activity ($p=3.3e-3$), calcium-ion transport ($p=4.3e-3$), voltage-gated channel activity ($p=1.0e-2$), calcium-signaling ($p=1.1e-3$), long-term-potential ($p=1.1e-3$), glutamatergic-synapses ($p=7e-3$) and many others. A full list can be found in the attached worksheet ‘S4 Data’, with a format similar to that of ‘S1 Data’. Because this bicluster was found within the neurotypical patients, it is possible that these genes play a protective role in delaying the development or onset of the psychiatric disorder associated with this data-set.

In addition to finding this bicluster, our covariate-corrected algorithm has also successfully ensured that this bicluster is balanced with regards to the continuous-covariate. This balance is illustrated in Figs 13 and 14, which shows the joint-distribution (across patients) of the first two mds-components as our algorithm proceeds. If we were to run our algorithm *without* correcting for the continuous-covariates, then we would find spurious biclusters involving patients that were highly concentrated in just a few regions of covariate-space (see Fig 15).

We remark that the patients in this data-set actually have more than just two mds-components; there are also higher-order mds-components numbered 3, 4, 5, etc. that may also correlate with the patients’ ancestry. For this particular example, however, the patients are all of european ancestry; there is not too much structure within the higher-order mds-components. Thus, even though we do not explicitly control for the higher-order mds-components, our algorithm nevertheless produces biclusters which are balanced with regards to these higher-order mds-components. We illustrate this balance by showing the joint-distribution of the mds-components 3+4 in Figs 16 and 17. While this joint-distribution alone does not certify that our algorithm has maintained a balance across all higher-order mds-components, we do indeed see a qualitatively similar phenomenon for the joint-distributions of the mds-components 1+3, 1+4, 2+3 and 2+4. A similar story holds for mds-components 5, 6, etc., which are even more unstructured than components 3 and 4.

It so happens that, for this data-set, each patient is not only associated with the continuous-covariate mentioned above, but also a categorical-covariate; each patient is drawn from one of $I_{\text{cat}} = 10$ studies. These different studies were carried out by different research groups within the PGC, involving different experimental designs and patient populations. Much like the continuous-covariate, these different studies can give rise to spurious signals within the data-set. While we could attempt to correct for both continuous- and categorical-covariates simultaneously, it turns out that many of the individual studies involve patient cohorts that are localized in covariate-space. In other words, for this example the continuous-covariate and categorical-covariate are correlated. Consequently, a bicluster that is balanced with respect to the continuous-covariate will typically involve patients drawn from many different studies. As a result, even though we chose to correct for continuous-covariates alone, this correction was sufficient to ensure that the bicluster found was rather well balanced with respect to study (in addition to being balanced with respect to the continuous-covariate). This balance is demonstrated in Figs 18 and 19.

In the following few sections we describe our algorithm in more detail. We start out with the simplest possible situation, explaining when we expect our algorithm to work and comparing its performance to that of a simple spectral method. Afterwards, we explain how to generalize our algorithm to incorporate controls, covariates and sparse data. Finally, we comment on some practical considerations, such as finding p-values for a bicluster and delineating the boundaries of a bicluster.

2 Simple case: D only

In the simplest situation there are no controls, covariates, or sparsity considerations, and we are tasked with finding low-rank biclusters within an $M \times N$ case-matrix D . In this case our algorithm reduces to the following very simple iteration:

Step 0 Binarize D , sending each entry to either +1 or -1, depending on its sign (i.e., $D = \text{sign}(D)$);

Step 1 Calculate row-scores and column-scores. In their simplest form these are: $Z_{\text{ROW}} = \text{diag}(DD^{\top}DD^{\top})$, and $Z_{\text{COL}} = \text{diag}(D^{\top}DD^{\top}D)$;

Step 2 Restrict attention to the row-indices for which Z_{ROW} is largest (i.e., most positive) and the column indices for which Z_{COL} is largest – e.g., throw away the rows/columns for which Z_{ROW} and Z_{COL} are smallest (i.e., most negative).

Step 3 Go back to step 1.

As a consequence of this simple iteration, the output of the algorithm is a listing of row- and col-indices in the order that they were eliminated. Those rows and columns which are retained the longest are most likely to be part of a low-rank

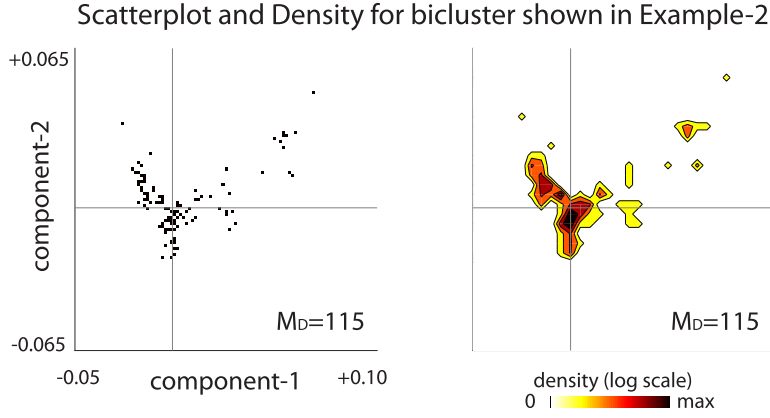


Figure 13: Joint-distribution (across patients) of continuous-covariates for the bicluster shown in Example-3. As mentioned in the introduction, our algorithm proceeds iteratively, removing rows and columns from the case-matrix until there are none left. One of our goals is to ensure that, during this process, our algorithm focuses on biclusters which involve case-patients that are relatively well balanced in covariate-space. On the left we show a scatterplot illustrating the 2-dimensional joint-distribution of covariate-components across the remaining $m = 115$ case-patients within the bicluster shown in Example-3 (i.e., Fig 11). The horizontal and vertical lines in each subplot indicate the medians of the components of the covariate-distribution. On the right we show the same data again, except in contour form (note colorbar). The continuous-covariates remain relatively well-distributed even though relatively few case-patients are left (compare with Fig 14).

bicluster. After finding the first bicluster in this manner, the entries of D corresponding to this first bicluster can be scrambled (i.e., destroying their low-rank structure), and the next bicluster can be found by running the algorithm again⁴.

There are several positive features of this algorithm:

- It works: Specifically, this algorithm solves the ‘planted-bicluster’ problem with high probability. That is to say, if the case-matrix D is a large random matrix containing a hidden low-rank bicluster ‘ B ’ of size $m_B \times n$, then this algorithm will almost always find B as long as the spectrum of B decays sufficiently quickly, and m_B and n are larger than $\sqrt{M_D}$ and \sqrt{N} , respectively. We discuss this in more detail below.
- It’s fast: Specifically, this algorithm requires only a handful of matrix-multiplications per iteration. Furthermore, the binarization step allows for very fast matrix-matrix multiplication that does not use floating-point operations. If desired, the recalculation of row- and col-scores in each iteration can be replaced by a low-rank update (see section 3.1), reducing the maximum total computation time across all iterations to $O(M_D N \max(M_D, N))$ – asymptotically equivalent to matrix-matrix multiplication.
- It’s easy: Specifically, this algorithm has few-to-no parameters. Notably, the user does not need to specify the number, size or rank of the biclusters beforehand. As long as any biclusters are sufficiently low-rank (e.g., rank $l = 1, 2, 3$) and sufficiently large (see the first point), then this algorithm will find them.
- It’s generalizable: Specifically, this algorithm can easily be modified to account for controls and/or covariates. We’ll discuss this after we explain why this algorithm works.

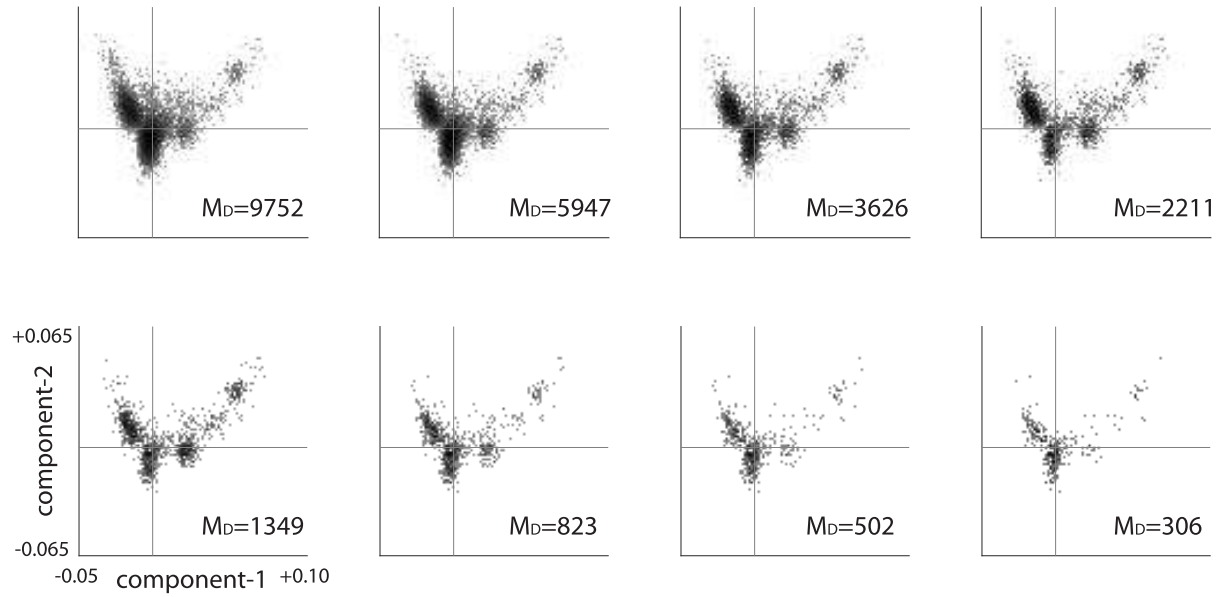
The reason this algorithm is successful is because, when calculated across a large matrix, the row- and col-scores Z_{ROW} and Z_{COL} are likely to be relatively high for row- and col-indices that correspond to any embedded low-rank submatrices, and relatively low for the other indices.

To see why this might be true, consider a random binary $M_D \times N$ D-matrix (with $+1/-1$ entries, as shown in Fig 20A) containing an embedded low-rank $m_B \times n$ bicluster B (tinted in pink). To aid discussion, let’s also assume that this embedded low-rank bicluster is perfectly rank-1 (i.e., with no noise). As seen in Fig 20A, this means that the rows and columns of the embedded bicluster are perfectly correlated; any pair of rows or columns are either equal or are negatives of one another. As we’ll see in a moment, this structure can be used to identify the bicluster.

To begin with we’ll look at 2×2 submatrices of this D -matrix, and for brevity we’ll refer to these 2×2 submatrices as ‘loops’. Each loop is described by two row-indices and two col-indices which, together, pick out four entries within the matrix D . Four loops are indicated in Fig 20A, each with a rectangle whose corners correspond to their 4 entries. Some loops either don’t pass through the bicluster at all, or have only 1 or 2 corners within the embedded bicluster (blue rectangles). Other loops are entirely contained within the bicluster (red rectangle).

⁴We discuss how exactly we delineate a bicluster later on in section 14.3.

Scatterplots of patients in covariate-space as algorithm proceeds: continuous-covariate correction



Density of patients in covariate-space as algorithm proceeds: continuous-covariate correction

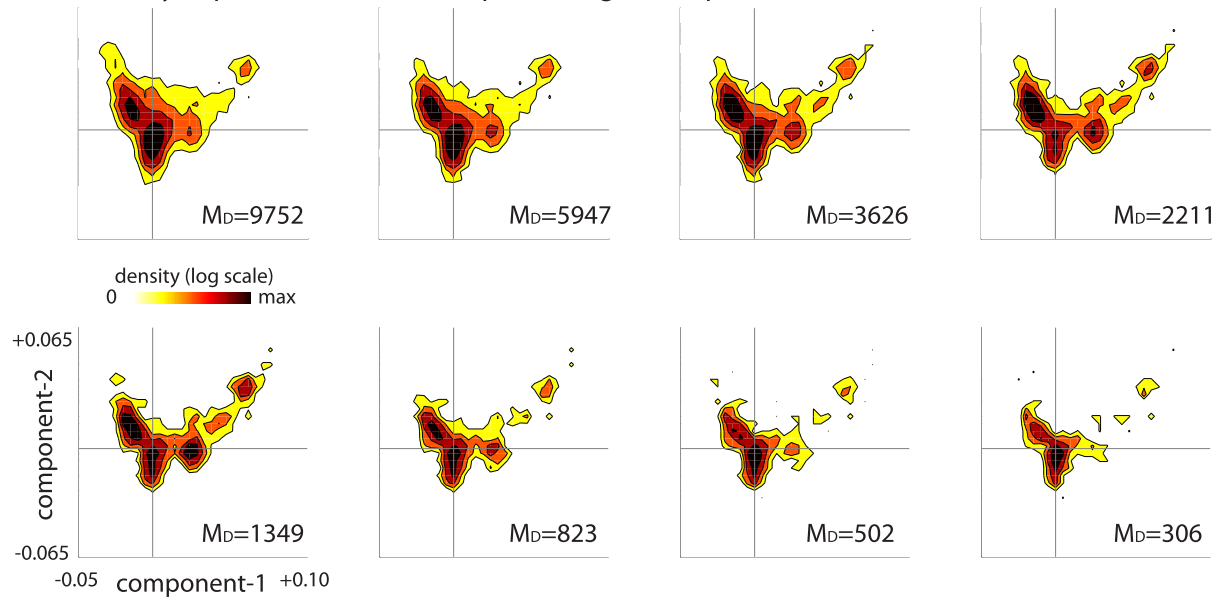
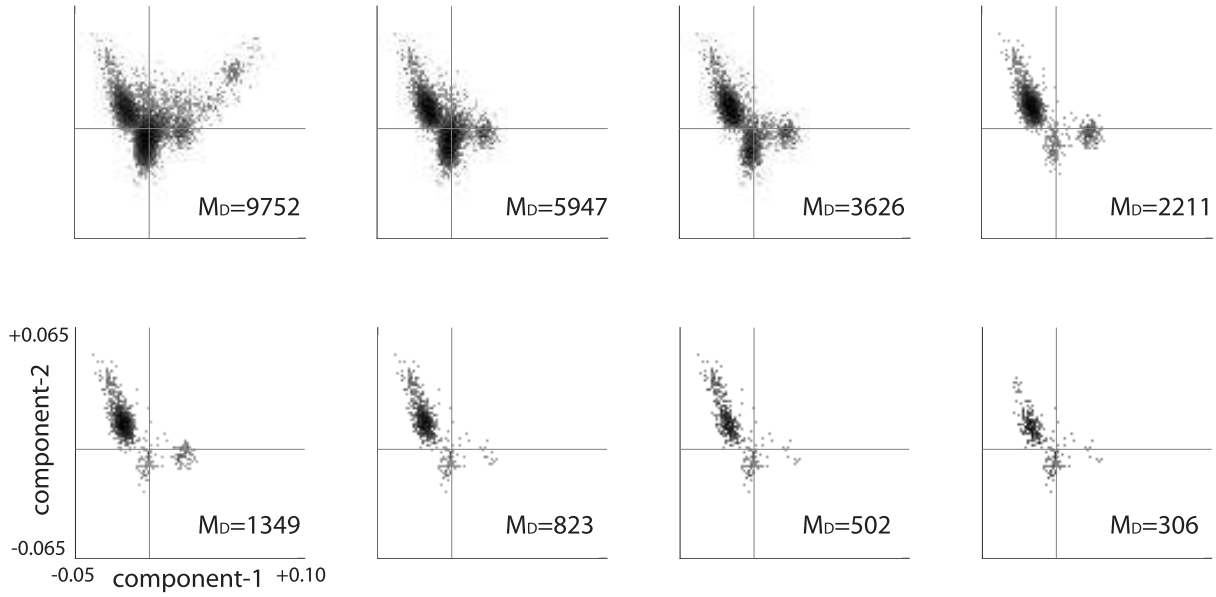


Figure 14: On top we show several scatterplots, sampling from different iterations as our algorithm proceeds. Each scatterplot illustrates the 2-dimensional joint-distribution of covariate-components across the remaining case-patients (i.e., the remaining M_D). The horizontal and vertical lines in each subplot indicate the medians of the components of the covariate-distribution. Below we show the same data again, except in contour form (note colorbar). Note that the covariate-distribution remains relatively well-distributed as the algorithm proceeds.

Scatterplots of patients in covariate-space as algorithm proceeds: No covariate correction



Density of patients in covariate-space as algorithm proceeds: No covariate correction

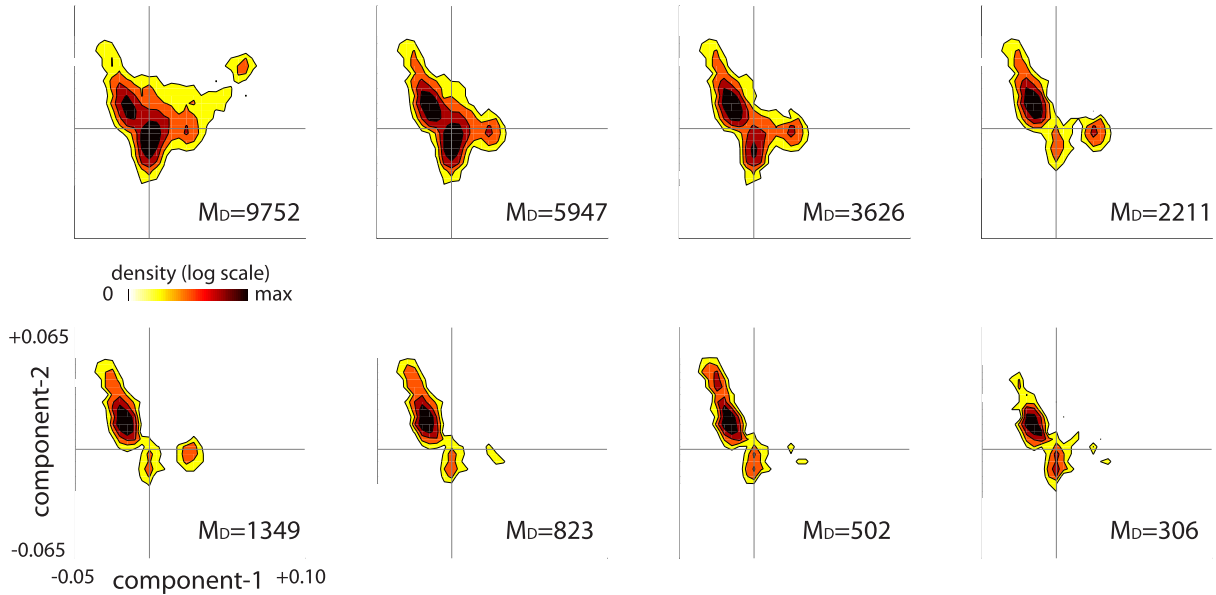


Figure 15: This figure has the same format as Fig 14, except that it shows the joint-distribution of continuous-covariates that would have resulted had we run our algorithm without correcting for continuous-covariates. Note that, in contrast to Fig 14, the covariate-distribution quickly becomes lopsided, involving mostly case-patients that are concentrated in a single quadrant of covariate-space.

Scatterplot and Density for bicluster shown in Example-2

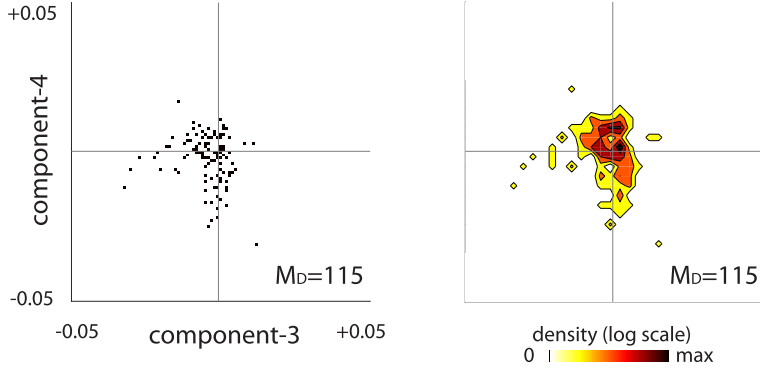


Figure 16: Joint-distribution of mds-components 3 and 4 for the bicluster shown in Example-3. This figure has the same format as Fig 13, with the exception that mds-components 3 and 4 are shown. Note that, even though we did not explicitly control for mds-components 3+4, they are relatively well distributed across the bicluster (see also Fig 17). A qualitatively similar story holds if we plot the joint-distribution of mds-components 1+3, 1+4, 2+3 or 2+4, and a similar trend holds for the later mds-components as well, which are even less structured than components 3 and 4.

The main observation that drives the success of our algorithm is that loops that are entirely contained within the embedded bicluster (such as the red loop) are guaranteed to be rank-1, whereas the other loops (such as the blue loops) are just as likely to be rank-2 as they are to be rank-1. Examples of rank-1 and rank-2 loops are shown in Fig 20B, with the row- and col-indices denoted by j, j' and k, k' , respectively (there are $2^4 = 16$ possibilities).

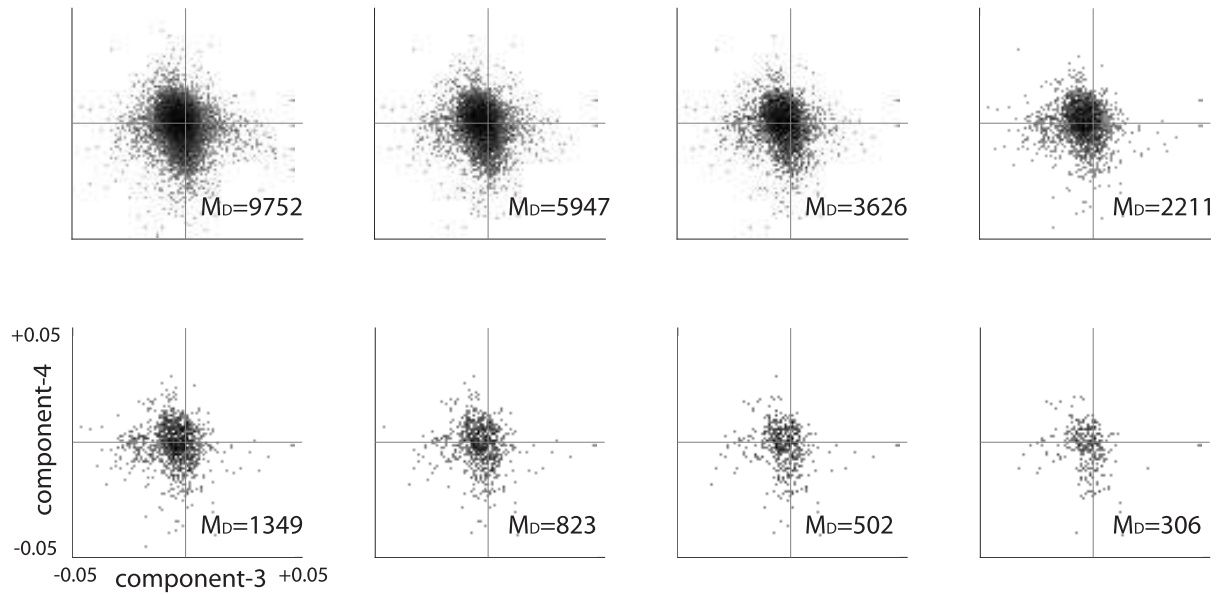
Given this observation, we can ascribe to each row-index j the following row-score $[Z_{\text{ROW}}]_j$. We consider all the loops traversing the given row- j and accumulate a sum; adding 1 for every rank-1 loop, and subtracting 1 for every rank-2 loop. If we consider a situation where j is a row that does not participate in the bicluster B , then $[Z_{\text{ROW}}]_j$ will sum over $(M_D - 1)N(N - 1)$ loops, roughly half of which will be rank-1 and half of which will be rank-2. This means that, when j does not participate in the bicluster, then the sum $[Z_{\text{ROW}}]_j$ will be roughly 0, with a standard-deviation (across the various j) close to $\sqrt{2}\sqrt{M_D N^2}$ (The factor of $\sqrt{2}$ arises because the loops are not all independent). On the other hand, when \tilde{j} is a row that does participate in the bicluster, then $[Z_{\text{ROW}}]_{\tilde{j}}$ will sum over $(m_B - 1)n(n - 1)$ loops that are fully contained within the bicluster B , and $(M_D - 1)N(N - 1) - (m_B - 1)n(n - 1)$ loops that straddle D as well as B . The loops within B will all be rank-1, and will collectively contribute a ‘signal’ of size $(m_B - 1)n(n - 1) \sim m_B n^2$ to $[Z_{\text{ROW}}]_{\tilde{j}}$. The loops that straddle D will be roughly half rank-1 and half rank-2, contributing an average of 0 to $[Z_{\text{ROW}}]_{\tilde{j}}$, with a ‘noise’ – or standard-deviation – of roughly $\sqrt{2}\sqrt{M_D N^2 - m_B n^2}$ (taken across the various \tilde{j}).

Based on these considerations, we expect that the collection of row-scores associated with rows outside of B will be distributed like a gaussian with a mean of 0 and a standard-deviation $\sim \sqrt{2}\sqrt{M_D N^2}$ (blue curve in Fig 20C). On the other hand, the collection of row-scores associated with rows inside B will be distributed like a gaussian with a mean of $\sim m_B n^2$ and a standard-deviation of at most $\sim \sqrt{2}\sqrt{M_D N^2}$ (red curve in Fig 20C). If $m_B n^2$ is comparable to or larger than $\sqrt{M_D N^2}$, then we expect the ‘signal’ to be somewhat distinguishable from the ‘noise’; the rows scores associated with the bicluster B should be significantly different from those associated with the rest of D .

Unfortunately, we usually don’t know which rows are which, and so we don’t see the blue- and red-curves shown in Fig 20C. Rather, after calculating all the row-scores we see something like Fig 20D. At first it might seem reasonable to guess that the rows corresponding to the highest scores are rows of B . While this statement is true when B is sufficiently large, it tends not to hold when B is much smaller than D . A better bet is to guess that the *lowest* row-scores are *not* from B . Quantitatively: assuming that $m_B n^2 \gtrsim \sqrt{M_D N^2}$, then the row with the lowest score is exponentially unlikely to be part of B .

Our strategy is built around this last observation: at every step we eliminate a few rows of D corresponding to the lowest row-scores. These eliminated rows are exponentially unlikely to come from B . We also do the same thing with the columns, eliminating the columns of D corresponding to the lowest col-scores (in this case the ‘signal’ associated with the columns of B is $\sim m_B^2 n$). After each such elimination, the row- and col-scores associated with the remaining rows and columns change, and so we recalculate the scores and repeat. Intuitively, we expect that, as we eliminate rows and columns, we are most likely to eliminate rows and columns of D , while leaving B untouched. Thus, we expect the ‘noise’ in our score-distribution to decrease (e.g., the noise associated with the row-scores is $\sim \sqrt{M_D N^2}$), whereas the ‘signal’ should remain relatively constant (e.g., the signal associated with the row-scores is $\sim m_B n^2$). This should result in the loop-scores of B becoming more and more distinguishable from the other loop-scores; the two distributions shown in Fig 20D should become narrower and narrower, while preserving their means. Put another way, as the algorithm progresses we expect the observed distribution of scores (shown in Fig 20D) to gradually evolve into a bimodal distribution with two

Scatterplots of patients in covariate-space as algorithm proceeds: continuous-covariate correction



Density of patients in covariate-space as algorithm proceeds: continuous-covariate correction

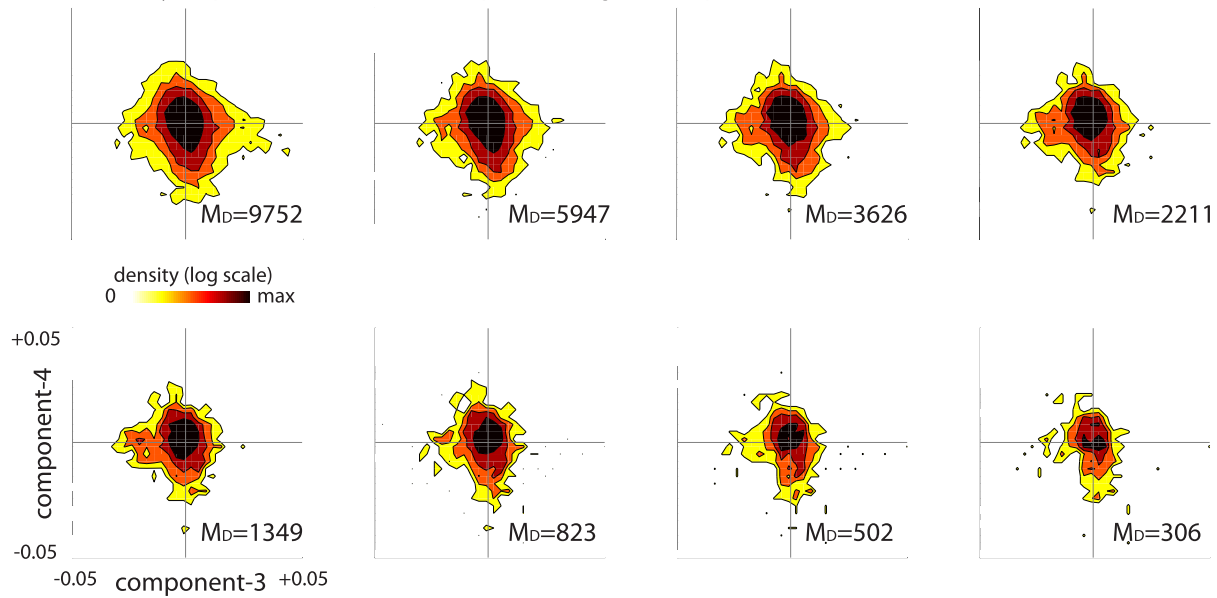


Figure 17: Joint-distribution of mds-components 3 and 4 as the algorithm proceeds. This figure has the same format as Fig 14, with the exception that mds-components 3 and 4 are shown. Note that these two mds-components are much less structured than the first two (shown in Fig 14). Even though we did not explicitly control for mds-components 3+4 when running our algorithm, these mds-components remain relatively well distributed as the algorithm proceeds. Just as in Fig 16, we see qualitatively similar results if we plot the joint-distribution of mds-components 1+3, 1+4, 2+3, 2+4, or any pair involving higher-order mds-components.

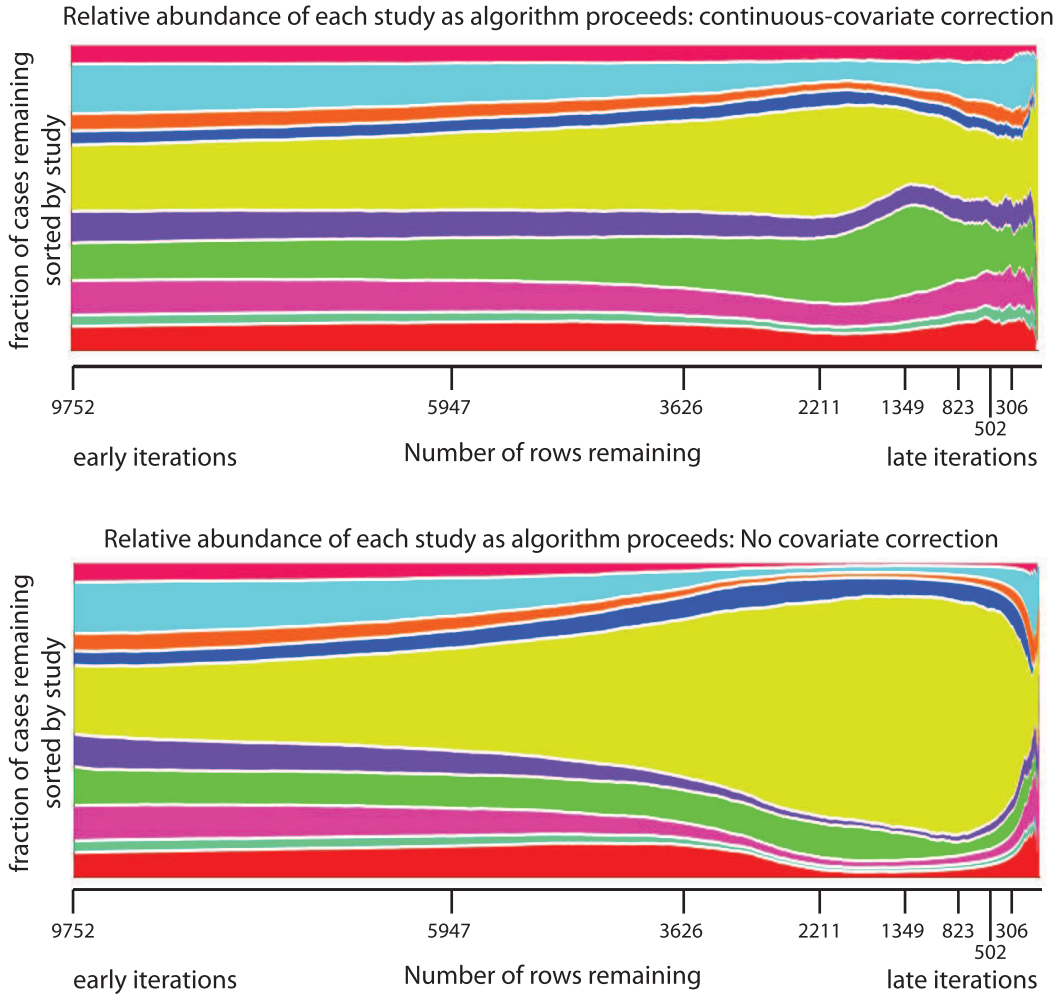


Figure 18: Study-distribution for Example-3. As mentioned previously, our algorithm proceeds iteratively, removing patients (i.e., rows) and alleles (i.e., columns) until none remain. The output of our algorithm includes a list of the case-patients in the order they were eliminated (with the patients that most strongly participate in the bicluster listed at the end). In this particular example each patient belongs to one of 10 studies, each collected by a different research group. It is important to ensure that any biclusters we find involve case-patients drawn from a reasonable mixture of studies. This is true for the Example shown here, even though we only explicitly corrected for continuous-covariates (and did not explicitly correct for study as a categorical-covariate). On top we display a stacked bar-graph illustrating the distribution of studies across the remaining case-patients as our algorithm proceeds. The vertical axis indicates study-fraction (different studies are colored differently), and the horizontal axis indicates the number of case-patients remaining. The iterations corresponding to the subplots in Fig 14 are indicated below the horizontal axis. Note that, for the most part, the remaining case-patients are relatively well-balanced across the 10 studies. The exact study-distribution for the 115 patients within the bicluster shown in Fig 11 is shown in Fig 19. On the bottom we show a similar stacked bar-graph for our algorithm run *without* covariate-correction of any kind. Note that – without covariate-correction – the latter half of the iterations are dominated by a few of the studies.

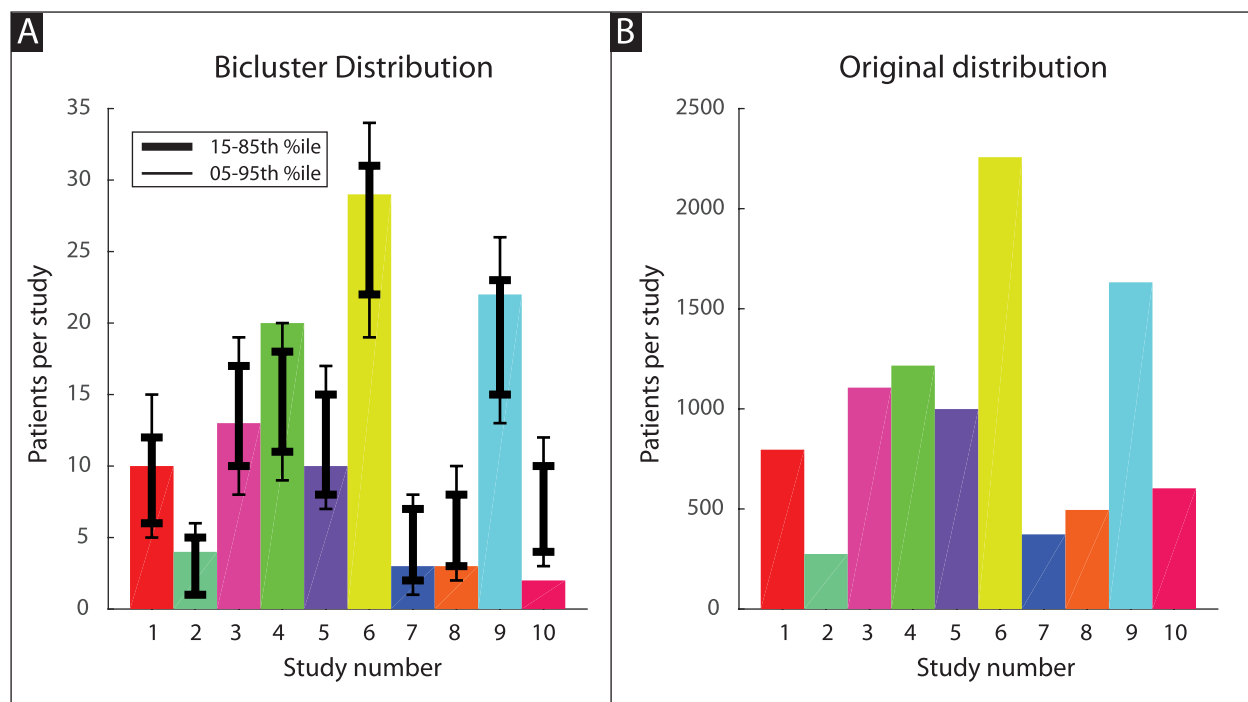


Figure 19: Study-distribution for the bicluster shown in Example-3. The particular bicluster shown in Example-3 was selected based on the peak of the z-score of the row-trace (see end of section 14.2). This bicluster involves 115 patients, distributed across the 10 studies mentioned in Fig 18. Even though we did not explicitly correct for study as a categorical-covariate, we did correct for continuous-covariates. As described above, since study is rather well correlated with the continuous-covariate, we expect that the bicluster we found should have a reasonable distribution across studies. This is indeed the case. In panel-A we show the study-distribution of the 115 patients within the bicluster. The color and ordering of each bar corresponds to the color and ordering of the 10 studies used in Fig 18. In panel-B we show the original study-distribution of the 9752 case-patients. Note that the study-distribution of the bicluster is not too different from the original study-distribution of the case-patients. To quantify this observation we drew multiple samples of 115 random patients from the original set, and measured the study-distribution of each random sample. Overlaid on panel-A we show the 5th, 15th, 85th and 95th percentiles (per study) for these randomly-sampled study-distributions. As can be seen, the study-distribution of the bicluster falls well within the bounds one might expect for a perfectly-balanced bicluster, with the possible exceptions of study 10 (which is underrepresented) and study 4 (which is slightly overrepresented).

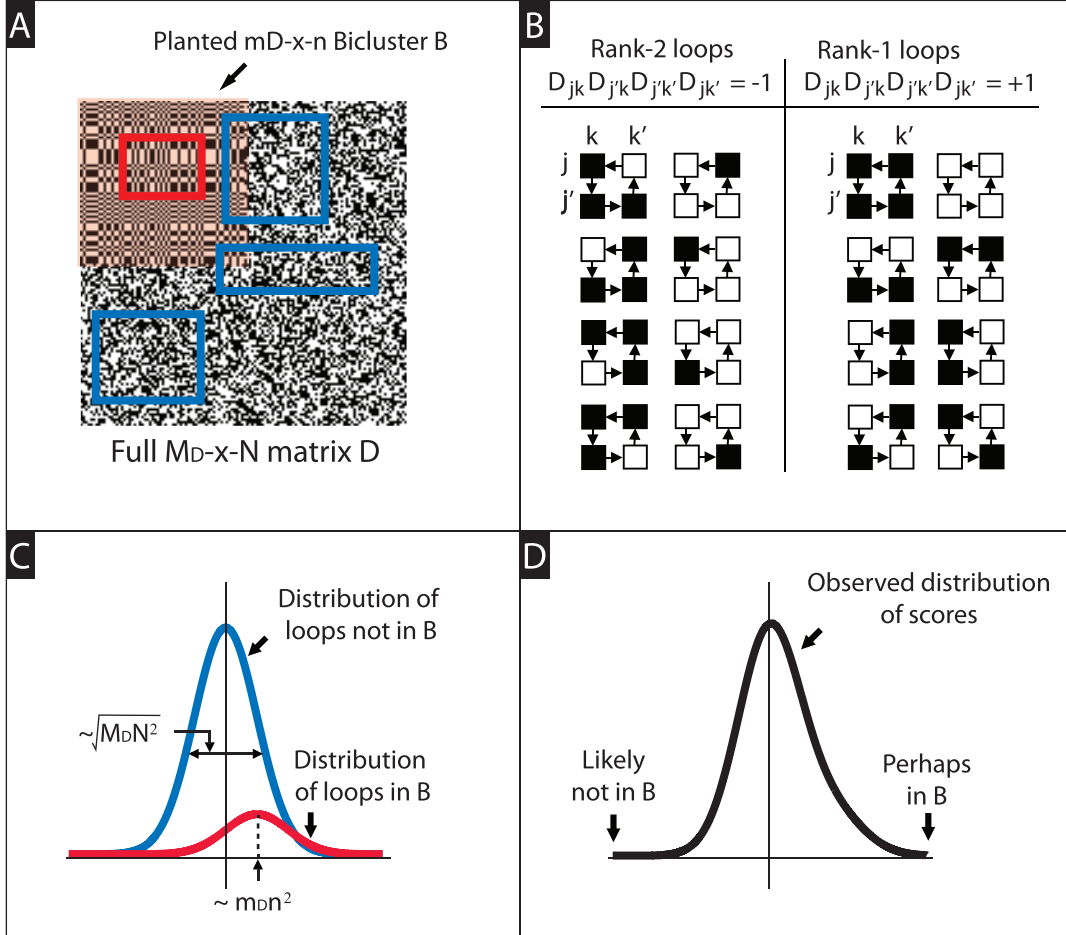


Figure 20: Illustration of the algorithm operating on a case-matrix alone (i.e., D only). In Panel-A we show a large $M \times N$ binarized matrix D (black and white pixels correspond to values of ± 1 , respectively). In the upper left corner of D we've inserted a large rank-1 bicluster B (shaded in pink). Our algorithm considers all 2×2 submatrices (i.e., 'loops') within D . Several such loops are highlighted via the blue rectangles (the corners of each rectangle pick out a 2×2 submatrix). Generally speaking, loops are equally likely to be rank-1 or rank-2. Some loops, such as the loop shown in red, are entirely contained within B . These loops are more likely to be rank-1 than rank-2. In Panel-B we show some examples of rank-2 and rank-1 loops. Given a loop with row-indices j, j' and column-indices k, k' , the rank of the loop is determined by the sign of $D_{jk}D_{j'k}D_{j'k'}D_{jk'}$. Our algorithm accumulates a 'loop-score' for each row j and each column k . In its simplest form, the loop-score for a particular row j is given by $\sum_{j', k, k'} D_{jk}D_{j'k}D_{j'k'}D_{jk'}$. Analogously, the loop-score for a column k is given by $[D^T D D^T D]_{kk}$. In Panel-C we show the distribution of loop-scores we might expect from the rows/cols of D that are not in B , whereas the red-curve corresponds to the distribution of scores expected from the rows/cols of B . In Panel-D we show the distribution of loop-scores we might expect by pooling all rows or columns of D . The rows or columns that correspond to the lowest scores are not likely to be part of B .

distinct peaks. While we don't have a proof of this phenomenon, our numerical experiments support this intuition.

3 Calculating the scores:

In terms of computation, the row-scores mentioned above can be computed as follows:

$$\begin{aligned}
[Z_{\text{ROW}}]_j &= \sum_{\substack{j \text{ fixed}, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k} D_{j'k'} D_{jk'} = \sum_{\substack{j \text{ fixed}, j' \neq j; \\ k' \neq k}} D_{jk} D_{kj'}^\top D_{j'k'} D_{k'j}^\top \\
&= \sum_{j \text{ fixed}, j', k', k} (1 - \delta_{jj'} - \delta_{kk'} + \delta_{jj'} \delta_{kk'}) D_{jk} D_{kj'}^\top D_{j'k'} D_{k'j}^\top \\
&= \sum_{j \text{ fixed}, j', k', k} D_{jk} D_{kj'}^\top D_{j'k'} D_{k'j}^\top - \sum_{j \text{ fixed}, k', k} D_{jk} D_{kj}^\top D_{j'k'} D_{k'j}^\top \\
&\quad - \sum_{j \text{ fixed}, j' k} D_{jk} D_{kj'}^\top D_{j'k} D_{k'j}^\top + \sum_{j \text{ fixed}, k} D_{jk} D_{kj}^\top D_{jk} D_{kj}^\top \\
&= [DD^\top DD^\top]_{j,j} - N^2 - M_D N + N \\
&= [DD^\top DD^\top]_{j,j} - N(M_D + N - 1).
\end{aligned}$$

The column-scores can be computed similarly, reducing to

$$[Z_{\text{COL}}]_k = [D^\top DD^\top D]_{k,k} - M_D(N + M_D - 1).$$

Note that these scores can be calculated using only a few matrix-matrix multiplications.

The final constant on the right-hand-side above is chosen so that the left-hand-side excludes the contribution of all the loops that only travel across a single row- or column-index. If we were to draw these loops in Fig 20, they would describe line-segments (or points) rather than full rectangles. These 'collapsed' loops do not have the distribution of ranks that we discussed above – indeed, they are all rank-1. There are $N(M_D + N - 1)$ of these collapsed-loops for each row, and $M_D(N + M_D - 1)$ for each column. If we were to include these collapsed-loops in our score, they would each add a '+1' to their respective totals; the mean score of any row- or column- would then include the contribution from all these collapsed-loops.

Within the context of our D -only algorithm described in section 2, we don't actually need to exclude these collapsed-loops. Specifically, the collapsed-loops contribute the exact same quantity $N(M_D + N - 1)$ to each row-score in the matrix, regardless of whether or not that row is part of a bicluster or not. A similar story holds for the columns; the collapsed-loops contribute $M_D(N + M_D - 1)$ to each column-score. Our algorithm only considers the ranking of row- and column-scores, the identity of the removed rows and columns will be the same whether or not we exclude the collapsed-loops.

Nevertheless, we still typically exclude the collapsed-loops – removing their contribution from the row- and column-scores. The reason we do this is because, later on, we'll consider more complicated situations involving controls and covariates. In these more complicated situations it is convenient to exclude collapsed-loops so that, in the absence of biclusters, the mean score of each row- and column will be 0.

While not necessary at this point, we'll later rescale these scores so that they range from negative one to positive one:

$$\begin{aligned}
[Z_{\text{ROW}}]_j &= \frac{1}{(M-1)N(N-1)} \sum_{\substack{j \text{ fixed}, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k} D_{j'k'} D_{jk'}, \text{ and} \\
[Z_{\text{COL}}]_k &= \frac{1}{(N-1)M(M-1)} \sum_{\substack{j' \neq j; \\ k \text{ fixed}, k' \neq k}} D_{jk} D_{jk'} D_{j'k'} D_{j'k}.
\end{aligned}$$

When considering large random matrices, these rescaled row- and column-scores will have means of 0 and variances of roughly $2/(MN^2)$ and $2/(M^2N)$, respectively. To represent these rescaled scores more easily, we'll later introduce the notation of a 'rescaled sum' $\widetilde{\Sigma}$ which implicitly includes a prefactor equal to the number of summands:

$$\begin{aligned}
[Z_{\text{ROW}}]_j &= \widetilde{\sum}_{\substack{j \text{ fixed}, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k} D_{j'k'} D_{jk'}, \text{ and} \\
[Z_{\text{COL}}]_k &= \widetilde{\sum}_{\substack{j' \neq j; \\ k \text{ fixed}, k' \neq k}} D_{jk} D_{jk'} D_{j'k'} D_{j'k}.
\end{aligned}$$

3.1 Updating the scores using a low-rank update:

When removing a row or column the row- and col-scores change. In the simplest implementation the new row- and col-scores can be recalculated from scratch. However, it is often more efficient to simply update the older scores via a low-rank update. To see why this might be true, let's assume that D has the block structure

$$D = \begin{bmatrix} E & c \\ r & d \end{bmatrix},$$

where E is size $M_E \times N_E$, c is $M_E \times N_c$, r is $M_r \times N_E$, and d is $M_r \times N_c$. We'll also assume that we have already calculated the row-scores of D , and plan on eliminating r , c and d from D , leaving E remaining. We would like to determine how the row-scores of E depend on those of D , as well as on the eliminated submatrices r , c and d . This relationship is derived mainly from the following formula (note that, for the convenience of presentation, we have assumed that the submatrix E spans the first M_E rows of D):

$$[DD^T DD^T]_{j,j} = [EE^T EE^T]_{j,j} + [EE^T cc^T + cc^T EE^T + cc^T cc^T + (Er^T + cd^T)(rE^T + dc^T)]_{j,j} \text{ for } j = 1, \dots, M_E.$$

Consequently, the row-scores for E can be calculated using those from D along with an update involving only matrix-vector multiplications such as $c^T E$, $(c^T E) E^T$, Er^T , and so forth. While forming the row-scores for D in the first place may require $O(M_D N \min(M_D, N))$ operation, updating those scores to form the row-scores of E requires only $O(M_E N_E)$ operations. These updates are performed in the matlab source-code included in the supplementary tutorial 'S1 Source Code', as well as in the c source-code included in the supplementary material 'S2 Source Code' (both of which also use similar observations to update the column-scores).

4 Interpreting the scores:

There are many different ways to interpret our row-scores and column-scores, and we mention some here.

Sum over loops: As discussed above, our scores $[Z_{\text{ROW}}]_j$ and $[Z_{\text{COL}}]_k$ tally the ranks of the loops associated with row- j and column- k of D , respectively.

Signal: These scores serve as an accumulation of 'signals' driven by the various structured elements within the data-matrix. As we'll discuss in section 5, a large random $M \times N$ data-matrix D will give rise to row- and column-scores that have a mean of 0. If the matrix D were to contain an $m \times n$ low-rank bicluster B involving row-subset J_B and column-subset K_B , then the presence of B will impact the scores of J_B and K_B . Depending on how noisy it is, B will add a 'signal' of $\sim mn^2$ to the row-scores of J_B , and $\sim m^2 n$ to the column-scores of K_B . If D were to contain multiple biclusters, then each would add their signals to the scores of their row- and column-subsets. Conversely, the score of any row or column in D will reflect the accumulation of signals it receives from all the biclusters it participates in. By eliminating the rows and columns with the smallest scores, we are eliminating those rows and columns that have the smallest signal. This removes rows and columns which either belong to no biclusters or belong to the fewest/weakest/smallest biclusters, allowing our algorithm to focus on the remaining rows and columns that have a strong signal. The ramifications of this iteration are discussed more in section 7.3.

Correlation: Our scores can also be interpreted as a measure of correlation. If we don't correct for collapsed-loops, the row- and column-scores are given by the diagonal entries of $DD^T DD^T$ and $D^T DD^T D$, respectively. The row-score $[Z_{\text{ROW}}]_j$ will then be proportional to the mean-squared correlation between row- j and all the rows of the case-matrix D . Similarly, the column-score $[Z_{\text{COL}}]_k$ is proportional to the mean-squared correlation between column- k and all the columns of D . By eliminating the rows and columns with the lowest scores we are eliminating the rows and columns that are least correlated with the rest, allowing the algorithm to focus on those remaining. As we describe in sections 5,6 and 7, this process facilitates the discovery of any tightly correlated subsets of rows and columns.

Trace: If we don't correct for collapsed-loops, the average row-score across all the rows in D will equal the average column-score across all the columns in D . Both averages will be proportional to the trace of $DD^T DD^T$, which is equal to $\sum \sigma_j^4$, where $\{\sigma_1, \dots, \sigma_{\min(M,N)}\}$ are the singular-values of D . This is equal to the nuclear norm of $DD^T DD^T$, which will be high if D is low-rank (see, e.g., [M35] for a discussion of the nuclear norm, and its applications to clustering and biclustering).

Rayleigh-Quotient: Again, if we don't correct for collapsed-loops, the row-score $[Z_{\text{ROW}}]_j$ will be proportional to the rayleigh-quotient $\bar{x}^T D^T D \bar{x}$, where $\bar{x}^T = D_{j,:}$ is the j^{th} -row of D . The column-score $[Z_{\text{COL}}]_k$ will be proportional to $\bar{y}^T DD^T \bar{y}$, where $\bar{y} = D_{:,k}$ is the k^{th} -column of D . Our algorithm attempts to focus on those rows and columns which

correspond to large rayleigh-quotients; i.e., to those rows and columns which are most parallel to the dominant eigenvectors of $D^\top D$ and DD^\top , respectively. This perspective is discussed in more detail in [M22] and [M40] and in section 22.

5 Generalization to noisy biclusters

The discussion above was prefaced by the rather idealistic assumption that the embedded bicluster B was perfectly rank-1; i.e., that all the rows and columns of B were perfectly correlated with one another. In practice, of course, the situation is far messier. Any hidden low-rank bicluster B will be noisy; B won't be exactly rank-1, and its rows and columns will be imperfectly correlated.

It turns out that our methodology above still works in these messier situations. The main reason is that, even when B is noisy (and even when B isn't exactly rank-1 itself), the loops within a binarized version of B are still more likely to be rank-1 than to be rank-2. While this probability 'g' won't be 100% (as in the ideal case above), it will still be significantly greater than 50%, provided that the numerical rank of B is sufficiently small and the singular-values of B decay sufficiently quickly. Under these conditions (which we'll quantify momentarily), B will still have a 'signal' distinguishing it from the rest of D . For example, the row-scores associated with B will be drawn from a distribution with a mean that, while lower than the ideal $m_B n^2$ above, will be $m_B n^2 (2g - 1)$, which is still on the same order as $m_B n^2$. Similarly, the col-scores of B will be drawn from a distribution with a mean of $m_B^2 n (2g - 1)$, which is on the same order as $m_B^2 n$.

Now we turn to a discussion of just how noisy B can be before our algorithm fails to detect it. This discussion is essentially the same as that in [M39]. To start with we imagine that $m_B < n$, and that each column of $B \in \mathbb{R}^{m_B \times n}$ is drawn from a distribution ρ in \mathbb{R}^{m_B} which is constructed as follows. We fix a small number $l \lesssim 4$ and take ρ to be a randomly oriented multivariate gaussian distribution which has l large principal values equal to 1, and the remaining principal values equal to $\varepsilon < 1$ (see Fig 21A). This gaussian-distribution ρ can be thought of as a distribution where a significant fraction of its variance is accounted for by its first l principal components. Specifically, this fraction is $l / (l + \varepsilon^2 (m_B - l)) \sim l / (l + \varepsilon^2 m_B)$, which will be large as long as $\varepsilon^2 m_B$ is small. This construction of B then implies that B will be of numerical rank l , with a 'fuzziness' of ε . The contours of ρ look like ellipsoids, with eccentricity determined by $1/\varepsilon$. For example, if ε were to be 0, then ρ would be compressed to an l -dimensional hyperplane, and B would be exactly rank- l (i.e., very eccentric). If, on the other hand, ε were to be 1, then ρ would be a uniform gaussian and B would not be low-rank at all (i.e., not eccentric). For most typical applications in gene-expression analysis we can think of a fuzziness ε on the order of $\sim 1/30$ to $1/3$, and m on the order of 10 to 100. For applications in GWAS m may be a little larger, ranging up to 1000 or so.

What we'll endeavor to show below is that, when l and ε are sufficiently small, then the distribution of loops drawn from B will be different than the distribution of loops drawn from the rest of D . Specifically, we'll quantify the probability g_{l,ε,m_B} that a randomly chosen loop drawn from B will – after binarization – be rank-1 rather than rank-2. As we'll see, a soft 'threshold' will be crossed when $\varepsilon \gtrsim 1/\sqrt{m_B}$, at which point the bicluster B is essentially uncorrelated, g_{l,ε,m_B} will be essentially 1/2, loops of B will look like loops of D , and our algorithm will fail. Looking back at our definition of ρ , this threshold is natural, for when $\varepsilon \gtrsim 1/\sqrt{m_B}$ the distribution ρ is no longer well captured by its first l principal components. For most typical applications, we can expect $\varepsilon\sqrt{m}$ to be on the order of ~ 0.1 to 1. These typical biclusters will be mostly correlated (i.e., they will be well approximated by their first principal component), but this correlation will not necessarily be exhibited in each and every entry (e.g., see Fig 22).

Now let us consider an arbitrary 2-by-2 submatrix (i.e., loop) within such an ε -fuzzy rank- l matrix B . This loop spans row-indices j, j' and col-indices k, k' . This loop can be described by first drawing two m_B -dimensional vectors from ρ (i.e., two columns k and k' from B), and then choosing two coefficients to read from (i.e., two row-indices j and j'). Based on our assumptions, such a loop is drawn from a distribution constructed in two steps: the first step involves sampling two vectors from ρ , whereas the second step involves projecting those two vectors onto the plane spanned by 2 randomly-chosen coordinate-axes.

Now recall that we defined B in such a way that the columns of B were drawn from a randomly-oriented gaussian-distribution ρ . The distribution obtained by (i) starting with a randomly-oriented gaussian and then projecting onto a fixed plane is the same as the distribution obtained by (ii) starting with a fixed gaussian, and then projecting onto a randomly oriented plane. Thus, without loss of generality, we can assume that loops of B are drawn in the following manner (where we have used $m = m_B$ for brevity):

1. First define an eccentric gaussian-distribution ρ on \mathbb{R}^m which has l principal-values equal to 1 and the remaining principal values equal to $\varepsilon < 1$. Then orient ρ so that the first l principal components align with the first l coordinate axes (and the remaining $(m - l)$ principal-components align with the other coordinate axes). This ρ is a fixed distribution that depends only on l , ε and m .
2. Now select a uniformly-distributed randomly-oriented planar orthogonal-projection $P^{2 \leftarrow m} : \mathbb{R}^m \rightarrow \mathbb{R}^2$, and define the projected distribution $\tilde{\rho} = P^{2 \leftarrow m} \rho$ as a distribution on \mathbb{R}^2 . Note that, as ρ was a multivariate gaussian, so too

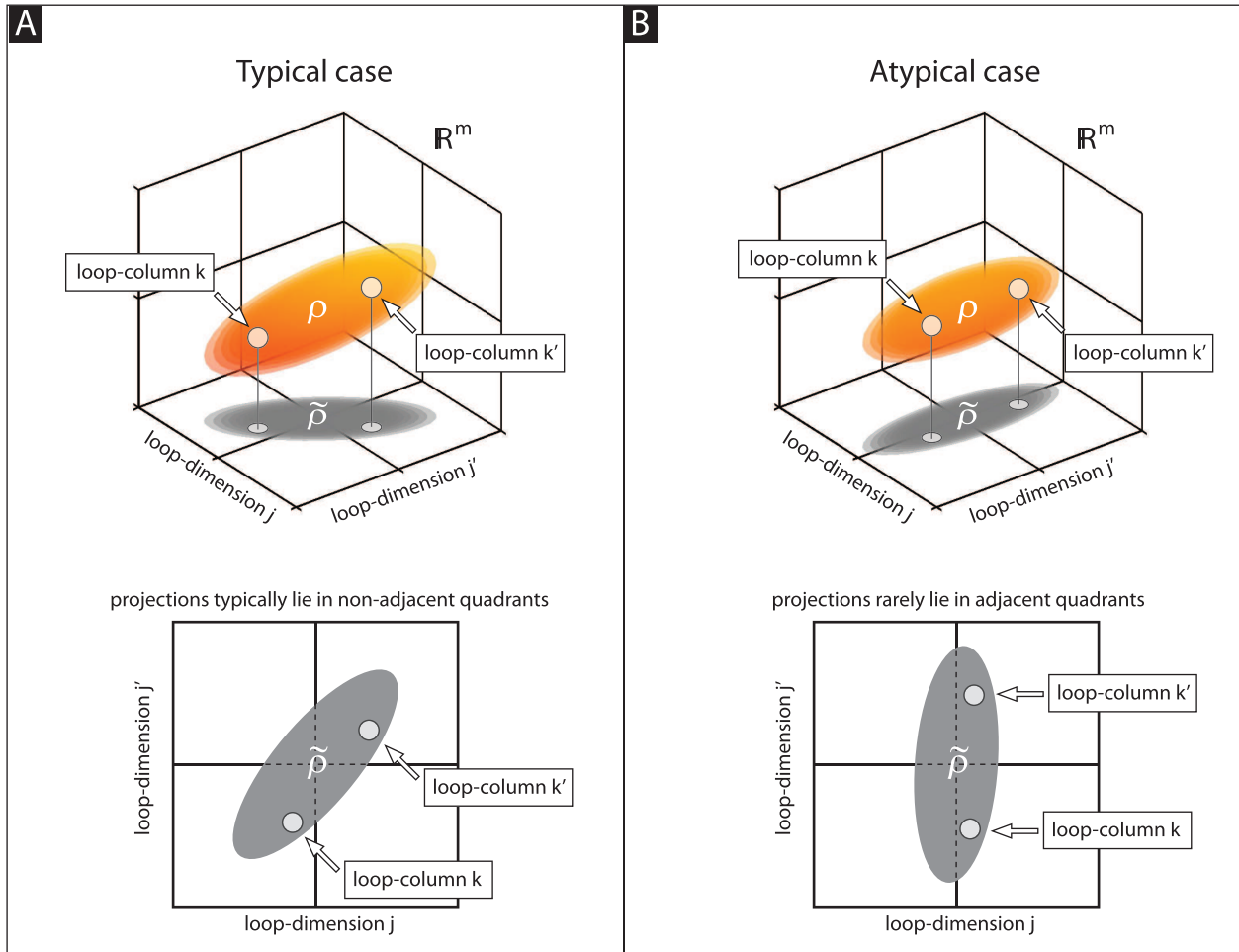


Figure 21: This figure illustrates the main geometric feature underlying the success of our algorithm: namely, a 2-dimensional projection of a randomly-oriented eccentric gaussian-distribution is typically concentrated in non-adjacent quadrants (see discussion in section 5). To set the stage, we sketch in Panel-A a randomly-oriented eccentric gaussian distribution ρ in \mathbb{R}^m (orange). Any loop drawn from such a distribution involves first projecting ρ onto a 2 dimensional distribution $\tilde{\rho}$, and then sampling two points at random from $\tilde{\rho}$. The row-indices j and j' of the loop determine the range of the projection, while the column indices k and k' label the random samples (indicated here with two circles). After binarization, the rank of such a loop is determined by where its two samples lie in the plane. On one hand, if the two samples lie in the same or opposite quadrants, then the loop will be rank-1; this is the typical case, illustrated in panel-A. On the other hand, if the two samples lie in adjacent quadrants, then the loop will be rank-2; this can occur almost half the time if the major axis of $\tilde{\rho}$ happens to be closely aligned with a coordinate-axes – one such example is illustrated in panel-B. As we describe in section 5, case-A is more likely than case-B, and this probability increases significantly as ρ becomes more eccentric. Later we'll quantify this probability with a value gl_{ϵ, m_B} , which measures the probability that a loop drawn from ρ will be rank-1 (rather than rank-2).

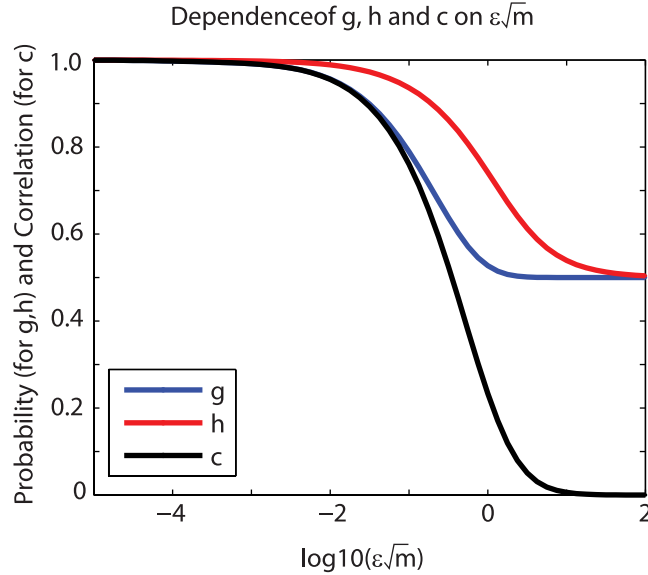


Figure 22: Dependence of $g_{1,\varepsilon\sqrt{m}} := g_{1,\varepsilon,m}$ (blue), $h_{1,\varepsilon\sqrt{m}}$ (red) and $c_{1,\varepsilon\sqrt{m}}$ (black) on $\varepsilon\sqrt{m}$. Given a randomly oriented rank-1 bicluster with spectral noise ε and size $m \times m$, the quantity $g_{1,\varepsilon\sqrt{m}}$ is the probability that a randomly chosen ‘loop’ will be rank-1 rather than rank-2 after binarization. The quantity $h_{1,\varepsilon\sqrt{m}}$ is the probability that a randomly chosen entry will have a sign opposite to that predicted by the rank-1 approximation to the bicluster. The quantity $c_{1,\varepsilon\sqrt{m}}$ is the expected absolute-value of the cross-correlation between different rows (or columns) within the bicluster.

will be $\tilde{\rho}$, and the contours of $\tilde{\rho}$ will look like ellipses on \mathbb{R}^2 . However, unlike ρ , the distribution $\tilde{\rho}$ is not fixed, and will itself be drawn from a distribution (denoted by the measure $d\tilde{\rho}$, below).

3. Once $P^{2\leftarrow m}$ has been selected and $\tilde{\rho}$ has been determined, a loop of B will correspond to two 2-dimensional vectors, say $v_1, v_2 \in \mathbb{R}^2$, each drawn independently from $\tilde{\rho}$.

Binarization of B will send each of v_1, v_2 to one of the four corners of the unit-square in \mathbb{R}^2 , determined by which quadrant of \mathbb{R}^2 the vector lies in. Thus, after binarization, the loop of B will be rank-1 if v_1 and v_2 happened to lie in the same quadrant (i.e., if, after binarization, v_1 and v_2 fall onto the same corner of the unit-square). Similarly, v_1 and v_2 will correspond to a rank-1 loop if they happened to lie in opposite quadrants (thus falling onto opposite corners of the unit-square). Only when v_1 and v_2 fall into adjacent quadrants (i.e., adjacent corners of the unit-square) will they correspond to a rank-2 loop. Finally, we note that since $\tilde{\rho}$ is a multivariate gaussian in \mathbb{R}^2 , the fraction of $\tilde{\rho}$ contained within any quadrant must be the same as the fraction of $\tilde{\rho}$ contained within the opposite quadrant (e.g., the fraction of $\tilde{\rho}$ within the first-quadrant equals the fraction of $\tilde{\rho}$ within the third-quadrant). Summarizing all of these observations, we can state that, if $p\{\tilde{\rho}\}$ is the fraction of $\tilde{\rho}$ which is restricted to the first- and third-quadrants, then the fraction of $\tilde{\rho}$ restricted to the second- and fourth-quadrants is $1 - p$. Consequently, the probability that the binarized loop of B will be rank-1 is simply $p^2 + (1 - p)^2$, or $1 - 2p + 2p^2$. Thus, we can see that

$$g_{l,\varepsilon,m} = \int [1 - 2p\{\tilde{\rho}\} + 2p^2\{\tilde{\rho}\}] d\tilde{\rho},$$

where $d\tilde{\rho}$ is the measure on the distributions $\tilde{\rho}$ that one would obtain by following the prescription above.

At this point we’ll embark on a description of $d\tilde{\rho}$, the measure from which $\tilde{\rho}$ is drawn. To simplify our notation, we will use $\rho_\sigma(x)$ to denote the normal distribution $N(0, \sigma^2)$ on x . In addition, we’ll use $\rho_{a,b,\theta}(\vec{x})$ to denote the mean 0 anisotropic multivariate gaussian distribution in 2-dimensions with variances a^2 and b^2 , and first principal-component oriented at angle θ (see section 16 for details).

The randomly chosen projection $P^{2\leftarrow m}$ can be determined by drawing two randomly chosen orthonormal vectors $u^x, u^y \in \mathbb{R}^m$, and then constructing $P^{2\leftarrow m} = [u^x, u^y]^\top$. Drawing u^x and u^y in this way would be slightly nontrivial, since we would need to constrain u^x to be \perp to u^y . However, for sufficiently large $m \gg l$, we can approximate $P^{2\leftarrow m}$ just by drawing two random vectors $[w^x, w^y]^\top$ instead, where each entry of w^x and w^y is drawn independently from $\rho_{\sqrt{1/m}}$. These new vectors w^x and w^y won’t be exactly length 1, nor exactly \perp to one another. Nevertheless, they will typically be close to orthonormal (i.e., orthonormal to $O(1/\sqrt{m})$), which will be sufficient for our purposes. This kind of replacement – i.e., using w^x, w^y instead of u^x, u^y – will incur a small error, but will allow us to easily construct an approximation to $\tilde{\rho}$ which will be valid when $m \gg l$.

Using our notation above, we can see that $\tilde{\rho}$ is well approximated by:

$$\tilde{\rho} \sim [w^x, w^y]^\top \rho = \sum_{j=1}^l \begin{bmatrix} w_j^x \\ w_j^y \end{bmatrix} x_j + \sum_{j=l+1}^m \begin{bmatrix} w_j^x \\ w_j^y \end{bmatrix} x_j,$$

$$\text{or equivalently, } \tilde{\rho} \sim \sum_{j=1}^l \hat{w}_j x_j + \sum_{j=l+1}^m \hat{w}_j x_j.$$

In the expression above each of the \hat{w}_j represents a 2-element vector $[w_j^x, w_j^y]^\top$, and each of the x_j are drawn from the fixed version of ρ described above; that is to say: x_1, \dots, x_l are each drawn independently from $\rho_1(x)$, and x_{l+1}, \dots, x_m are each drawn independently from $\rho_\varepsilon(x)$. Each of the terms $\hat{w}_j x_j$ is a distribution in \mathbb{R}^2 , but one that is compressed onto the line running through \hat{w}_j . Because of how we constructed the w^x, w^y , each pair \hat{w}_j is drawn from $\rho_{\sqrt{1/m}, \sqrt{1/m}, 0}$. Since this latter distribution is isotropic, we can deduce that the orientation θ_j of the vector \hat{w}_j is uniformly distributed in $[0, 2\pi]$. Similarly, we can deduce that the magnitude r_j is drawn from the Rayleigh distribution $mr \exp(-mr^2/2)$, which can also be written as $r\sqrt{2\pi m} \rho_{\sqrt{1/m}}(r)$ if we use our shorthand for a gaussian distribution (see section 16.4). This means that, when $j \leq l$, each of the terms $\hat{w}_j x_j$ is represented by some $\rho_{r_j, 0, \theta_j}$. Consequently, the sum of any two $\hat{w}_j x_j + \hat{w}_{j'} x_{j'}$ will just be drawn from a distribution which is the convolution of the distributions from which the individual $\hat{w}_j x_j$ and $\hat{w}_{j'} x_{j'}$ are drawn. The latter sum $\sum_{j=l+1}^m \hat{w}_j x_j$, on the other hand, represents what is essentially a random 2-d projection of a uniform ε^2 -variance gaussian distribution on \mathbb{R}^m (since $m \gg l$). Such a projection will again be a uniform ε^2 -variance gaussian distribution, looking like $\rho_{\varepsilon, \varepsilon, 0}$.

Combining all these observations, $\tilde{\rho}$ can be rewritten as:

$$\tilde{\rho} \sim \{\rho_{r_1, 0, \theta_1} \star \dots \star \rho_{r_l, 0, \theta_l}\} \star \rho_{\varepsilon, \varepsilon, 0},$$

where each r_1, \dots, r_l is drawn independently from $r\sqrt{2\pi m} \rho_{\sqrt{1/m}}(r)$, and each $\theta_1, \dots, \theta_l$ is drawn independently from the uniform distribution on $[0, 2\pi]$. The multiple convolution $\{\rho_{r_1, 0, \theta_1} \star \dots \star \rho_{r_l, 0, \theta_l}\}$ is again a multivariate gaussian $\rho_{\alpha, \beta, \omega}$ for some α, β, ω . As a result, the distribution $\tilde{\rho}$ can be approximated by:

$$\tilde{\rho} \sim \rho_{\alpha, \beta, \omega} \star \rho_{\varepsilon, \varepsilon, 0}$$

for some (yet undetermined) α, β, ω . This expression is simply a uniformly- ε -mollified version of the eccentric gaussian $\rho_{\alpha, \beta, \omega}$, which is merely a slightly less eccentric gaussian with the same orientation. That is to say,

$$\tilde{\rho} \sim \rho_{\sqrt{\alpha^2 + \varepsilon^2}, \sqrt{\beta^2 + \varepsilon^2}, \omega}.$$

Given this representation of $\tilde{\rho}$, a simple calculation shows that the fraction $p\{\tilde{\rho}\}$ of $\tilde{\rho}$ which is restricted to the first- and third-quadrants is:

$$p\{\tilde{\rho}\} = \frac{1}{\pi} \operatorname{arccot} \left(\left(\sqrt{\frac{\beta^2 + \varepsilon^2}{\alpha^2 + \varepsilon^2}} - \sqrt{\frac{\alpha^2 + \varepsilon^2}{\beta^2 + \varepsilon^2}} \right) \frac{1}{2} \sin 2\omega \right).$$

While precisely determining α, β, ω is tedious when $l > 1$, one can observe that the principal-values α, β of

$$\rho_{\alpha, \beta, \omega} = \{\rho_{r_1, 0, \theta_1} \star \dots \star \rho_{r_l, 0, \theta_l}\}$$

depend on m in a simple way: both α and β scale with $1/\sqrt{m}$. As a result, $p\{\tilde{\rho}\}$ depends only on the combination $\varepsilon\sqrt{m}$, and not on ε or \sqrt{m} independently. Because $g_{l, \varepsilon, m}$ is also a function of $\tilde{\rho}$ we expect that, for fixed l and sufficiently large m , the probability $g_{l, \varepsilon, m}$ will also be a function of $\varepsilon\sqrt{m}$.

At this point we can write down the probability $g_{l, \varepsilon, m}$ for the various l . In the case $l = 1$, $\rho_{\alpha, \beta, \omega}$ is trivially $\rho_{\alpha, \beta, \omega} = \rho_{r_1, 0, \theta_1}$. This means that the probability $g_{1, \varepsilon, m}$ can be expressed as:

$$g_{1, \varepsilon, m} \sim \int_{r_1=0}^{r_1=\infty} \int_0^{2\pi} \frac{1}{2\pi} \left[1 - 2p\left(\theta_1, \frac{\varepsilon\sqrt{m}}{r_1}\right) + 2p\left(\theta_1, \frac{\varepsilon\sqrt{m}}{r_1}\right)^2 \right] r_1 \exp\left(\frac{-r_1^2}{2}\right) d\theta_1 dr_1, \quad (1)$$

$$\text{with } p(\omega, \delta) = \frac{1}{\pi} \arctan \left(\frac{\delta\sqrt{1 + \delta^2}}{-\frac{1}{2} \sin 2\omega} \right).$$

For the case $l > 1$ we can write out formulae for $g_{l, \varepsilon, m}$ via induction using Eq. 2 in section 16. This expression for $g_{l, \varepsilon, m}$ is very accurate even for moderate values of $m \gtrsim 64$. Numerical experiments confirming the accuracy of our approximation

for $g_{l,\varepsilon,m}$ are shown in Fig 23. As can be seen from these figures, when $l \lesssim 5$, the value of $g_{l,\varepsilon,m}$ is significantly greater than $1/2$ so long as $\varepsilon\sqrt{m} \lesssim 1$.

To summarize: When considering the bicluster B , the row-scores corresponding to B will take on values of $\sim mn^2(2g_{l,\varepsilon,m} - 1)$ on average. If we hope to detect B , then these average scores should be on the same order as (or larger than) the standard-deviation of the remaining scores associated with rows and columns of D that are not in B . That is to say, we would expect to detect B when $mn^2(2g_{l,\varepsilon,m} - 1) \sim \sqrt{2MN^2}$. In principle this implies that we should be able to detect B for arbitrarily low values of $g_{l,\varepsilon,m}$, so long as m, n are sufficiently large. However, in practice, this kind of limit is not typically achieved. For most real gene-expression data-sets we should expect $m := m_B$ to be no larger than ~ 32 to 100 , (with M_D no larger than ~ 1000 to $10,000$), implying that our algorithm will succeed so long as $\varepsilon \lesssim 0.1$ or so.

5.1 Constructing a low-rank bicluster B :

In the above discussion (and further on below) we perform numerical experiments which involve constructing and binarizing $m \times n$ matrices B which are randomly-oriented, rank- l , and ε -error. When m, n are small we construct B using our original prescription above:

1. Given l, m, n, ε , construct a randomly-oriented multivariate gaussian distribution ρ in \mathbb{R}^m , with l large principal values equal to 1 and the remaining principal values equal to ε .
2. Construct A to be an $m \times n$ matrix by drawing each of the n columns of A independently from ρ .
3. Construct $B = \text{sign}(A)$.

When m, n are sufficiently large (e.g., $m, n > 128$), it becomes cumbersome for us to construct B this way, and instead we approximate B as follows:

1. Construct \tilde{A} to be an $m \times n$ matrix with independent entries drawn from a standard normal distribution.
2. Form the singular-value-decomposition $\tilde{A} = USV^\top$.
3. Redefine the diagonal entries of S by setting $S(l, l) = 1$, and $S(j, j) = \varepsilon$ for $j > l$.
4. Redefine \tilde{A} via $\tilde{A} := USV^\top$.
5. Construct $\tilde{B} = \text{sign}(\tilde{A})$

The matrices \tilde{A} formed using this svd-based method will not be drawn from the same distribution as the matrices A formed via our original prescription. Nevertheless, the matrices \tilde{A} will still be rank- l with error- ε ; when m is large the binarized matrices B will be very close (in distribution) to the binarized matrices \tilde{B} .

When m, n are even larger (e.g., $m, n > 1024$), even this svd-based prescription might be too slow. In these large- m, n scenarios, we take the following shortcut (in the cases where $l = 1$ and $m \leq n$):

1. Given $l = 1, m, n \gg 1$ and ε , determine $g := g_{1,\varepsilon,m}$ using Eq. 1 above.
2. Define the ‘flip-probability’ $f := f_{\varepsilon\sqrt{m}}$ as follows: $1-g = 4f^3(1-f) + 4f(1-f)^3$, or $1-g = 4f(1-f)(1-2f(1-f))$. This can be done by first solving for $F = f(1-f)$ via $1-g = 4F(1-2F)$, and then solving for f .
3. Construct \hat{A} to be a rank-1 $m \times n$ binary-matrix $\hat{A} = uv^\top$, where u and v are both random binary vectors of length m and n respectively.
4. Choose fmn of the entries of \hat{A} at random, and flip their sign.
5. Define $\hat{B} = \hat{A}$.

Note that the flip-probability f can be approximated by:

$$f \approx 1 - g \approx \frac{4}{\pi} \varepsilon \sqrt{m} \left[\frac{1}{\sqrt{2\pi}} + \frac{2}{\sqrt{2\pi}} \log \left(\frac{\pi}{2\varepsilon\sqrt{m}} \right) - \frac{2}{\pi} [0.7961] \right] \text{ when } \varepsilon\sqrt{m} \lesssim 0.01,$$

$$\text{and } f \approx \frac{1}{2} - \frac{1}{\sqrt{2\pi} \cdot \varepsilon\sqrt{m}} \text{ when } \varepsilon\sqrt{m} \gtrsim 10.$$

This sign-flipping method produces a random binary $m \times n$ matrix \hat{B} . Note that the matrices \hat{B} will not be drawn from the same distribution as the matrices B or \tilde{B} . For example, the entries of the covariance matrix $\hat{B}\hat{B}^\top$ will be drawn from a

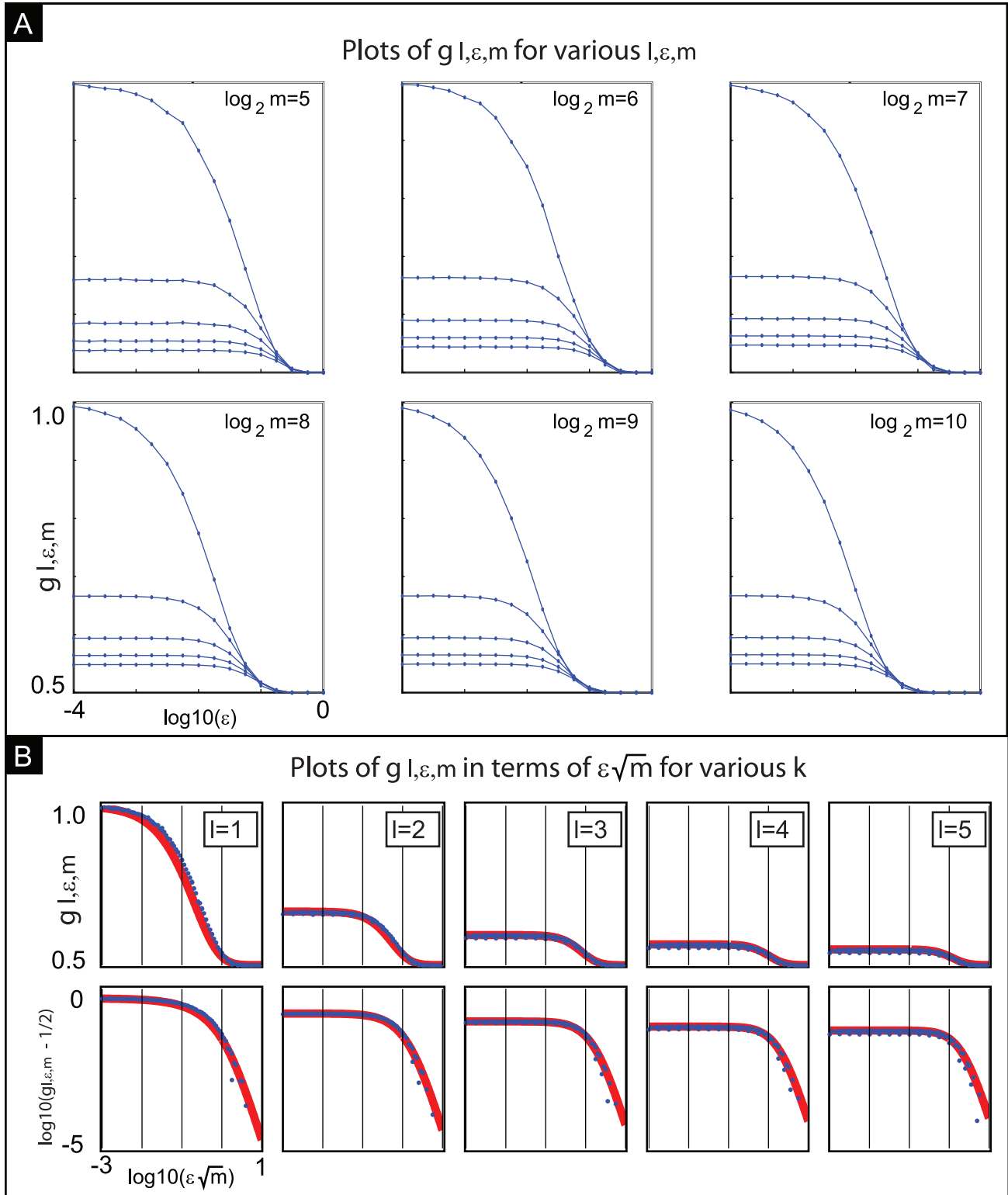


Figure 23: Numerical experiments confirming the behavior of $g_{l,\varepsilon,m}$. In Panel-A we show several curves for $g_{l,\varepsilon,m}$ as a function of ε for various m . Each subplot contains curves for $l = 1, 2, \dots, 5$, with the higher curves corresponding to lower values of l . In Panel-B we rearrange the data from our numerical experiments as a function of $\varepsilon\sqrt{m}$. We show this data for $g_{l,\varepsilon,m}$ on top and $\log_{10}(g_{l,\varepsilon,m} - 0.5)$ on the bottom. Overlaid on these numerical experiments is the large- m approximation for $g_{l,\varepsilon,m}$ derived above (red curve).

different distribution than the entries of the covariance matrices BB^\top or $\tilde{B}\tilde{B}^\top$. Nevertheless, the mean-squared-covariance of \hat{B} will (on average) match the mean-squared-covariance of B and \tilde{B} . That is to say, the average value (across multiple trials) of $\text{mean}\left(\text{diag}\left(\hat{B}\hat{B}^\top\hat{B}\hat{B}^\top\right)\right)$ will equal the average value (across multiple trials) of $\text{mean}\left(\text{diag}\left(BB^\top BB^\top\right)\right)$, as well as the average value of $\text{mean}\left(\text{diag}\left(\tilde{B}\tilde{B}^\top\tilde{B}\tilde{B}^\top\right)\right)$. This is equivalent to saying that the average row-score $[Z_{\text{ROW}}]_j$ of a row from \hat{B} will equal the average row-score of a row from B or \tilde{B} ; a similar statement holds for the column-scores $[Z_{\text{COL}}]_k$. Put in more colloquial terms, the average level of coherence between rows of \hat{B} is the same as the average level of coherence between rows of B or \tilde{B} . Thus, we see little difference in our numerical experiments if we use $B = \text{sign}(A)$, $\tilde{B} = \text{sign}(\hat{A})$ or $\hat{B} = \hat{A}$.

Finally, we note that all the matrices generated above will have roughly the same number of positive and negative entries. That is to say, the fraction of positive entries (i.e., sparsity-coefficient) will be $p \sim 0.5$, and the fraction of negative entries will be $q = 1 - p \sim 0.5$. If we wish to generate a rank-1 matrix that has, say, a sparsity-coefficient $\tilde{p} < 0.5$ (i.e., more negative entries than positive entries), then we can use a slightly different algorithm, described in section 5.2.

5.2 Generating a sparse low-rank B

To generate a rank-1 error- ε matrix that has, say, a sparsity-coefficient $\tilde{p} < 0.5$ (i.e., more negative entries than positive entries), then we take the following shortcut:

1. Given ε , determine $g := g_{1,\varepsilon,m}$ and $f := f_{\varepsilon\sqrt{m}}$ as in the construction of \hat{A} in section 5.1.
2. Given \tilde{p} , Set probability $\bar{f} = \min(f, \tilde{p})$, and $h = (\tilde{p} - \bar{f}) / (1 - 2\bar{f})$.
3. Set probability $\hat{p} = 0.5 - 0.5\sqrt{1 - 2\min(h, 1 - h)}$ and $\hat{q} = 1 - \hat{p}$
4. Generate an $m \times 1$ binary vector \vec{u} by choosing each entry to be either $+1$ or -1 , with probability \hat{p} and \hat{q} , respectively.
5. Generate an $n \times 1$ binary-vector \vec{v} by choosing each entry to be either $+1$ or -1 , with probability \hat{q} and \hat{p} , respectively. Note that the sign-associations for \vec{v} are opposite to those of \vec{u} .
6. Form $\bar{A} = \vec{u} \times \vec{v}^\top$.
7. Choose $\bar{f}mn$ of the entries of \bar{A} at random and flip their sign.
8. Define $\bar{B} = \bar{A}$.

This method will generate a binary-matrix \bar{B} that has a fraction \tilde{p} of positive entries. While the spectrum of \bar{B} will not look like the spectrum of B or \tilde{B} from section 5.1, a loop drawn from \bar{B} will have the same probability $g_{1,\varepsilon,m}$ of being rank-1.

If \tilde{p} were to be > 0.5 instead of < 0.5 , then we could perform the above procedure for a sparsity-coefficient of $1 - \tilde{p}$, and then negate the result.

6 Application to the Planted-Bicluster problem:

The algorithm we propose in section 2 is often capable of solving the ‘planted bicluster’ problem. As a demonstration, we consider a large $M \times M$ data-matrix matrix D with entries chosen independently from a distribution with median 0. After creating D we’ll embed within D an $m \times m$ submatrix B , which is rank-1 with error ε (as described in section 5). We’ll assume that $M \gg m \gg 1$. Our algorithm will detect B with high probability whenever $m^3(2g_{1,\varepsilon,m} - 1) \gtrsim \sqrt{2M^3}$. This means that, when $\varepsilon\sqrt{m} < 1$, our algorithm generally will work well if $m \gtrsim \sqrt{M}$, but not if $m = o(\sqrt{M})$. In addition, if $\varepsilon\sqrt{m} > 1$, then (by using the asymptotic formula for $g_{1,\varepsilon,m}$ in section 17) we see that our algorithm will work well if $m \gtrsim \sqrt[3]{\pi^2/4} \cdot [\varepsilon\sqrt{m}]^{4/3} \cdot \sqrt{M}$. However, as commented on above, most applications in gene-expression analysis do not involve sufficiently many patients (and do not have sufficiently large biclusters) for this second statement to be very meaningful. If we limit ourselves to the typical ranges of m and M which we see in most applications, we can state that our algorithm works well when the following two detection-thresholds are met:

Size-Threshold: m should be $\gtrsim \sqrt{M}$, and **Noise-Threshold:** $\varepsilon\sqrt{m}$ should be $\lesssim 1$.

These two conditions represent two distinct detection-thresholds associated with our algorithm (see Figs 24, 25 and 26). Analogous detection-thresholds can be derived using $g_{l,\varepsilon,m}$ when the planted bicluster B is numerical-rank l instead of 1 (see Figs 30 and 31).

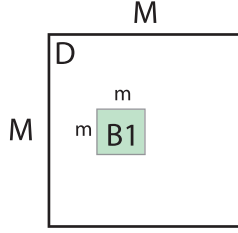


Figure 24: This figure illustrates the setup for the numerical experiments shown in Fig 26. These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving case-patients only (i.e., D only). For each numerical experiment we let D be a large $M \times M$ random matrix with entries chosen independently from a distribution with median 0. We implant within D a smaller $m \times m$ bicluster B that is rank-1 with error ε (see discussion at the end of section 5, as well as Fig 25). Our algorithm produces a list of row- and column-indices of D in the order in which they are eliminated; those rows and columns retained the longest are expected to be members of B . For each numerical experiment we calculate the auc A_R (i.e., area under the receiver operator characteristic curve) associated with the row-indices of B with respect to the output list from our algorithm. The value A_R is equal to the probability that: given a randomly chosen row from B as well as a randomly chosen row from outside of B , our algorithm eliminates the latter before the former (i.e., the latter is lower on our list than the former). This value A_R is equal to $(\bar{r} - (m + 1) / 2) / (M - m)$, where \bar{r} is the average rank of the rows of B within our list. We calculate the auc A_C for the columns similarly. Finally, we use $A = (A_R + A_C) / 2$ as a metric of success; values of A near 1 mean that the rows and columns of B were filtered to the top by our algorithm. Values of A near 0.5 mean that our algorithm failed to detect B .

We note that these two detection-thresholds are not the same as the ‘information-theoretic’ phase-transitions for the planted-bicluster problem. That is to say, our algorithm cannot typically detect biclusters with a size $m \ll \sqrt{M}$ or a noise $\varepsilon\sqrt{m} \gg 1$, even though such biclusters might be very significant (in a statistical sense). For a more detailed discussion regarding these information-theoretic phase-transitions see section 19.

Behavior when D is correlated: Our analysis above also sheds light on situations when D is itself numerically low-rank (instead of having independent entries). To discuss this situation, let’s imagine that the $M \times M$ matrix D were numerical-rank l_D with error ε_D prior to binarization. We’ll embed within D an $m \times m$ planted-bicluster B of numerical-rank l_B with error ε_B , involving the row-subset J_B and column-subset K_B . Let’s refer to the matrix-entries of D that are not in B as B^c .

Just as before, a typical loop drawn from B will be rank-1 with probability $g_{l_B, \varepsilon_B, m} > 1/2$. However, unlike before, a typical loop drawn from D will also be rank-1 with probability $g_{l_D, \varepsilon_D, M} > 1/2$. This means that the overall ‘signal’ coming from B now has to compete with a signal coming from D . In fact, the typical row-score of a row in J_{B^c} is no longer 0, but now has an expected-value of

$$\widetilde{\sum}_{j \in J_{B^c}} [Z_{\text{ROW}}]_j \sim M^3 (2g_{l_D, \varepsilon_D, M} - 1),$$

which we’ll denote by \bar{Z} . In addition, the standard-deviation of the row-score across rows in J_{B^c} is no longer simply $\sqrt{2M^3}$, but is now a function of $\varepsilon_D\sqrt{M}$, as well as l_D , which we’ll denote by σ_Z . Our algorithm will detect B with high probability only when the typical value of a row-score from J_B is on the same order as (or larger than) $\bar{Z} + \sigma_Z$. If $\varepsilon_D\sqrt{M}$ and l_D are sufficiently large, then $\bar{Z} \sim 0$ and $\sigma_Z \sim \sqrt{2M^3}$; the detection-thresholds for our method will be comparable to the case where the entries of D are drawn independently. On the other hand, if l_D and $\varepsilon_D\sqrt{M}$ are sufficiently small (e.g., $l_D = 1$ and $\varepsilon_D\sqrt{M} \sim 0.1$), then $\bar{Z} + \sigma_Z$ can be several orders of magnitude larger than $\sqrt{2M^3}$, and our method will fail unless $m = O(M)$.

7 More detailed discussion of the algorithm:

As described in section 2, our algorithm has three main components: (i) Binarization, (ii) calculation of loop-scores, and (iii) iteration. In this section we discuss these three main components in more detail, providing some justification for our choices. In this section we’ll often (but not always) use the planted-bicluster problem to frame our discussion. We’ll also sometimes compare our algorithm to a simple spectral algorithm (discussed in more detail within section 22).

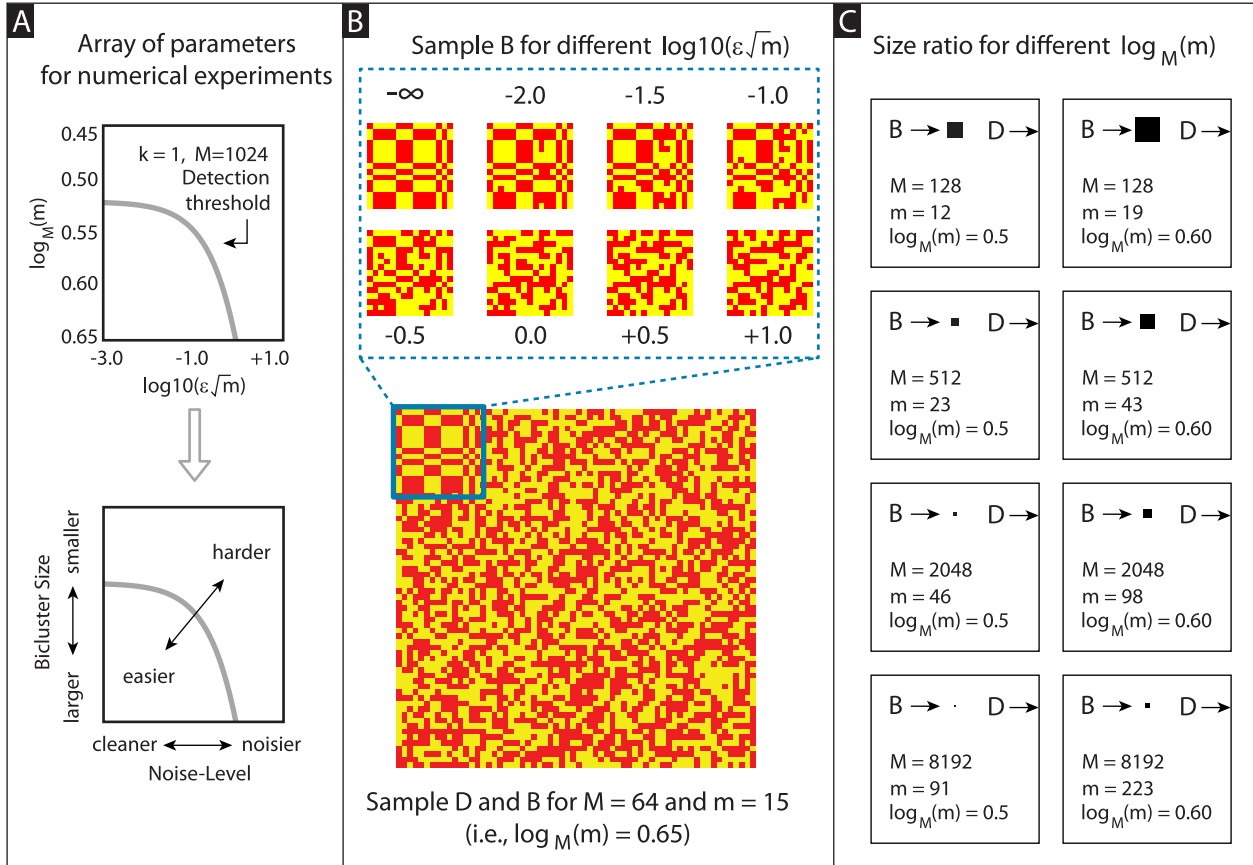


Figure 25: This figure describes some of the qualitative features of the numerical experiments shown in Fig 26. In Panel-A we show the array of parameter values used for $\epsilon\sqrt{m}$ and $\log_M(m)$ within our numerical experiments. The first of these parameters controls the noise-level (horizontal axis), while the second controls the bicluster size (vertical axis). Situations where the noise-level is low and the bicluster-size is large correspond to planted-biclusters which are ‘easier’ to find. By contrast, situations where the noise-level is large and the bicluster-size is small correspond to planted-biclusters which are ‘harder’ to find. The thin grey line shown in Panel-A corresponds to the detection-threshold of our loop-counting algorithm when applied to the planted-bicluster problem with rank $l = 1$ and $M = 1024$ (see Fig 26). Our loop-counting algorithm typically succeeds at solving planted-bicluster problems that are on the ‘easy side’ of this detection-threshold. In Panel-B we show a sample matrix D of size $M = 64$ used for our numerical experiments. We have implanted within D a rank-1 bicluster B of size $m = 15$, corresponding to $\log_M(m) = 0.65$ and $\epsilon\sqrt{m} = 0$. For illustration, we have rearranged the rows and columns of D to reveal B in the upper-left corner (see cyan box). This bicluster is large and the noise-level is very low, so finding B is very easy (similar to the lower left corner of our parameter-array). If we were to increase the noise-level, the bicluster B would no longer be exactly rank-1; sample biclusters with varying noise-level are shown above. In Panel-C we illustrate the size-ratio between D and B for different values of M and the parameter $\log_M(m)$. In each case the outer box (not filled) represents the size of D , whereas the interior box (filled) represents the size of B . Note that, when M is large and $\log_M(m)$ is small, the bicluster B can be quite a bit smaller than D .

Performance of biclustering algorithm on D-only

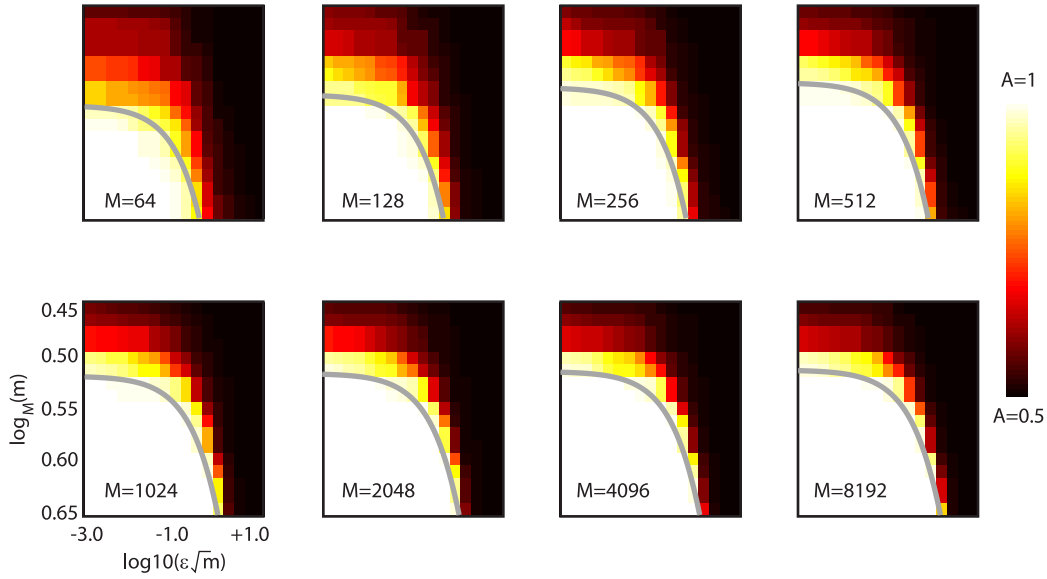


Figure 26: These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving case-patients only (see Figs 24 and 25). In the panels shown we plot the trial-averaged value of A as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. Each subplot takes the form of a heatmap, with each pixel showing the value of A for a given value of $\log_{10}(\epsilon\sqrt{m})$ and $\log_M(m)$ (averaged over at least 32 trials). The different subplots correspond to different values for M . Our algorithm is generally successful when $\log_{10}(\epsilon\sqrt{m}) \lesssim 0$ and $\log_M(m) \gtrsim 0.5$. The grey line superimposed on the heatmap is the detection-boundary calculated from the formula $m^3(2g_{1,\epsilon,m} - 1) = \sqrt{2M^3}$. See Figs 30 and 31 for similar results pertaining to rank-2 and rank-3 planted-biclusters.

7.1 Why binarize? Advantages and disadvantages

The first step of our algorithm is to binarize the data: that is, to send each entry of D to -1 or $+1$, depending on its sign. This binarization step carries with it two main advantages.

Speed: The first and most obvious advantage is speed: A binarized matrix can be stored efficiently, and calculating row- and column-scores for a binarized matrix can be accomplished without use of floating-point operations (instead using bitwise operation such as xor). For example, storing a large data-matrix containing 2 million measurements taken across 2 thousand patients, requires 32 GB of memory to store in double-precision, but only 500 MB after binarization. Calculating simple vector-vector and matrix-vector products associated with such a binarized matrix can be $\gtrsim 64$ times faster than analogous operations performed in double-precision.

Normalization: Binarization serves as a canonical normalization which amplifies signals that are sharp, but of low magnitude. This comes at a small cost (which we'll touch on at the end of this section), but serves to expose many kinds of structures that would otherwise be difficult to detect.

To set the stage for this discussion, let's consider a large data-matrix D comprising N gene-expression measurements taken across M patients. A priori, we expect the entries of D to be mostly uncorrelated (or, say, weakly correlated). This is because, generally speaking, there will be multiple complicated biological mechanisms driving the gene-expression within D . Some mechanisms work together, others are antagonistic, while others are independent; we typically expect the accumulation of all these different mechanisms to act like a source of 'noise', de-correlating the genes in D .

Now let's assume that D contains an $m \times n$ low-rank bicluster B spanning row-subset J_B and column-subset K_B . The low-rank structure of B implies that, unlike the typical genes in D , the genes in K_B are strongly correlated (i.e., co-expressed) across the patients in J_B . The success of our methodology depends on how – and why – the genes in B are correlated.

More specifically, if B were indeed biologically significant, then we would expect there to be some underlying physiological reason for the co-expression within B . While there are many different possible mechanisms which might underly this co-expression (i.e., the drivers of gene-expression can be very complicated), we can abstractly group these mechanisms into two rough paradigms:

Spectral-amplification In this first paradigm we imagine that the mechanisms driving the co-expression in B do so by *adding* a new strong source of co-variation to the genes in K_B .

Spectral-suppression For the second paradigm we imagine that the mechanisms driving the co-expression in B do so by *removing* many of the small weak sources of noise which would ordinarily de-correlate the genes in K_B .

These two paradigms can be idealized mathematically with the following two generative models:

Spectral-amplification The low-rank bicluster B could be produced by amplifying low-order spectral information. For example, we can initially draw the entries of B from a noisy (uncorrelated) distribution. Once drawn, we then increase the magnitude of the low-order (i.e., dominant) principal-components of B , while leaving the high-order (i.e., less dominant) principal-components unchanged. This process will magnify the first few principal-components of B , adding a large new source of covariance to the bicluster’s genes.

Spectral-suppression The bicluster B could instead be produced by suppressing high-order spectral information. For example, as above, we can start by drawing the entries of B from a noisy distribution. This time, however, we subsequently decrease the magnitude of the high-order principal-components of B , leaving the low-order principal-components unchanged. This process will end up introducing correlations into B by removing (or quenching) many of the smaller sources of variance that would have otherwise decorrelated the bicluster’s genes.

Both of the paradigms described above give rise to low-rank biclusters, but with rather different properties. Foremost among these differences is that spectrally-suppressed biclusters are harder to detect than spectrally-amplified biclusters. The reason for this is that the rows and columns of a spectrally-suppressed bicluster are typically not large, and are usually buried within the distribution associated with the rest of the data. By contrast, the rows and columns of a spectrally-amplified bicluster are typically large, and often stick out of the rest of the data. Consequently, a spectrally-amplified bicluster can usually be detected simply by looking for ‘outliers’ in the data (e.g., a simple search based on entry-by-entry amplitude). Indeed, if the original data set lacks an abundance of outliers, one can usually rule out the existence of spectrally-amplified biclusters, leaving the harder task of detecting any hidden spectrally-suppressed biclusters.

It is for these reasons that we’ve designed our algorithm with the goal of detecting spectrally-suppressed biclusters. One of the major issues that arises when trying to detect spectrally-suppressed biclusters is that, even if a spectrally-suppressed bicluster has a ‘sharp’ Pearson’s correlation (i.e., pairs of rows or columns within the bicluster have a normalized correlation close to ± 1), the actual covariances of that bicluster (i.e., dot-products of rows or columns) will typically be low in magnitude. Consequently, without some kind of normalization, the signal associated with a spectrally-suppressed bicluster will be proportional to its low-amplitude covariance, rather than to its possibly-high Pearson’s correlation. Our binarization step rectifies this problem by putting the spectrally-suppressed bicluster on a similar footing with the rest of the data, allowing our loop-scores to detect it. An example of this kind of phenomenon is given in Fig 27. Note that the unbinarized loop-scores are uninformative, whereas the loop-scores after binarization expose the spectrally-suppressed bicluster.

We remark that our binarization step does indeed reduce the magnitude of the entries within any spectrally-amplified biclusters. However, because these biclusters were so easy to detect to begin with, we believe that binarization is an appropriate choice for most applications (i.e., when spectrally-amplified biclusters have already been ruled out).

Information loss?: It turns out that, in addition to obscuring signals associated with spectrally-amplified biclusters, binarization can also obscure the signals produced by biclusters that have essentially no noise. We’ll argue later that this isn’t a serious issue, but for now let’s consider an ‘essentially noiseless’ bicluster with $\varepsilon\sqrt{m} \sim 0$. Regardless of whether this bicluster has been created through spectral-amplification or spectral-suppression, it can be easily detected prior to binarization using the following simple ‘angle-matching’ algorithm: This angle-matching algorithm involves sifting through the loops in the data-matrix, associating with each loop the angle θ between its two rows. Most loops will have an angle that ranges across the interval $[-\pi, +\pi]$, whereas the loops within the bicluster will have an angle that is very close to either 0 or $\pm\pi$. The bicluster can then be picked out simply by retaining the loops with $\theta \in \{0, \pm\pi\}$. The reason this angle-matching algorithm works is because, when $\varepsilon\sqrt{m} \sim 0$, there is a great deal of angular information in the distribution of θ . In this idealized no-noise scenario, binarizing the data destroys this extra angular information, obscuring the bicluster and preventing the angle-matching algorithm from working well.

We can quantify this phenomenon more precisely using the reasoning of section 5. Assuming that the bicluster B is defined as described in section 5, a pair of randomly chosen columns within the bicluster gives rise to a 2-dimensional distribution of the form $\tilde{\rho}$; i.e., each 1×2 row-vector within this column-pair is drawn from $\tilde{\rho}$. Consequently, $\tilde{\rho}$ induces a distribution $\rho_B(\omega)$ on the angle ω of each row-vector drawn from $\tilde{\rho}$ (see Fig 28A). Note that the angle θ associated with any loop in this column-pair will take the form of $\theta = \omega_1 - \omega_2$, with both ω_1, ω_2 drawn from $\rho_B(\omega)$; i.e., θ is drawn from $\rho_B \star \rho_B^c$, where $\rho_B^c(\omega) = \rho_B(-\omega)$.

The angular-information contained within this distribution of θ is thus derived from the angular-information contained within $\rho_B(\omega)$. When $\varepsilon\sqrt{m}$ is very tiny the 2-dimensional distribution $\tilde{\rho}$ will be a very eccentric ellipse dominated by its first principal-component; the angular-distribution $\rho_B(\omega)$ will be very sharply peaked around the major-axis of this ellipse, and the angular-distribution of θ will be peaked around $\{0, \pm\pi\}$. The angular-information contained within $\rho_B(\omega)$ can

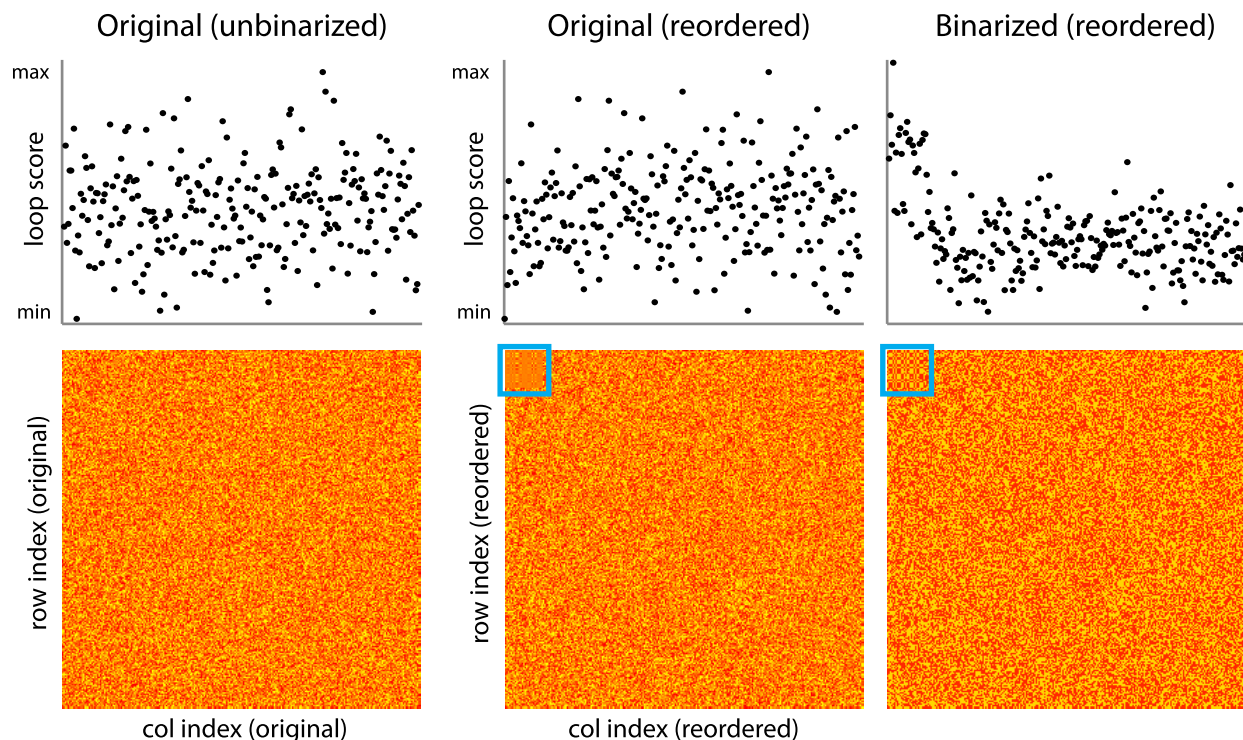


Figure 27: Binarization helps reveal spectrally suppressed biclusters. On the left we show a random matrix D (bottom) in which a spectrally-suppressed bicluster B has been created. In the middle (bottom) we rearrange the rows and columns of D to reveal the embedded bicluster B in the upper left corner (cyan square). The bicluster B was created by first choosing a subset of rows and columns of D . We then calculated a principal-component-analysis USV^\top of this submatrix. We then reduced the high-order principal-values $S_{2,2}$, $S_{3,3}$, etc. by a factor of 10, while retaining the low-order principal-value $S_{1,1}$. We then reconstituted $B = USV^\top$, and reinserted B into D . The column-scores $[Z_{\text{COL}}] = \text{diag}(D^\top D D^\top D)$ for each column of the (unbinarized) data-matrix D are shown above each matrix. Note that – prior to binarization – the column-scores do not clearly indicate the columns which constitute the bicluster B (neither would scores produced by the simple spectral biclustering method applied to this non-binarized version of D). On the right (bottom) we show the binarized matrix $D := \text{sign}(D) = 2(D > 0) - 1$, with the rows and columns reorganized to reveal B . The column-scores for each column of the (binarized) data-matrix D are shown above; note that – post binarization – the column-scores associated with D do indicate the columns of B (as would scores produced by the simple spectral method). The row-scores exhibit a similar phenomenon.

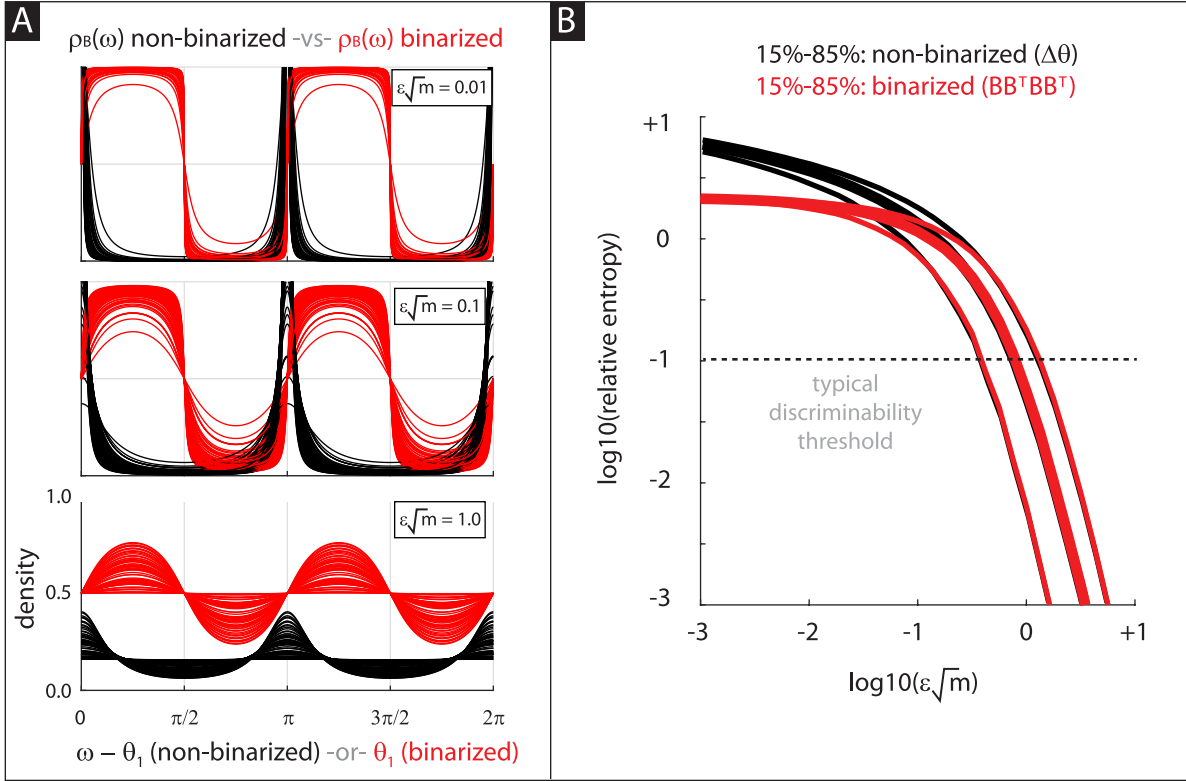


Figure 28: In panel-A we show a few sample distributions $\rho_B(\omega)$ associated with $m = 128$ and various values of $\varepsilon\sqrt{m}$. The original distribution $\rho_B(\omega)$ is shown in black (as a function of $\omega - \theta_1$), and the binarized version is shown in red (as a function of θ_1 , assuming ω is in the first or third quadrant). Note that $\rho_B(\omega)$ becomes more and more similar to the uniform distribution $\rho_D(\omega)$ (not shown) as $\varepsilon\sqrt{m}$ increases. In panel-B we show the typical relative-entropy $\mathcal{D}_{KL}(\varepsilon\sqrt{m})$ for a large number of samples of non-binarized (black) and binarized (red) random rank-1 matrices (with m fixed at $m = 128$). We show the median, 15th and 85th percentiles of \mathcal{D}_{KL} as a function of $\varepsilon\sqrt{m}$. Note that binarization makes little difference for $\varepsilon\sqrt{m} \gtrsim 1/10$. Even though this particular figure was generated with $m = 128$, changing the value of m makes little qualitative difference; binarization has essentially no detrimental effect as long as $\varepsilon\sqrt{m} \gtrsim 1/10$.

be quantified via the relative-entropy $\mathcal{D}_{KL}(\varepsilon\sqrt{m}) = \int \rho_B \log(\rho_B/\rho_D) d\omega$, which compares ρ_B to the uniform-distribution $\rho_D(\omega)$ expected outside of the bicluster. As we alluded to above, when $\varepsilon\sqrt{m} \sim 0$ this relative-entropy \mathcal{D}_{KL} is very high for the original data, but is much lower for the binarized data (i.e., where half the loops in the data-matrix have $\theta = 0$ or $\pm\pi$). In this situation binarizing the data destroys this extra angular information, dramatically reducing the relative-entropy.

It turns out, however, that this phenomenon is restricted only to essentially noiseless biclusters. When $\varepsilon\sqrt{m}$ is significantly larger than 0 (say, $\gtrsim 0.01$), then the distribution $\tilde{\rho}$ is not so eccentric; the angular-distribution $\rho_B(\omega)$ will be distributed more evenly, and the distribution of θ will be closer to uniform. Thus, when $\varepsilon\sqrt{m} \gtrsim 0.01$, the relative-entropy \mathcal{D}_{KL} is not much higher prior to binarization than afterwards. In fact, when $\varepsilon\sqrt{m} \gtrsim 1/10$ there is essentially no loss of angular information after binarization. We illustrate this point in Fig 28A, which shows many samples of $\rho_B(\omega)$ both pre- and post-binarization, as well as Fig 28B, which shows $\mathcal{D}_{KL}(\varepsilon\sqrt{m})$ sampled across many rank-1 biclusters (the details involved in generating Fig 28 are described in section 20). Consequently, if we expect our data to contain realistic biclusters (i.e., with $\varepsilon\sqrt{m} \gtrsim 1/10$), we don't have to consider any sort of angle-matching algorithm (which would be useless in practice), and we can binarize our data without fearing any information-loss.

In summary: We expect that realistic data-sets will not contain perfectly correlated biclusters; for realistic biclusters $\varepsilon\sqrt{m}$ will be on the order of 0.1, at which point binarization does not destroy a significant amount of angular-information. In addition, binarization has the added advantage of amplifying the signals produced by any spectrally-suppressed biclusters. While binarization does suppress the signals of any spectrally-amplified biclusters, these can usually be detected simply by looking for outliers in the data.

Finally, we remark that the binarization-step in our algorithm can be modified to accommodate a 'soft' binarization, wherein some values in the data-array are set to an intermediate number, rather than to ± 1 . For example, if there is some reason to suspect that the noise within the data involves small-amplitude structured perturbations around the binarization-threshold, then all entries in the data-array close to the binarization-threshold can be set to 0.

7.2 Why use loops? Advantages and disadvantages

The second step of our algorithm involves calculating the loop-scores. These loop-scores will be indicative of the location of any hidden low-rank biclusters, provided that these hidden biclusters are sufficiently large (and not too noisy). These loop-scores have the obvious advantage that they are easy to calculate and, as we’ll see later, it is easy to correct them for case-control status and various covariates. It is also the case that the loop-scores are ‘close to optimal’ regarding the size-transition in the low-noise limit; within the limits of our framework it is difficult to construct a more sophisticated score that is much better at detecting small planted-biclusters.

To see why this might be the case let’s return to a special case of the planted-bicluster problem: let’s assume that we have a large $M \times M$ binary-matrix D containing a planted-bicluster B of size $m \times m$ which is exactly rank-1 (i.e., $\varepsilon\sqrt{m} = 0$). These ‘low-noise’ biclusters are the easiest to detect, and will allow us to focus on the size-transition associated with the planted-bicluster problem. In terms of notation, let’s say that the bicluster B is formed via $B = uv^\top$, where u and v are both random binary vectors of size m . The bicluster B is then implanted into D using the randomly chosen row- and column-subsets J_B and K_B (each of size m).

Recall that our algorithm proceeds by calculating the ‘loop-scores’ associated with each row and column of D . For the purposes of this discussion, let’s use $l[j_1, j_2; k_1, k_2]$ to denote the ‘loop’ (i.e., 2×2 submatrix) spanning rows j_1 and j_2 as well as columns k_1 and k_2 . Our loop-scores $[Z_{\text{ROW}}]_j$ for a given row j are formed as follows. First, in Step (i), we consider the set S_j of all loops $l[j_1, j_2; k_1, k_2]$ with $j \in \{j_1, j_2\}$. Then, in Step (ii), we measure something like the rank $\mu(l)$ for each $l \in S_j$; in our case $\mu(l)$ is chosen to be 3 minus twice the rank of l . Finally, in Step (iii), we aggregate the sum $[Z_{\text{ROW}}]_j = \sum_{l \in S_j} \mu(l)$.

Our hope is that the loop-scores Z_{ROW} (and also Z_{COL} , which are constructed similarly) are indicative of the rows J_B (and columns K_B) of B . As we’ve seen above, our loop-scores will be informative when the size of B is larger than the detection-threshold \sqrt{M} (i.e., when $m \gtrsim \sqrt{M}$). A natural question is: are there better ways to score each row and column of D ? Could we do better than the size-threshold of \sqrt{M} ? Note that, for any given row j , our loop-score is one of a large class of ‘ c -scores’ which are constructed using the following general template:

Step (i) Consider the set S_j of all $c \times c$ -submatrices $l[j_1, \dots, j_c; k_1, \dots, k_c]$ with $j \in \{j_1, \dots, j_c\}$.

Step (ii) Measure some scalar quantity $\mu(l)$ for each $l \in S_j$;

Step (iii) Aggregate the sum $[Z_{\text{ROW}}]_j = \sum_{l \in S_j} \mu(l)$.

Perhaps we could gain an advantage if we allowed for the more general steps (i) and (ii) above? Specifically, instead of merely looking at loops $l[j_1, j_2; k_1, k_2]$, we might consider larger $c \times c$ submatrices $l[j_1, \dots, j_c; k_1, \dots, k_c]$. Moreover, instead of simply measuring the rank $\mu(l)$, we could use some more informative measurement instead (e.g., any other scalar-valued function $\mu : \{-1, +1\}^{c \times c} \rightarrow \mathbb{R}$). It turns out that neither of these modifications can substantially improve the c -score; no matter which fixed c we choose, or how we construct μ , we won’t be able to overcome the size-threshold of \sqrt{M} .

More specifically, we argue that – for fixed c and μ in the limit as m and M go to ∞ – no c -score will be informative when $m = o(\sqrt{M})$. The details of our argument can be found in section 21. We remark that our analysis only applies asymptotically, and does not guarantee that our method is the ‘best’ when m is on the order of \sqrt{M} (e.g., if m were to be a constant multiple of \sqrt{M} in the limit). Nevertheless, we believe that our method performs adequately when $m \sim \sqrt{M}$. For a demonstration of our method applied to the planted-bicluster problem, see Fig 29, as well as Figs 30 and 31, and section 22.

We should also emphasize that one can certainly design algorithms that perform better than ours on the planted-bicluster problem by stepping outside the restrictions of the template above. For example, one could consider the joint-distribution across multiple rows or columns of various $c \times c$ -submatrices. Such an algorithm would avoid the simple aggregation in step (iii), and may involve constructing a score which depends on multiple row- and column-indices, rather than just one. The reason we do not consider such algorithms is because it is not yet clear to us how to make them competitive in terms of speed and storage.

Alternatively, one could consider a simple spectral method which computes scores related to the coefficients of the dominant singular-vectors of D . Such spectral scores adopt a very specific μ , but involve an unbounded c (i.e., $c \rightarrow \infty$), thus avoiding the finite- c restriction in step (i) of the template above (and potentially allowing for better performance). The reason that we do not pursue spectral methods is because the spectral scores are usually less useful than our loop-scores when attempting to locate small biclusters. This phenomenon is illustrated in Figs 29, 30, 31 and is discussed in more detail in section 22.

A more general approach is to consider a message-passing algorithm akin to [M33], [M19], or [M41]. These message-passing algorithms operate by (i) setting up a bigraph between the rows and columns of the data matrix, (ii) iteratively accumulating a nonlinear function of ‘messages’ between nodes and edges in the bigraph, and (iii) adaptively refining this nonlinearity over each iteration to maximize the signal to noise ratio in the data. Such an approach sidesteps many of the

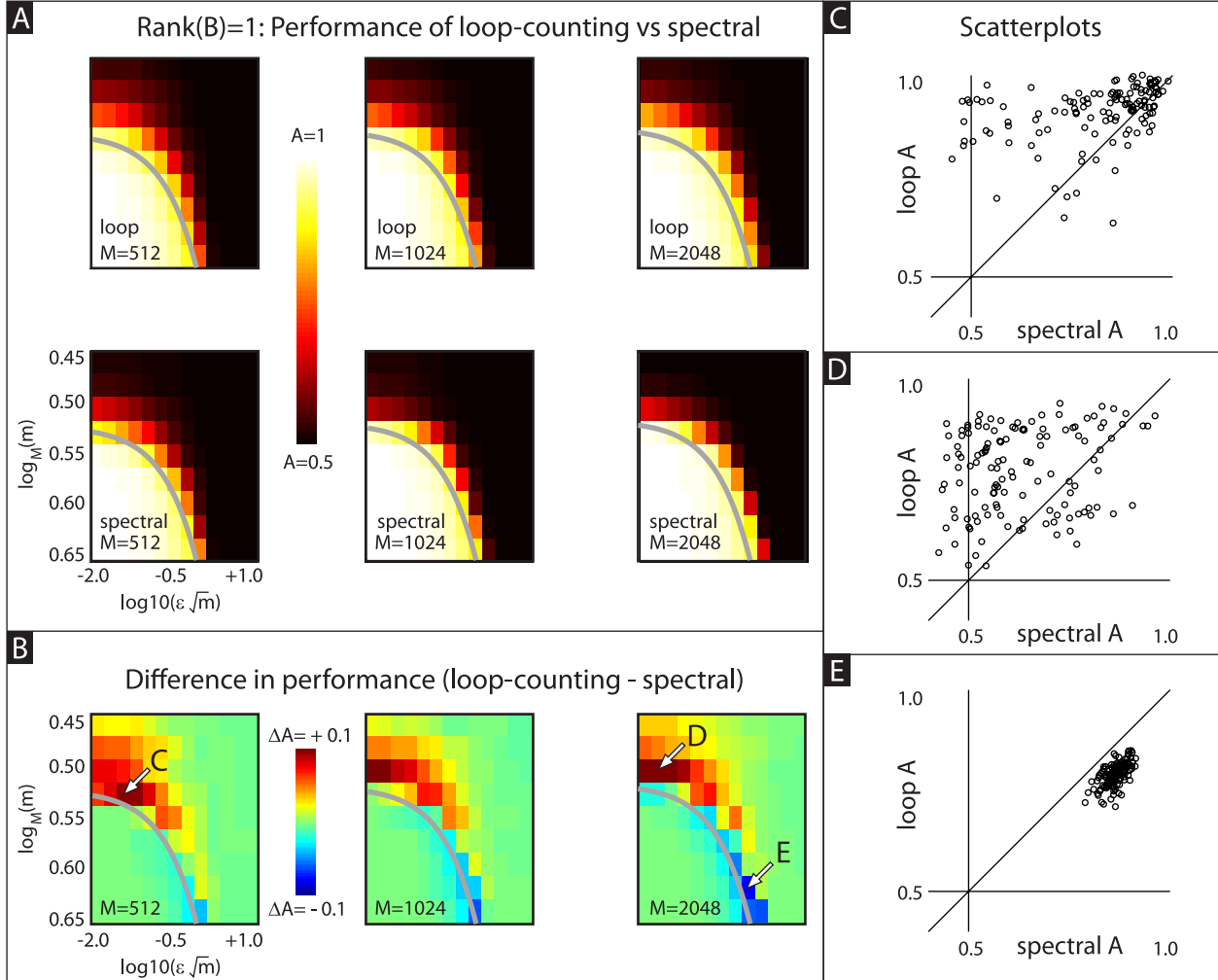


Figure 29: Performance of loop-counting vs spectral-biclustering applied to the rank-1 planted-bicluster problem. The format for this figure is similar to Fig 26. On top we show the trial-averaged auc A for our loop-counting method. In the middle we show the analogous auc for a simple implementation of the spectral method. For this simple spectral method we take the row- or column-score to be the entrywise-square of the first row- or column-wise principal vector of the binarized data (see section 22). On the bottom we show the difference in trial-averaged A between these two methods (see colorbar on right for scale). Note that, when the noise is small (i.e., when $\varepsilon\sqrt{m} \lesssim 1/3$), our loop-counting method has a higher rate of success than the spectral method. On the other hand, when the noise is large (i.e., $\varepsilon\sqrt{m} \gtrsim 1/3$) the spectral method has a higher rate of success. The thin grey line shows the detection-boundary for our loop-counting method (calculated using $m^3(2g_{1,\varepsilon,m} - 1) = \sqrt{2M^3}$). Panels C,D,E show a few scatterplots of some of the same data used to construct the pictures in panels A and B. More specifically, Panel-C shows the results of 128 trials performed with $M = 512$, $\log_M(m) = 0.525$, and $\log_{10}(\varepsilon\sqrt{m}) = -1.5$ (indicated with an arrowhead in Panel-B). For each of these trials we plot the A -value for the spectral-method on the horizontal-axis, and the A -value for the loop-counting method on the vertical-axis. For this choice of parameters the loop-counting method typically achieves moderate success (e.g., $A \geq 0.75$), whereas the spectral method sometimes fails completely (e.g., $A \sim 0.5$). Panels D and E show similar scatter plots for other parameter values (indicated with arrowheads in Panel-B).

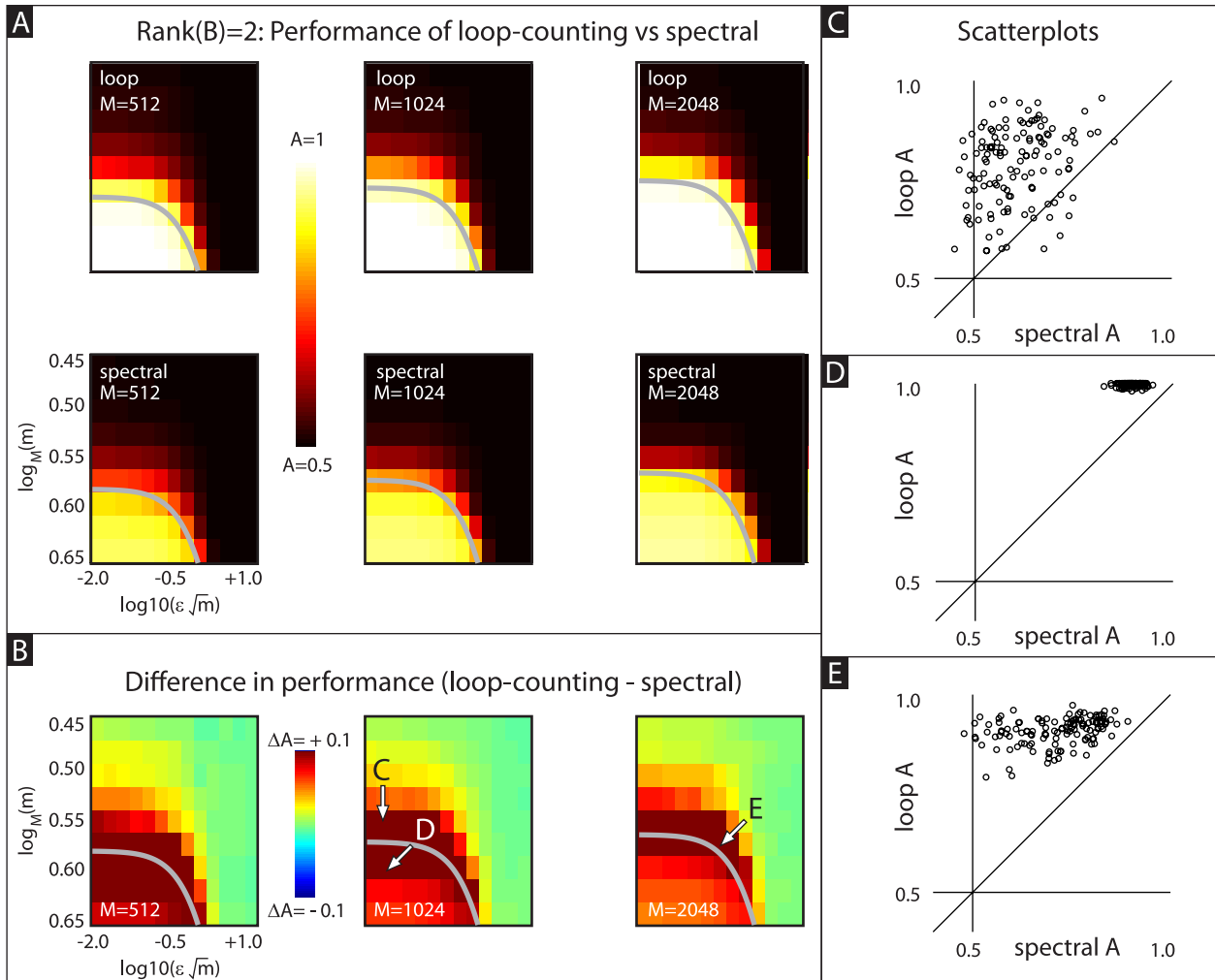


Figure 30: Performance of loop-scores vs spectral-biclustering for the rank-2 planted-bicluster problem. This figure has the same format as Fig 29. The thin grey line shows the detection-boundary for our loop-counting method (calculated using $m^3 (2g_{2,\varepsilon,m} - 1) = \sqrt{2M^3}$).

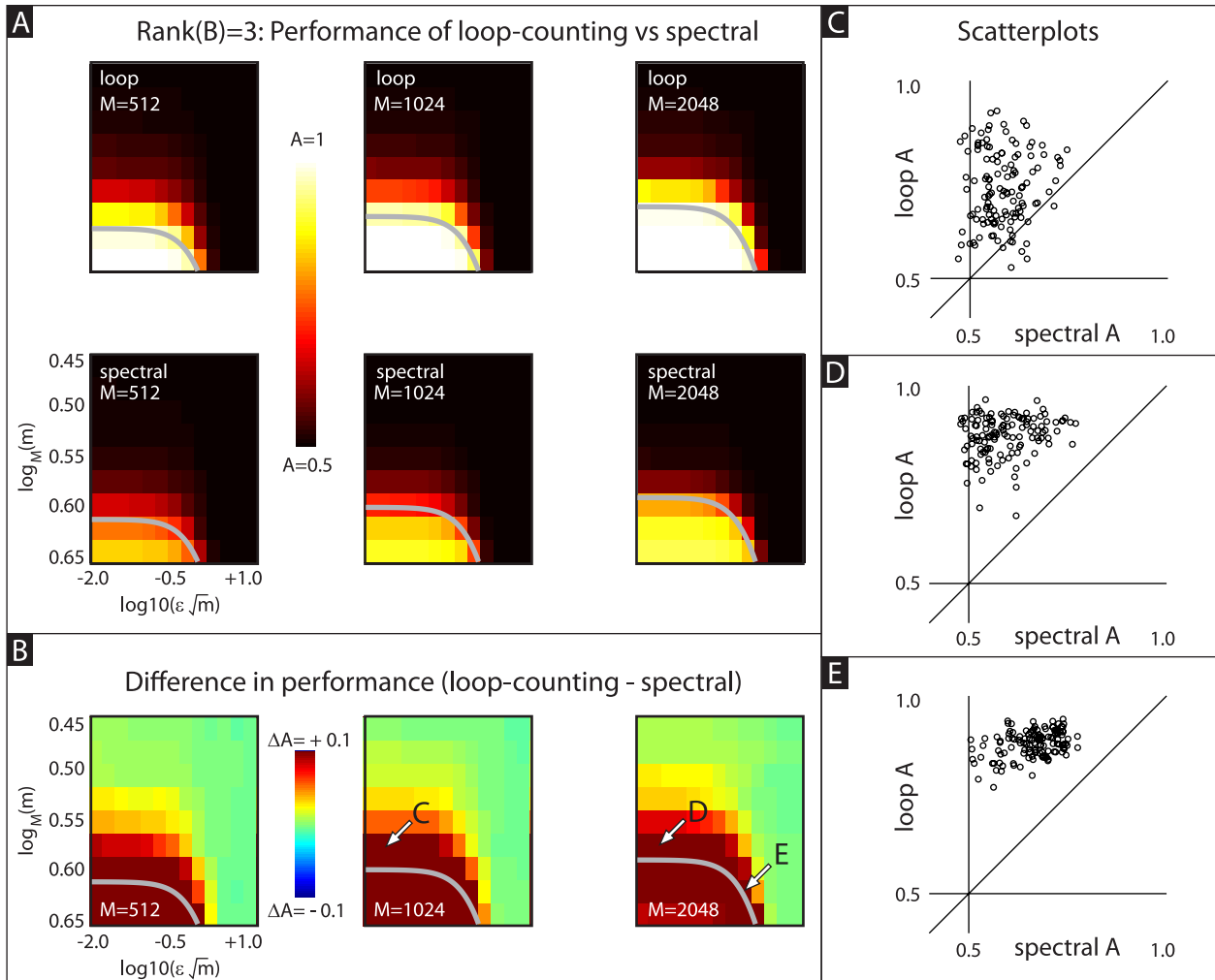


Figure 31: Performance of loop-scores vs spectral-biclustering for the rank-3 planted-bicluster problem. This figure has the same format as Fig 29. The thin grey line shows the detection-boundary for our loop-counting method (calculated using $m^3 (2g_{3,\epsilon,m} - 1) = \sqrt{2M^3}$).

restrictions of the template above; by iteratively updating their messages in step (ii) these algorithms have no fixed c , and by adaptively choosing the nonlinearity in step (iii) they are not limited to a fixed μ . Furthermore, the framework of a message passing algorithm allows for messages to be processed ‘jointly’, and is not limited a simple aggregation of scores. As we discuss briefly in section 22.4, this methodology seems very promising, and will be a focus of our future work.

7.3 Why iterate?

The third major step of our algorithm is to iterate; that is, to eliminate the rows and columns with the lowest scores, recalculate the loop-scores for the remaining rows and columns, and repeat. This iteration serves two purposes.

Improved Performance: First and foremost, the iterative procedure improves the performance of our algorithm with regards to the planted-bicluster problem. While the rows and columns of the planted-bicluster may not have the highest loop-scores, they are likely not to have the lowest loop-scores. By eliminating these lowest-scoring rows and columns first, we increase the probability that the highest loop-scores in the remaining data-matrix correspond to the planted-bicluster. This iterative approach is naturally associated with a parameter ‘ γ ’, indicating the fraction of remaining rows and/or columns to eliminate at each step. Fig 32 illustrates this performance of our algorithm for various γ . Because we don’t see much of a difference in our simulations for values of $\gamma \leq 0.05$, we have chosen $\gamma = 0.05$ or lower for the gene-expression and GWAS examples in this paper.

Focus on a single bicluster: In practice it is often the case that the data-matrix has multiple structures embedded within it. A single-pass algorithm can often be confused by the competing signals within the data, and may intermingle the various embedded structures. In these situations the iteration-step allows for the possibility that – as the rows and columns are eliminated – the remaining rows and columns will focus on a single embedded structure (usually the one with the strongest signal). An illustration of this phenomenon is given in Fig 33. For this example we embed two biclusters within a single data-matrix. The first bicluster $B1$ is indicated by cyan tags, the second bicluster $B2$ is indicated by magenta tags. When applying our iterative algorithm we not only detect both $B1$ and $B2$, but also separate them. A single-pass algorithm can also detect $B1$ and $B2$, but will typically intermingle the two.

We remark that the situation considered in Fig 33 is very idealized. In practice we rarely have crisp rectangular biclusters. Instead, the biclusters that appear in practice often have a soft ‘core’ of strongly correlated rows and columns, along with a loose collection of additional rows and columns that are less and less strongly correlated with the core. Iteration often affords an advantage in these situations as well; an iterative algorithm may be able to separate the ‘cores’ associated with different biclusters, focusing on the core with the strongest signal. An example illustrating this phenomenon is shown in Fig 34.

Finally, we comment that there are many different kinds of structures that can be embedded in a data-matrix, many of which can’t even be described as a bicluster. To elaborate, the very concept of a bicluster assumes that there is some linear ordering of rows (and another linear ordering of columns) which reveals the bicluster. That is, a row- and column-list such that rows and columns higher up on the list ‘participate’ more strongly in the bicluster than rows and columns farther down on the list. There are many kinds of structures which cannot be represented with a single set of lists; one such example is shown in Fig 35.

7.4 Comparison with some other biclustering methods

As demonstrated via numerical experiments in Figs 29, 30, 31, our loop-counting method compares rather favorably to a simple spectral biclustering method when applied to the planted-bicluster problem described in Fig 24.

In this section we perform analogous numerical experiments, allowing us to compare our loop-counting method to several publicly available implementations of other biclustering methods. The methods we consider include ‘BiMax’, ‘Cheng-and-Church’, ‘Plaid’, ‘Quest’, ‘Spectral’ and ‘Xmotif’, as implemented in the ‘biclust’ package in R⁵, with references in [9, 10, 11, 12] and [M27] and [M40]. We also consider the ‘QuBic’ and ‘UniBic’ algorithms [13, 14], as implemented by the authors⁶. We also consider the ‘BicMix’ algorithm [M30], as implemented by the authors⁷.

It is important to note that comparing these various biclustering methods is not entirely straightforward; by and large, each of these methods was designed to solve a different kind of biclustering problem. To briefly summarize, some of these other methods try and group the rows/columns of the data-matrix into disjoint (or overlapping) biclusters, others try and find large biclusters containing mostly ‘large’ values, others try and decompose the data-matrix into various ‘factors’, and others try and find more subtle patterns involving gene-expression. To make matters more complicated, the performance of many of these methods can be implementation dependent, relying on internal parameters and search-strategies that might be hard-coded into the software.

⁵available at <https://r-forge.r-project.org/>

⁶see <https://github.com/maqin2001/qubic> and <https://github.com/zhenjiawang157/UniBic>

⁷see <http://beehive.cs.princeton.edu/sw/BicMix-Code-for-distribution.zip>

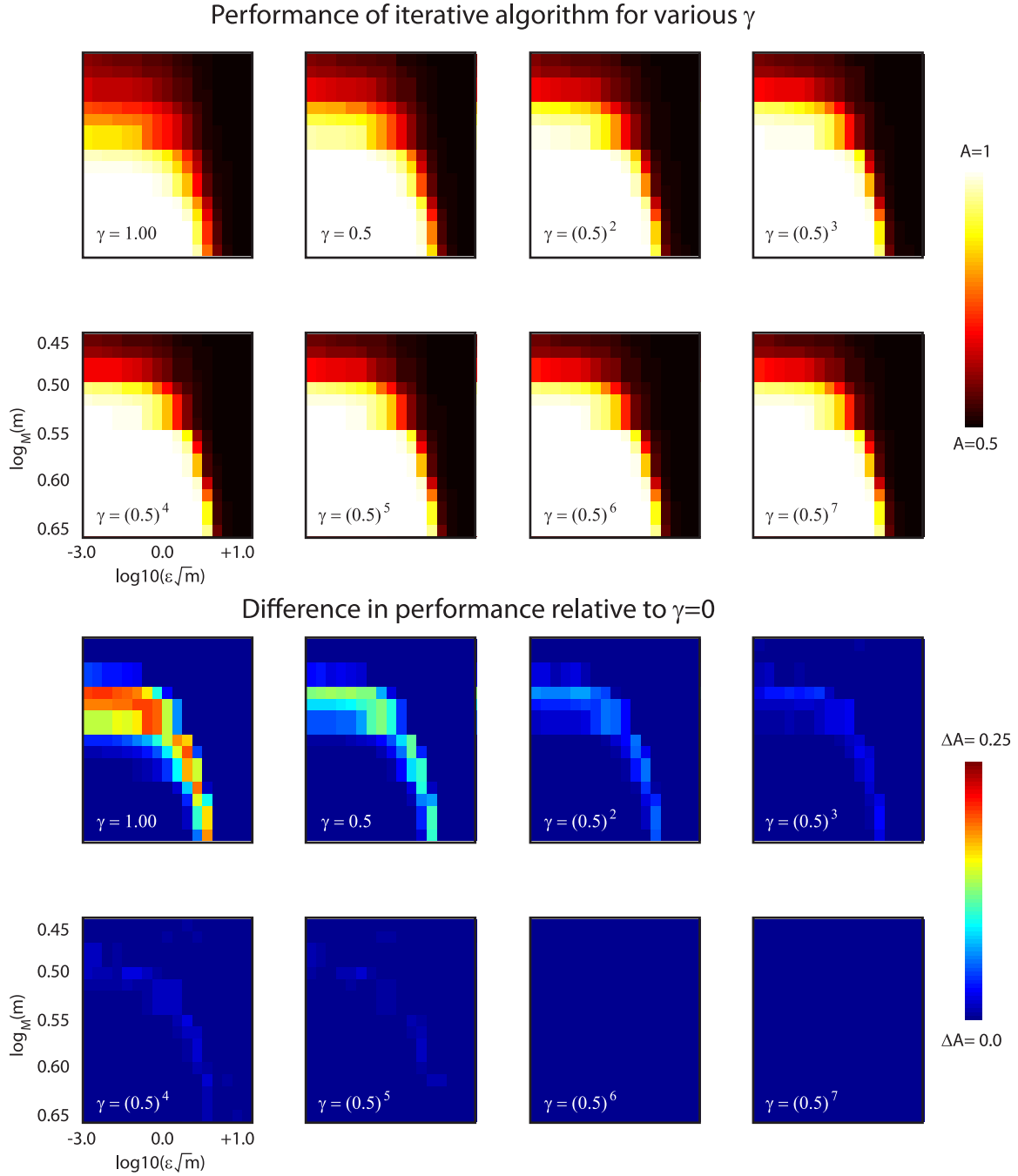


Figure 32: Performance of our algorithm on the planted-bicluster problem with $M = N = 1024$ for various choices of the iteration-parameter γ . The parameter γ is the fraction of rows and columns eliminated at each step of the algorithm. When $\gamma = 1$ the loop-scores are only calculated once and then used to order all the rows and columns. When $\gamma \rightarrow 0$ we eliminate only a single row or column with each iteration. Note that the performance of our algorithm improves as γ decreases, and for values of $\gamma \lesssim 0.5^5$ the algorithm seems to ‘converge’, in the sense that the performance of the algorithm approaches the $\gamma = 0$ case. In practice we notice little to no difference between our algorithm’s performance when $\gamma \lesssim 5\%$.

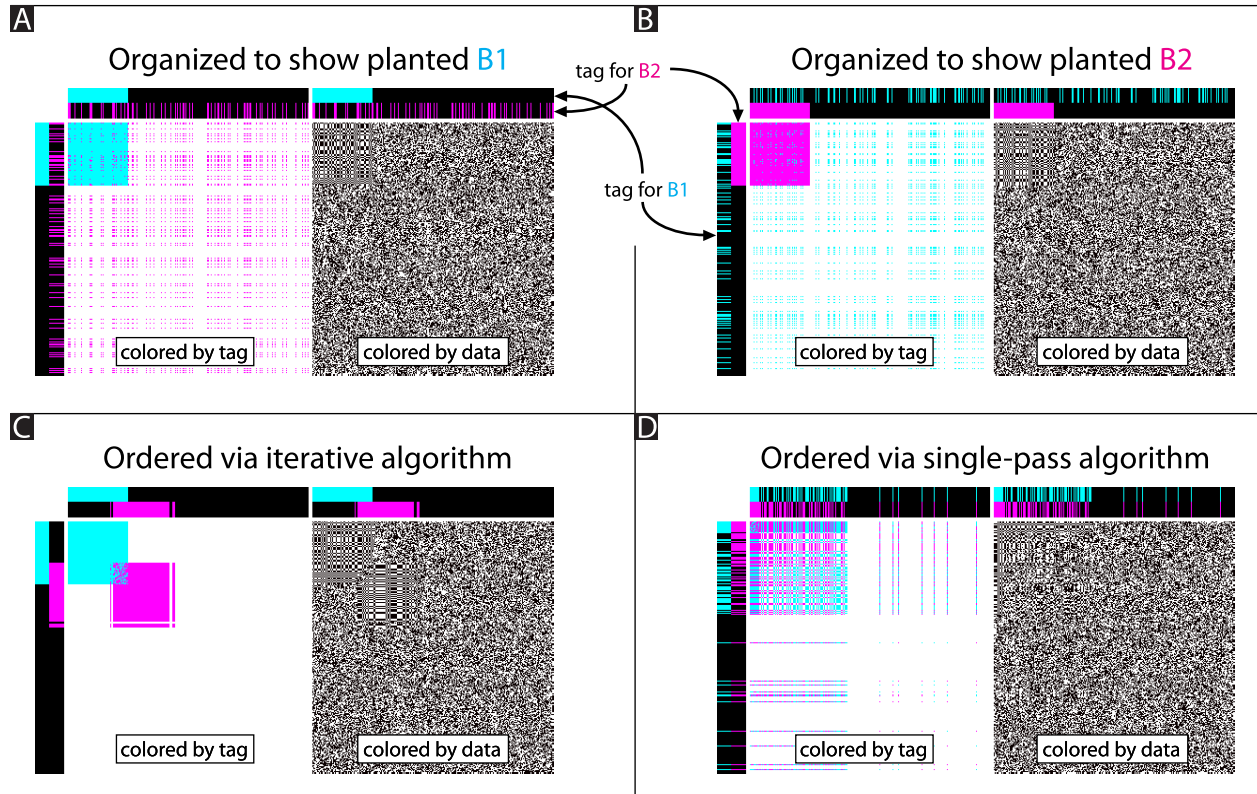


Figure 33: Demonstration of the focusing effects of an iterative algorithm. For this numerical experiment we generate a 256×256 data-matrix, within which we implant two biclusters $B1$ and $B2$. Both of these biclusters are square in shape and both are rank-1. The biclusters themselves overlap, both in terms of rows and columns. For illustration, each index (j, k) of the data-matrix is ‘tagged’ by $B1$ if the index is part of $B1$, and ‘tagged’ by $B2$ if the index is part of $B2$. We also calculate cumulative ‘row-tags’ for $B1$ and $B2$; these row-tags are 1 if the row or column is part of $B1$ or $B2$ respectively, and 0 otherwise. We similarly calculate column-tags for $B1$ and $B2$. In the figure shown we illustrate these tags in four different subplots. Subplots A and B organize the rows and columns of the data-matrix to reveal the planted biclusters $B1$ and $B2$, respectively. Subplots C and D organize the rows and columns based on the output of our iterative loop-score algorithm (bottom left) and the single-pass spectral-biclustering algorithm described in section 22 (bottom right). In each of the four subplots we show two copies of the data-matrix side by side; one colored by index-tag (cyan vs magenta for $B1$ vs $B2$) and the other colored by the actual data (black vs white for $+1$ vs -1). Along the sides and top of each subplot we draw vertical and horizontal strips indicating the row- and column-tags for $B1$ and $B2$. Note that, for the iterative algorithm in subplot C, rows and columns that are eliminated first show up in the bottom-right of the data-matrix, whereas the rows and columns retained the longest are placed in the upper-left. For the single-pass algorithm in subplot D, the rows and columns with the lowest scores are placed in the bottom-right, whereas those with the highest scores are placed in the upper-left. Note that the iterative algorithm not only detects both $B1$ and $B2$, but naturally separates them. This separation occurs because of the iteration; if a row or column of $B1$ is eliminated before any rows or columns of $B2$, the overall strength of the signal associated with $B1$ is reduced relative to $B2$, increasing the chance that another row or column of $B1$ will be eliminated before any rows or columns of $B2$. The same holds if a row or column of $B2$ were to be eliminated first; in this case the biclusters are symmetrically distributed, and the symmetry is broken randomly. Note also that the single-pass algorithm does not separate $B1$ and $B2$.

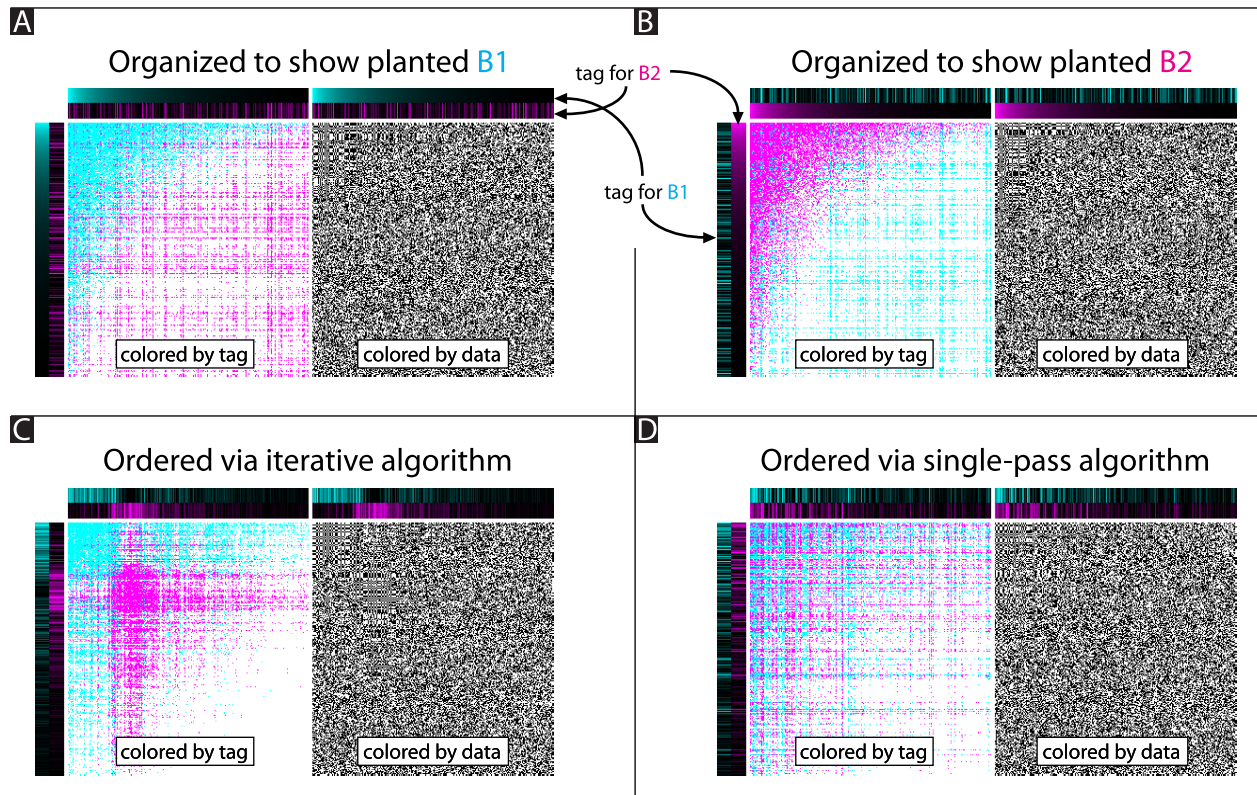


Figure 34: This figure is similar to Fig 33, except that the planted biclusters $B1$ and $B2$ are no longer perfect crisp squares. Instead, each bicluster is formed by constructing a loose collection of rows and columns, a few of which are tightly correlated with each other (forming the basis of a soft ‘core’), while others are less and less correlated. In this case we calculate the row- and column-tags for $B1$ and $B2$ so that they are proportional to the number of entries within that row or column that participate in $B1$ or $B2$ respectively. As shown in subplots A and B, the biclusters are quite noisy, and the boundaries of their cores are not clearly delineated. In these kinds of situations our iterative algorithm (subplot C) can still often separate the cores of the embedded structures, whereas a single-pass algorithm (subplot D) often intermingles them.

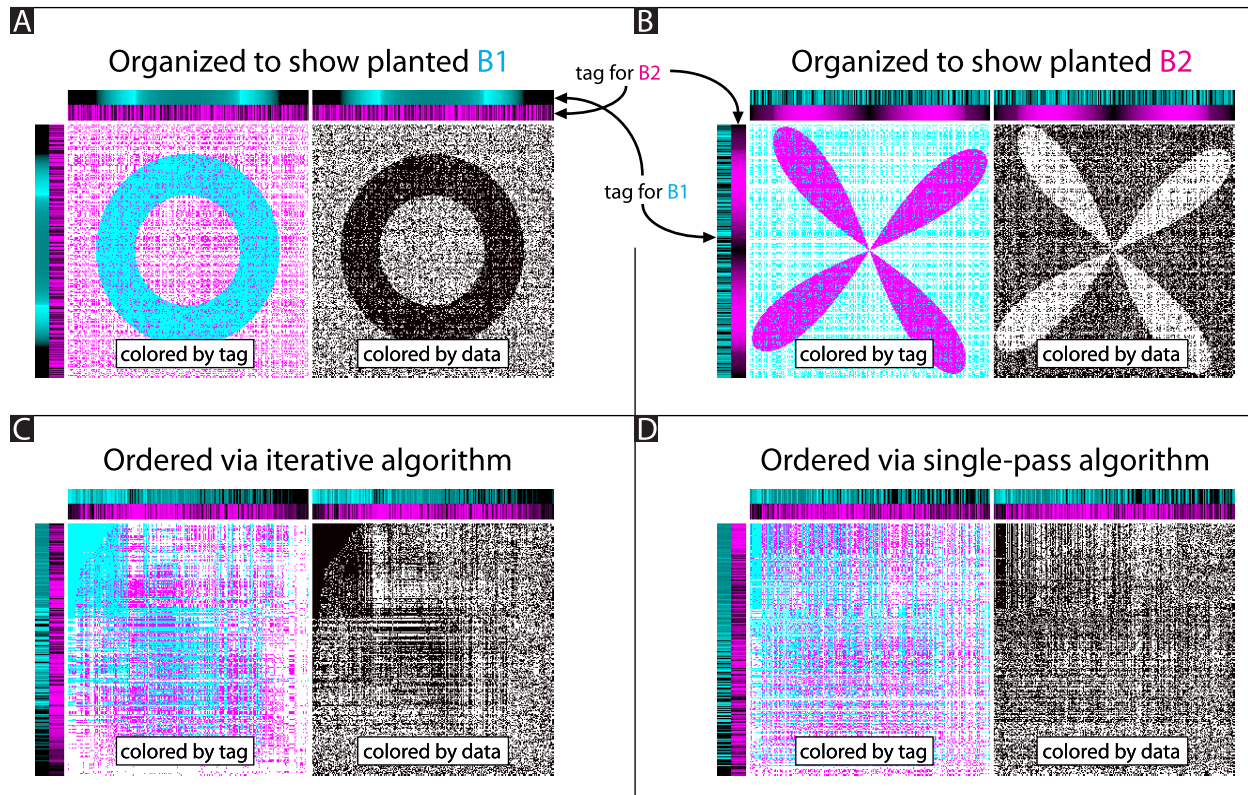


Figure 35: This figure is also similar to Fig 33, except that the planted structures $B1$ and $B2$ are no longer biclusters. These structures cannot easily be associated with a single list of row- and column-indices. Neither the iterative algorithm nor the single-pass algorithm can separate these structures from one another easily.

None of these other algorithms was designed with the same goals as our loop-counting algorithm; namely, to detect small correlated structures from within a large and noisy data-matrix. Consequently, we shouldn't expect these other algorithms to perform as well as our loop-counting algorithm when applied to the planted-bicluster problem described in section 6. We emphasize that this doesn't mean that these other methods are 'worse', or that our loop-counting method is 'better'. Indeed, the overarching goal of 'detecting structure' is so broad that no single algorithm (or implementation) will function well for every problem. There are almost surely problems out there for which these other methods outperform our loop-counting method.

Because of this discrepancy in design, we do not immediately try and directly apply these other biclustering methods to our planted-bicluster problem described in Fig 24. Instead, we first make one important change: Rather than implanting a low-rank bicluster into a random matrix D , we implant a rank-1 bicluster B that has been conditioned to have mostly positive entries (i.e., a 'rank-0' bicluster, as shown in Fig 1A of the main text). This modified 'rank-0' planted-bicluster problem is simpler than the original problem, and is more similar to some of the problems for which these other methods have been designed.

In terms of details, we start by constructing our planted-bicluster B to be numerical-rank $l = 1$ with noise ε (using the same methodology as before, described in section 5.1). Because B is close to rank-1, we can approximate $B \sim \vec{u}\vec{v}^T$ for some vectors \vec{u}, \vec{v} . Given this approximation, we then multiply B on the left by $\text{diag}(\text{sign}(\vec{u}))$, which flips the signs of each negative element of \vec{u} (as well as flipping the signs of the associated entries in the other left-principal-vectors of B). We also multiply B on the right by $\text{diag}(\text{sign}(\vec{v}))$, which flips the signs of each negative element of \vec{v} (and also flips the signs of the associated entries in the other right-principal-vectors of B). These two operations ensure that most of the entries of B are positive, but don't change the spectrum of B . The result is a bicluster B that has the same noise-level ε as before, but is almost entirely composed of positive entries when ε is small. We then proceed as before, implanting B into D (where the entries of D are chosen from a distribution with median 0) and then binarizing the result. An illustration of the kinds of matrices this procedure generates is shown in Fig 36.

Note that our loop-counting algorithm is completely insensitive to sign-changes applied to the rows (or columns) of D ; the performance of our loop-counting algorithm on this 'rank-0' planted-bicluster problem is identical to the results for the rank-1 planted-bicluster problem shown in Figs 26 and 29.

Results for the 'biclust' package:

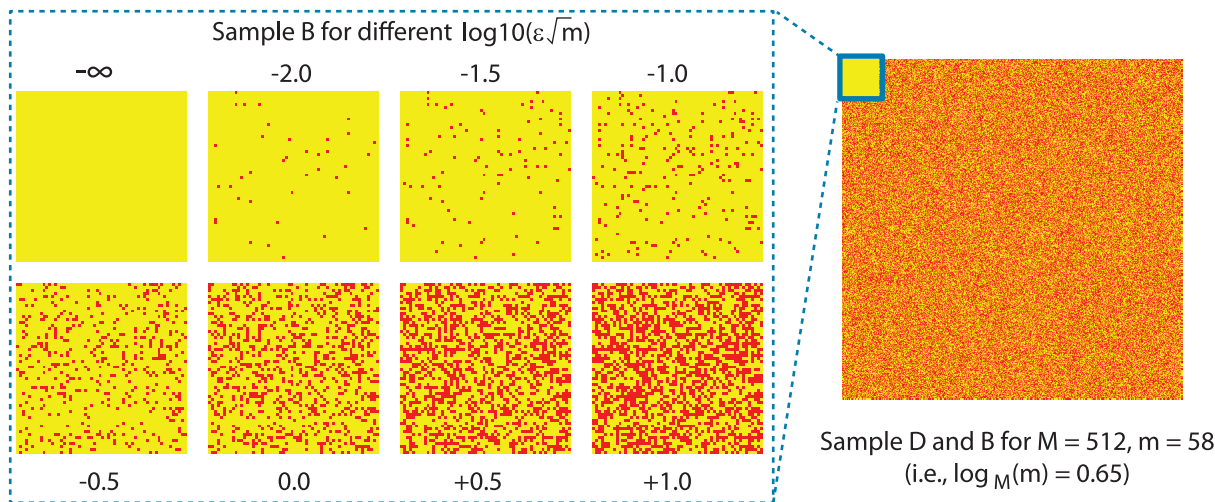


Figure 36: This figure illustrates the sample D and B matrices used for the numerical experiments shown in Fig 37. The format of this figure is similar to that of Fig 25. On the right we show a sample D of size $M = 512$, along with an implanted B of size $m = 58$, corresponding to $\log_M(m) = 0.65$. For illustration, we have rearranged the rows and columns of D to reveal B in the upper-left corner (see cyan box). This bicluster is large and noiseless; finding such a bicluster should be easy. Sample biclusters with varying noise-level are shown on the right.

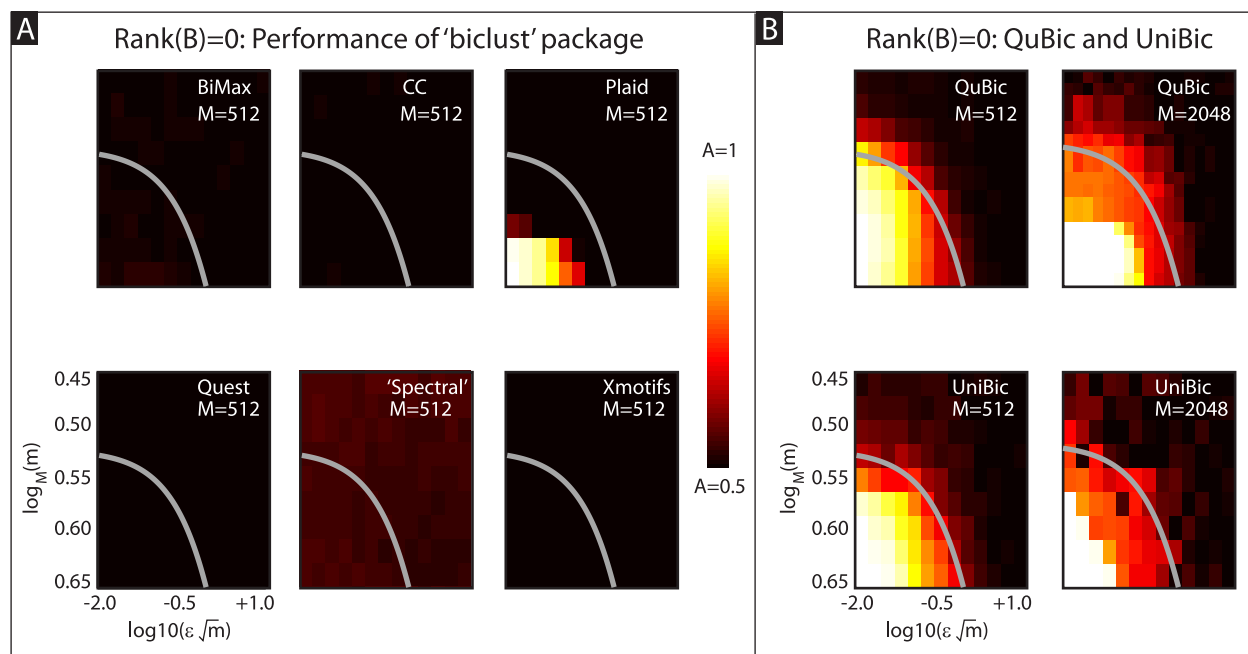


Figure 37: This figure illustrates the performance of several implementations of various biclustering algorithms (see text) on the ‘rank-0’ planted-bicluster problem described in section 7.4 and illustrated in Fig 36. In Panel-A we show the results associated with the ‘biclust’ package. The format of this figure is analogous to that of Fig 29A, with the exception that we fix $M = 512$, and we average over 64 trials for each choice of m and ε (i.e., for each pixel in the parameter-array). The thin grey line is the detection-threshold of our loop-counting algorithm (taken from Fig 26 and 29). Note that the results for the ‘Spectral’ method (from the biclust package) are artificially inflated (see text). In Panel-B we show the results for the ‘QuBic’ and ‘UniBic’ algorithms. For the $M = 512$ panel we average over 64 trials per pixel. Due to runtime constraints for the QuBic implementation, we only average over 8 trials per pixel when $M = 2048$. Similarly, due to runtime constraints for the UniBic implementation, we only show results for a single trial per pixel when $M = 2048$.

Shown in Fig 37A are the results for each of the publicly available algorithms listed above from the ‘biclust’ package; the format of this figure is analogous to that of Fig 29A, with the exception that we fix $M = 512$, and we average over 64 trials for each choice of m and ε .

In terms of details, we used the following parameter settings for each of the algorithms (with other parameters set to their default values). We tried to give each method a good chance of finding the planted-bicluster, and so we increased any available parameters associated with the number of internally generated random samples, number of iterations, etc:

BiMax: We set the ‘number’ parameter to 1.

Cheng and Church: We set the ‘number’ parameter to 1.

Plaid: We increase the ‘iter.startup’ parameter from the default of 5 to 200 and we increase the ‘iter.layer’ parameter from the default of 10 to 100.

Quest: We set the ‘number’ parameter to 1.

Spectral: We set the ‘normalization’ parameter to bistochastization (choosing the ‘irrc’ option instead for independent-rescaling of rows and columns did not seem to affect the performance).

Xmotifs: We increase the ‘nd’ parameter from the default of 10 to 100, and increase the ‘sd’ parameter from the default of 5 to 50. We also set the ‘number’ parameter to 1.

For all the algorithms aside from the ‘Plaid’ and ‘Spectral’ methods we were able to set the ‘number’ parameter to 1. Setting this ‘number’ parameter to 1 informed each method that only 1 bicluster was in the data, and asked each method to output only a single bicluster for each trial. For each trial, we used this single output-bicluster to order the rows of the original matrix D ; rows within the output-bicluster were ranked higher than the others. This row-ranking then allowed us to calculate the auc A_R , as described in Fig 24. We defined A_C analogously (using the columns), and then used $A = (A_R + A_C)/2$ as a metric of success for that trial. If the output-bicluster overlapped strongly with the planted-bicluster B , then A would be high for that trial (close to 1). On the other hand, if the output-bicluster did not overlap with B , then A would be low for that trial (close to 0.5).

Even though we could not set a ‘number’ parameter for the ‘Plaid’ method, this method only ever returned a single output-bicluster, and so we calculated ‘ A ’ for the ‘Plaid’ method in the same fashion.

On the other hand, the implementation of the ‘Spectral’ method returned a varying number of biclusters for each trial, typically ranging from 10 to 25 or more. To give this ‘Spectral’ method the benefit of the doubt, we calculated the A value for each of the biclusters it returned on any given trial, and used the maximum as the result for that trial. As a result of this procedure, our results for the ‘Spectral’ method are artificially inflated (i.e., typically, one of the 20 or so output biclusters for each trial would have a small overlap with the planted-bicluster B based purely on chance).

Note that, of all the algorithms in the ‘biclust’ package, only the implementation of the ‘Plaid’ algorithm was able to reliably detect the planted-bicluster across a subset of our numerical experiments. Nevertheless, the ‘biclust’ implementation of the ‘Plaid’ algorithm does not appear to succeed all that often near the detection-threshold of the loop-counting algorithm.

Results for ‘QuBic’ and ‘UniBic’:

Shown in Fig 37B are the results for the ‘QuBic’ and ‘UniBic’ algorithms. For each implementation of the ‘QuBic’ and ‘UniBic’ algorithms we used the default values, requesting a single output bicluster (i.e., the ‘-o1’ option). For the $M = 512$ panel we average over 64 trials per pixel.

Note that the implementations of these two algorithms tend to perform rather well when $M = 512$. Unfortunately, as shown in the $M = 2048$ panels to the far right, their performance appears to degrade as M increases. Due to runtime constraints for the QuBic implementation, we only average over 8 trials per pixel when $M = 2048$. Similarly, due to runtime constraints for the UniBic implementation, we only show results for a single trial per pixel when $M = 2048$.

Results for ‘BicMix’:

We also ran the same set of numerical experiments using the implementation of the ‘BicMix’ algorithm mentioned above. We used the default parameters suggested in the documentation file for this implementation (e.g., the number of factors was initialized to 50 via the ‘--nf 50’ option). We were not capable of finding any biclusters using this implementation across all our numerical experiments.

Results for the rank- l planted-bicluster problem:

Shown in Figs 38 and 39 are analogous results for the rank-1 and rank-2 planted-bicluster problems. Unsurprisingly, none of these other methods perform well on these more difficult planted-bicluster problems; as alluded to above, none of these other methods were designed with the goal of finding small correlated structures from within a large and noisy data-matrix.

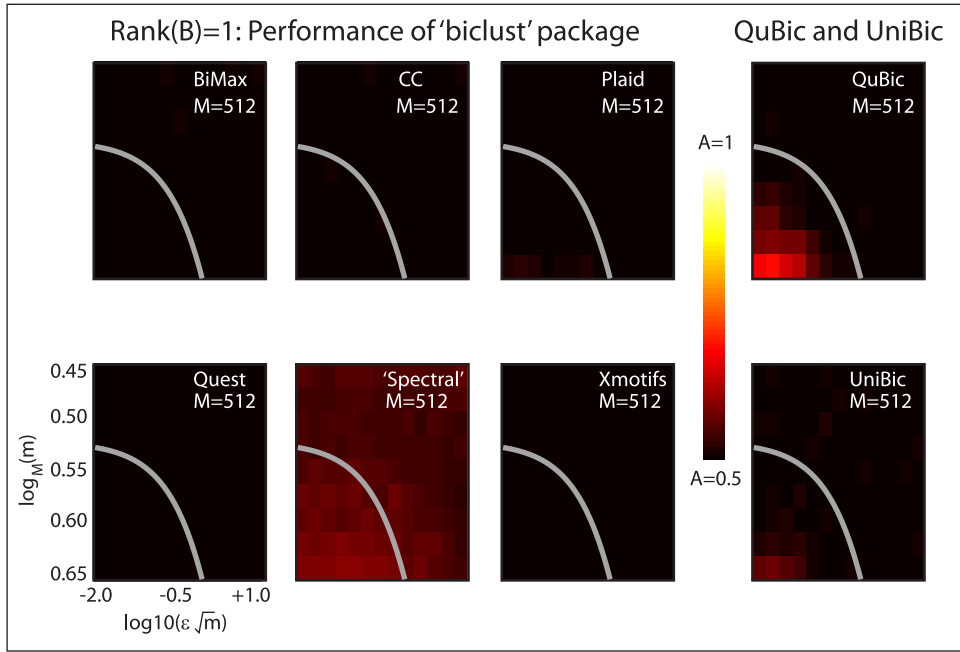


Figure 38: This figure illustrates the performance of several implementations of various biclustering algorithms on the rank-1 planted-bicluster problem. The format of this figure is analogous to that of Fig 37. The thin grey line is the detection-threshold of our loop-counting algorithm (taken from Fig 29). Note that the results for the ‘Spectral’ method (from the biclust package) are artificially inflated (see text).

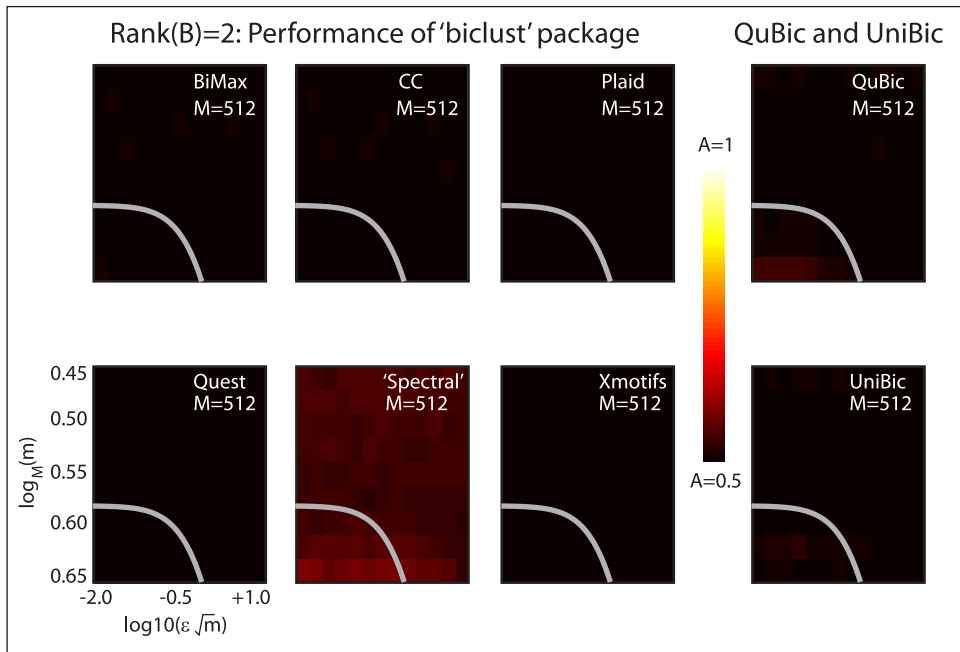


Figure 39: This figure illustrates the performance of several implementations of various biclustering algorithms on the rank-2 planted-bicluster problem. The format of this figure is analogous to that of Fig 37. The thin grey line is the detection-threshold of our loop-counting algorithm (taken from Fig 30). Note that the results for the ‘Spectral’ method (from the biclust package) are artificially inflated (see text).

8 Correcting for Controls: D and X

When analyzing genetic data, there often exist large subsets of genes which are correlated across large subsets of the population (e.g., genes related to development). Because these common genetic signatures correspond to massive biclusters within the data, the algorithm presented in section 2 is not typically very informative; the largest biclusters include large swathes of the patient population, and are not usually of interest. More typically, one is interested in finding biclusters which are restricted to a subset of the case-patient-population – say patients with a given disease – and which exhibit correlations which are *not* found in the control-population (e.g., patients without that disease). Such a bicluster might pinpoint genes that are linked to disease mechanisms and are useful for diagnosis.

In terms of notation, we will refer to:

- The $M_D \times N$ case-matrix D : The entry $D(j, k)$ is a real-number recording the gene-expression value for the k^{th} -gene of the j^{th} -case-patient.
- The $M_X \times N$ control-matrix X : The entry $X(j, k)$ is a real-number recording the gene-expression value for the k^{th} -gene of the j^{th} -control-patient.

Ideally, we would like to find a subset of n_B genes, as well as a subset of m_D case-patients such that the $m_D \times n_B$ bicluster lies entirely within D , and exhibits a low-rank structure which is *not* significantly exhibited within the M_X controls (i.e., the bicluster is case-specific, and does not extend into X). In a more realistic scenario, we are willing to consider a subset of n_B genes which is correlated across m_B case-patients and m_X control-patients, as long as m_D/M_D is significantly greater than m_X/M_X . In both scenarios we would like to ignore any non-specific biclusters which include a comparable fraction of cases and controls (e.g., for which $m_D/M_D = m_X/M_X$) or which are over-represented amongst the controls (e.g., $m_D/M_D < m_X/M_X$). These goals can be achieved by slightly modifying ‘Step-0’ and ‘Step-1’ of the D -only algorithm above. In Step-0 we now need to binarize both D and X . In Step-1 we need to calculate the scores in a slightly different way, which we’ll explain now.

As before, we look at all loops within the data, focusing on a given row j for the moment. There will be two types of loops that involve row- j : (i) loops that are contained within D , and (ii) loops that travel through X . If row- j were part of a bicluster which was restricted to D , then we would expect row- j to participate in an abundance of rank-1 loops of type-(i), but not of type-(ii). Based on this intuition, we score the first type of loop positively if it is rank-1 and negatively if it is rank-2. In addition, we score the second type of loop the other way; negatively if it is rank-1 and positively if it is rank-2 (see Fig 40).

Collectively, this amounts to defining subscores Z_{ROW}^{DD} and Z_{ROW}^{DX} , and then combining the results to form Z_{ROW} :

$$\begin{aligned} [Z_{\text{ROW}}^{DD}]_j &= \frac{1}{(M_D - 1)N(N - 1)} \sum_{\substack{j \text{ fixed, } j' \in D, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k} D_{j'k'} D_{jk'}, \\ [Z_{\text{ROW}}^{DX}]_j &= \frac{1}{M_X N(N - 1)} \sum_{\substack{j \text{ fixed, } j' \in X; \\ k' \neq k}} D_{jk} X_{j'k} X_{j'k'} D_{jk'}, \\ [Z_{\text{ROW}}]_j &= [Z_{\text{ROW}}^{DD}]_j - [Z_{\text{ROW}}^{DX}]_j. \end{aligned}$$

The scaling factors ensure that the collection of loops within D and the collection of loops passing through X are weighted equally (e.g., the Z_{ROW} does not change if M_X is doubled simply by duplicating each control-patient).

Note that, in the equations above, we write our sums so that each loop involves two distinct row- and column-indices (i.e., we specifically exclude collapsed-loops). We have also chosen the normalization factors to equal the number of terms in each sum (also excluding collapsed-loops). We have adopted this convention so that the row-scores $[Z_{\text{ROW}}^{DD}]_j$ and $[Z_{\text{ROW}}^{DX}]_j$ will each depend monomially on the size of each bicluster they are a part of. If we were to include collapsed-loops in our sum, then this dependence would no longer be monomial, with $[Z_{\text{ROW}}^{DD}]_j$ and $[Z_{\text{ROW}}^{DX}]_j$ each containing a small constant term. While the inclusion or exclusion of a constant term won’t affect the control-correction discussed in this section, it will be convenient for the covariate-correction described later on. Thus, to maintain uniformity throughout our methodology, we exclude collapsed-loops here as well.

Because we have excluded collapsed-loops and normalized appropriately, the row-scores above have the following properties. First, when considering large random matrices the row-scores $[Z_{\text{ROW}}^{DD}]_j$ and $[Z_{\text{ROW}}^{DX}]_j$ will have mean 0 and variances $\sim 2/(M_D N^2)$ and $\sim 2/(M_X N^2)$, respectively. Now let’s imagine that D and X contain a bicluster B that is a ε -error rank- l bicluster of size $(m_D + m_X) \times n_B$, with m_D rows in D (corresponding to the row-set J_B), m_X rows in X , and n_B columns from D (corresponding to the column-set K_B). The final row-score $[Z_{\text{ROW}}]_j$ is designed so that, if $j \notin J_B$ then its average row-score will be 0. However, if $j \in J_B$, then its average row-score will be $\sim (m_D/M_D - m_X/M_X)(n_B/N)^2(2g - 1)$, where g is the fraction of loops in B that are rank-1. We refer to this as the

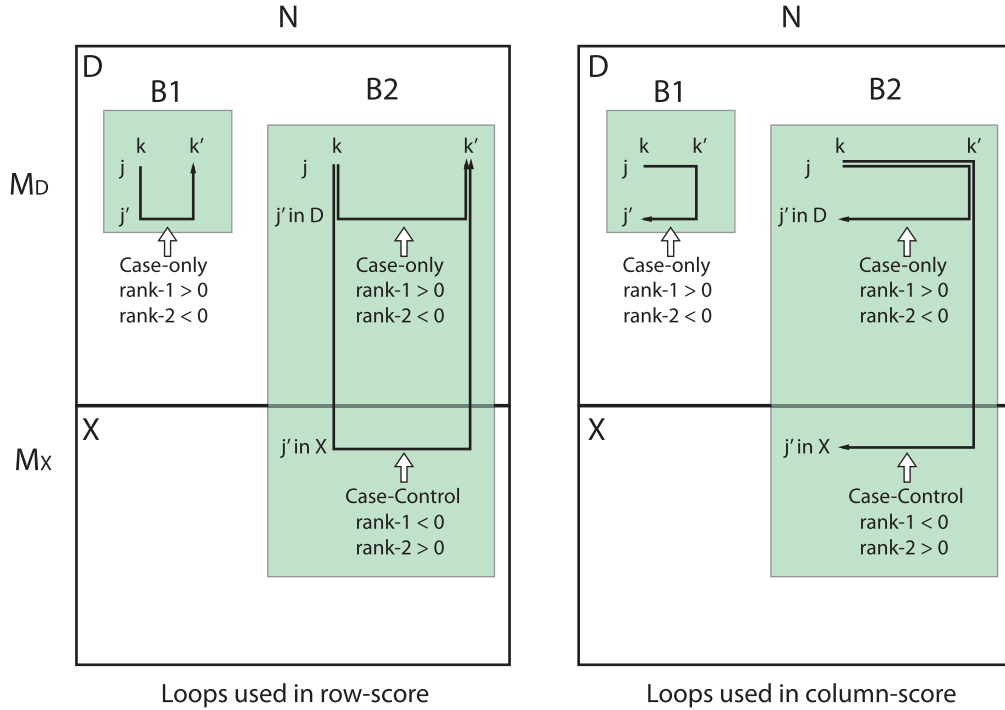


Figure 40: This figure illustrates the considerations associated with correcting for controls. In both panels we illustrate a case-matrix D (top) as well as a control-matrix X (bottom). Some loops associated with the row-scores are shown on the left panel, whereas some loops associated with the column-scores are shown on the right-panel. To aid in the discussion, we imagine that our data has within it two low-rank biclusters; $B1$, which is restricted to the cases, and $B2$, which straddles both the cases and controls. We would like to design scores that focus on case-specific biclusters such as $B1$, while ignoring non-specific biclusters such as $B2$. One way to do this is to divide the loops within our data-matrix into case-only loops and case-control loops. The former remain within D , whereas the latter extend from D to X and back again. Both biclusters $B1$ and $B2$ contain an abundance of rank-1 case-only loops (some of which are indicated in the figure). However, $B2$ also contains an abundance of rank-1 case-control loops, whereas only half the case-control loops starting from within $B1$ are rank-1. We take advantage of this observation when designing our row-score. For any given row j within D we accumulate all loops passing through that row; case-only loops contribute positively if they are rank-1, and negatively if they are rank-2; case-control loops have their sign switched, and contribute negatively if they are rank-1 and positively if they are rank-2. We adopt the same convention when designing our column-score. For any given column k within D we accumulate all loops passing through that column; case-only loops contribute positively only if they are rank-1, and case-control loops contribute positively only if they are rank-2.

scaled row-signal of B . Note that, if B is constructed as described in section 5.1, then g will equal $g_{l,\varepsilon,m_D+m_X}$. Note also that, if j participates in multiple independent and disjoint biclusters, then its average row-score will be the sum of the scaled row-signals of each of those biclusters.

The column-scores are calculated similarly:

$$\begin{aligned} [Z_{\text{COL}}^{DD}]_k &= \frac{1}{(N-1)M_D(M_D-1)} \sum_{\substack{j',j \in D, j' \neq j; \\ k \text{ fixed}, k' \neq k}} D_{jk} D_{jk'} D_{j'k'} D_{j'k}, \\ [Z_{\text{COL}}^{DX}]_k &= \frac{1}{(N-1)M_D M_X} \sum_{\substack{j \in D, j' \in X; \\ k \text{ fixed}, k' \neq k}} D_{jk} D_{jk'} X_{j'k'} X_{j'k}, \\ [Z_{\text{COL}}]_k &= [Z_{\text{COL}}^{DD}]_k - [Z_{\text{COL}}^{DX}]_k. \end{aligned}$$

These scores are designed so that the following properties hold. First, when considering large random matrices, $[Z_{\text{COL}}^{DD}]_k$ and $[Z_{\text{COL}}^{DX}]_k$ will have mean 0 and variances $\sim 2/(M_D^2 N)$ and $\sim 1/(M_D M_X N)$, respectively. Second, when considering an embedded bicluster B like the one described above (i.e., spanning column-subset K_B), the average column-score of a column $k \notin K_B$ will be 0. On the other hand, the average column-score of a column $k \in K_B$ will be equal to the scaled column-signal of B , which is $\sim (m_D/M_D)(m_D/M_D - m_X/M_X)(n_B/N)(2g-1)$. Again, if k participates in multiple independent and disjoint biclusters, then its average column-score will be the sum of the scaled column-signals of each of those biclusters.

Note that the signal produced by a case-only bicluster will be the same (on average) as the signal produced in our original D -only algorithm, but the signal produced by a non-specific bicluster (i.e., including a comparable fraction of cases and controls) will be demoted to 0. Consequently, these scores allow our algorithm to ignore any non-specific biclusters while still focusing on the biclusters which include a larger relative fraction of cases than controls. Note also that, if a row or column participates in multiple non-independent biclusters (e.g., potentially overlapping), then the overall score for that row or column will depend on the relative strengths of the various contributing signals and whether or not those signals cancel one another out. A simple example along these lines is shown in Figs 41 and 42, where we consider a variation on the planted-bicluster problem involving both a planted case-specific bicluster and a planted non-specific bicluster.

In terms of notation, it is convenient for us to introduce the ‘rescaled sum’ $\tilde{\Sigma}$ to denote a sum which is scaled by a factor equal to the total number of summands. With this notation our row- and column-scores become:

$$\begin{aligned} [Z_{\text{ROW}}^{DD}]_j &= \tilde{\Sigma}_{\substack{j \text{ fixed}, j' \in D, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k} D_{j'k'} D_{jk'}, & [Z_{\text{ROW}}^{DX}]_j &= \tilde{\Sigma}_{\substack{j \text{ fixed}, j' \in X; \\ k' \neq k}} D_{jk} X_{j'k} X_{j'k'} D_{jk'}, \\ [Z_{\text{COL}}^{DD}]_k &= \tilde{\Sigma}_{\substack{j', j \in D, j' \neq j; \\ k \text{ fixed}, k' \neq k}} D_{jk} D_{jk'} D_{j'k'} D_{j'k}, & [Z_{\text{COL}}^{DX}]_k &= \tilde{\Sigma}_{\substack{j \in D, j' \in X; \\ k \text{ fixed}, k' \neq k}} D_{jk} D_{jk'} X_{j'k'} X_{j'k}. \end{aligned}$$

As in the D -only case, these scores involve only a few matrix-matrix multiplications, and can be easily updated using only matrix-vector multiplications after rows or columns of D are removed. For example, when actually computing these scores one can use:

$$\begin{aligned} [Z_{\text{ROW}}^{DD}]_j &= \frac{1}{(M_D-1)N(N-1)} \left\{ [DD^\top DD^\top]_{jj} - N(N+M_D-1) \right\}, \\ [Z_{\text{ROW}}^{DX}]_j &= \frac{1}{M_X N(N-1)} \left\{ [DX^\top XD^\top]_{jj} - M_X N \right\}, \text{ and} \\ [Z_{\text{COL}}^{DD}]_k &= \frac{1}{(N-1)M_D(M_D-1)} \left\{ [D^\top DD^\top D]_{kk} - M_D(M_D+N-1) \right\}, \\ [Z_{\text{COL}}^{DX}]_k &= \frac{1}{(N-1)M_D M_X} \left\{ [D^\top DX^\top X]_{kk} - M_D M_X \right\}. \end{aligned}$$

In summary, this control-correction is designed to allow our loop-counting algorithm to focus on low-rank biclusters that include a disproportionately large number of cases, while ignoring low-rank biclusters which extend across a similar fraction of cases and controls. Consequently, we expect this control-correction to be particularly useful when searching for biclusters that comprise subsets of genes that are correlated within the cases, but uncorrelated across the controls.

We conclude by pointing out that this control-correction may not always be appropriate. For example, if one is interested in searching for ‘differentially-expressed’ biclusters (i.e., where each gene is either high across the cases and

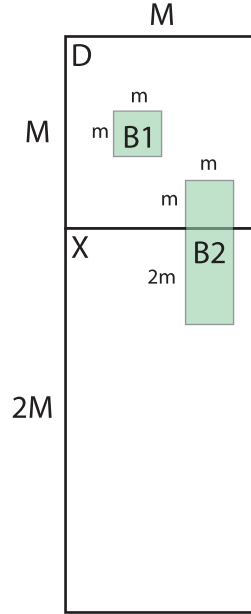


Figure 41: This figure illustrates the setup for the numerical experiments shown in Fig 42. These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving both case- and control-patients (i.e., D as well as X). For each numerical experiment we let D be a large $M \times M$ random matrix with entries chosen independently from a distribution with median 0. We construct X in a similar fashion, except that X is $2M \times M$ (to simulate the fact that real data sets often have more controls than cases). For each experiment we implant within the case-matrix D a small $m \times m$ bicluster $B1$ that is rank-1 with error ε . The rows and columns of $B1$ are chosen at random. We also implant a second rank-1 error- ε ‘red-herring’ bicluster $B2$ within both the case-matrix D as well as the control-matrix X . To implant $B2$ we first randomly choose m columns K and m cases J^D from D , as well as $2m$ controls J^X within X . We then construct a $3m \times m$ bicluster $B2$ that is rank-1 with error ε , and implant the top third of $B2$ into the chosen cases J^D and the bottom two-thirds of $B2$ into the chosen controls J^X (in each case using the chosen K columns). Our illustration is somewhat misleading, as $B1$ and $B2$ are not typically disjoint. Because the row- and column-subsets for these biclusters are chosen randomly, there is typically a small overlap between $B1$ and $B2$. For the purposes of this numerical experiment entries within this overlap are randomly assigned to one of the biclusters. For each numerical experiment we calculate the average auc A_1 and A_2 for biclusters $B1$ and $B2$, respectively. Our goal is to find the former, despite the masking effect of the latter.

low across the controls, or vice-versa), then one should not use the control-correction described in this section. This is because such a differentially-expressed structure is – by definition – a low-rank structure which extends across both the cases and the controls; the control-correction described above is deliberately designed to ignore such structures. Instead, if one wishes to search for ‘differentially-expressed’ biclusters, we recommend a modified (and simpler) version of our loop-counting algorithm – described below in section 15.2.

9 Correcting for Categorical Covariates:

Another important feature of real genetic data is that there are typically many important covariates to consider. For example, the different patients may be drawn from different categories (e.g., different studies performed at different locations or times). In these situations it is important to ‘correct’ for this categorical-covariate, and find biclusters that are in some sense independent of the covariate-category. In terms of notation, we’ll separate our case- and control-matrices by category, using a post-script i . We’ll refer to:

- The $M_{D_i} \times N$ case-matrix D_i : The entry $D_i(j, k)$ is a real-number recording the gene-expression value for the k^{th} -gene of the j^{th} -case-patient from category- i .
- The $M_{X_i} \times N$ control-matrix X_i : The entry $X_i(j, k)$ is a real-number recording the gene-expression value for the k^{th} -gene of the j^{th} -control-patient from category- i .

For simplicity we’ll discuss a situation with case-patients only, each belonging to one of $I_{\text{cat}} = 3$ covariate-categories. The number of case-patients in category $i \in \{1, 2, 3\}$ will be denoted by M_{D1} , M_{D2} , and M_{D3} , respectively. The data for the case-patients will be recorded in data-matrices $D1$, $D2$, and $D3$, respectively (i.e., D_i is $M_{D_i} \times N$).

Performance of biclustering algorithm on case-control problem :

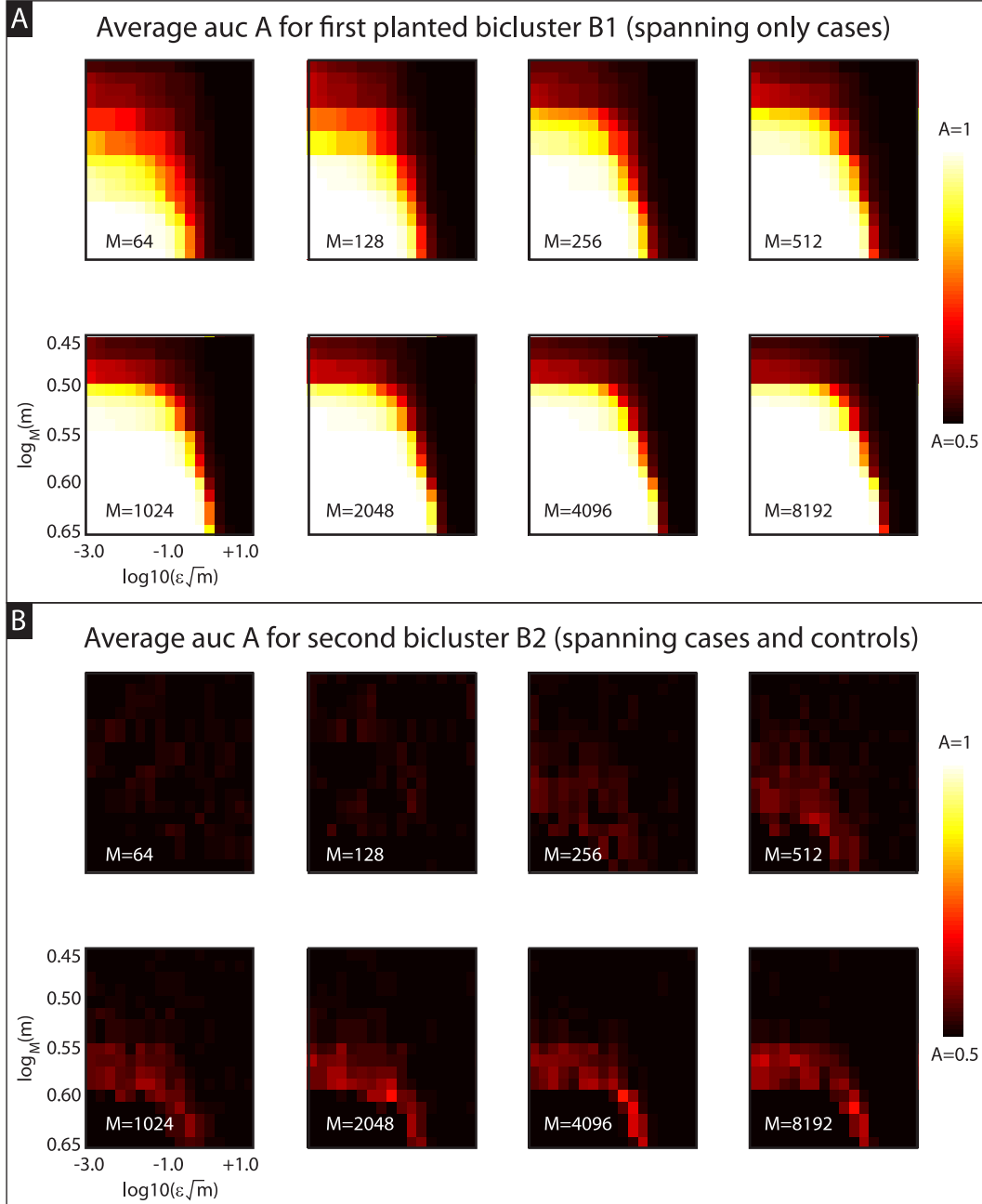


Figure 42: These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving both case- and control-patients (see Fig 41). In panel-A we show the trial-averaged value of A_1 as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. In panel-B we show the trial-averaged value of A_2 . While not quite as accurate as the D -only case (e.g., compare panel-A with Fig 26), our algorithm is still generally successful when $\log_{10}(\epsilon\sqrt{m}) \lesssim 0$ and $\log_M(m) \gtrsim 0.5$.

Our goal when presented with this data will be to find any biclusters that extend to encompass many of the covariate-categories. In an ideal scenario, if a bicluster B involves m_{D1} patients from $D1$, m_{D2} from $D2$ and m_{D3} in $D3$, then we would like m_{D1}/M_{D1} to equal both m_{D2}/M_{D2} and m_{D3}/M_{D3} (i.e., m_{Di}/M_{Di} should be the same for all i). In a more realistic scenario we are willing to consider biclusters for which the m_{Di}/M_{Di} are different, so long as the bicluster is well represented in many of the categories. We typically want to ignore biclusters that are limited to a single category, or for which $m_{Di}/M_{Di} \sim 0$ for most of the i .

These considerations imply that there are several ways to correct for categorical-covariates, and the user needs to specify exactly what kind of correction is desired. In our case we expect a single number: an integer $I_{\text{req}} \in [1, \dots, I_{\text{cat}}]$. The value I_{req} indicates the minimum number of covariate-categories a bicluster must encompass in order to produce a signal. In other words, $I_{\text{cat}} - I_{\text{req}}$ corresponds to the maximum number of covariate-categories a bicluster can avoid while still producing a signal. In the simple example above, a choice of $I_{\text{req}} = 2$ will allow our algorithm to ignore biclusters which are limited to only a single covariate-category, while focusing on biclusters that encompass 2 or 3 of the covariate-categories.

In terms of actual calculation, we modify the score-calculation within Step-1 of our algorithm. For each pair of categories i and i' we calculate sub-scores $Z_{\text{ROW}}^{i'}$ and $Z_{\text{COL}}^{ii'}$ as follows:

$$\begin{aligned} \left[Z_{\text{ROW}}^{i'} \right]_{j \text{ from category } i} &= \widetilde{\sum}_{\substack{j \text{ fixed, } j' \in \text{category } i', \\ \text{ensuring that } j' \neq j \text{ if } i' = i; \\ k' \neq k}} D i_{jk} D i'_{j'k} D i'_{j'k'} D i_{jk'} \\ \left[Z_{\text{COL}}^{ii'} \right]_k &= \widetilde{\sum}_{\substack{j \in \text{category } i, \\ j' \in \text{category } i', \\ \text{ensuring that } j' \neq j \text{ if } i' = i; \\ k \text{ fixed, } k' \neq k}} D i_{jk} D i_{j'k'} D i'_{j'k} D i'_{j'k}. \end{aligned}$$

The row sub-scores $\left[Z_{\text{ROW}}^{i'} \right]_j$ measure the signal for row- j coming from loops traveling through category- i' . The column sub-scores $\left[Z_{\text{COL}}^{ii'} \right]_k$ measure the signal for column- k which comes from loops traveling through categories i and i' .

Note that, similar to section 8, we have excluded collapsed-loops from these row and column sub-scores. While not strictly necessary, this exclusion is convenient – ensuring that the subscores for any row or column depend monomially on the size of any biclusters that row or column participates in. If we were to include the contribution of collapsed-loops, then each subscore would also include a constant term depending on the size of the categories involved.

Once we have calculated the sub-scores above, we consider the collection $\left[Z_{\text{ROW}}^{i'} \right]_j$ for each j across all I_{cat} values of i' , and sort this collection in descending order. We then choose $[Z_{\text{ROW}}]_j$ to be element number I_{req} on this sorted list (i.e., the I_{req} -order-statistic). We'll represent this by the notation:

$$[Z_{\text{ROW}}]_j = \left[Z_{\text{ROW}}^{\#I_{\text{req}}} \right]_j$$

In the case of our simple example, we would have $I_{\text{cat}} = 3$ values within each such collection (e.g., $[Z_{\text{ROW}}^1]_j$, $[Z_{\text{ROW}}^2]_j$ and $[Z_{\text{ROW}}^3]_j$), and we would choose $[Z_{\text{ROW}}]_j$ to be the second largest value. We also consider the collection $\left[Z_{\text{COL}}^{ii'} \right]_k$ for each k across all $(I_{\text{cat}})^2$ values of i, i' , and sort this in descending order as well. We choose $[Z_{\text{COL}}]_k$ to be element number I_{req}^2 on this sorted list, i.e.,

$$[Z_{\text{COL}}]_k = \left[Z_{\text{COL}}^{\#I_{\text{req}}^2} \right]_k.$$

In terms of our simple example this collection would contain the 9 elements $\{ [Z_{\text{COL}}^{11}]_k, [Z_{\text{COL}}^{12}]_k, \dots, [Z_{\text{COL}}^{33}]_k \}$, and we would choose $[Z_{\text{COL}}]_k$ to be the 4th-largest element in this collection.

This procedure ensures that, in the limit as $M_{Di} \gg m_{Di} \gg 1$, any bicluster which extends to encompass at least I_{req} categories will contribute a signal to the row- or column-scores, whereas a bicluster which encompasses fewer than I_{req} categories will not. To elaborate, let's imagine an ε -error rank- l bicluster B comprising components Bi involving m_{Di} patients from Di (each associated with row-set J_{Bi}) and n columns associated with column-set K_B . Now the row sub-score $\left[Z_{\text{ROW}}^{i'} \right]_j$ for a row j in category i will, on average, be 0 if $j \notin J_{Bi}$. On the other hand, if $j \in J_{Bi}$ then the average row sub-score will be equal to the scaled row-signal coming from Bi' ; namely, $\left\{ (m_{Di}/M_{Di'}) (n_B/N)^2 (2g - 1) \right\}$, (with g equal to $g_{l,\varepsilon,m}$ where $m = \sum_i m_{Di}$ is the total number of rows in B).

Note that these average row sub-scores depend monomially on the ratios $m_{Di'}/M_{Di'}$ and n_B/N , scaling linearly with $m_{Di'}/M_{Di'}$. As long as $j \in J_{Bi}$, larger values of $\left[Z_{\text{ROW}}^{i'} \right]_j$ necessarily imply larger values of $m_{Di'}/M_{Di'}$ (i.e., a larger

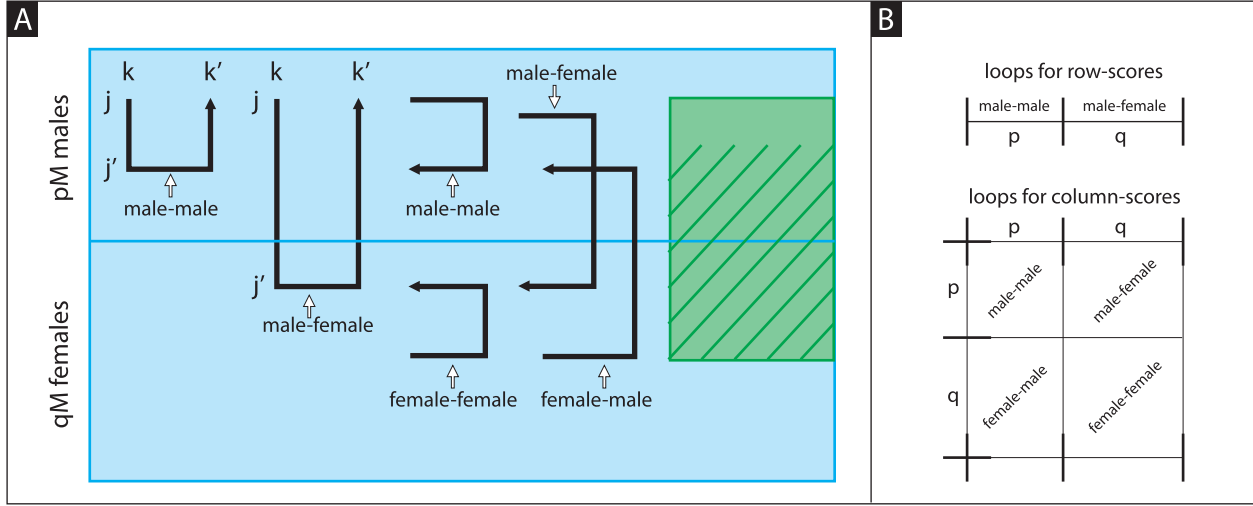


Figure 43: Correcting for a binary covariate (e.g., gender). In Panel-A we illustrate an $M \times N$ case-matrix D with pM males and qM females. Within this matrix, from left to right, we illustrate the two kinds of loops involved in the row-score for a male (row- j), and the four kinds of loops involved in the column-score for any given column. On the right side of D we illustrate an embedded bicluster B (green solid). This bicluster is not balanced across the covariates, having an abundance of males and a paucity of females (relative to the ratios p and q , respectively). The scores we devise in section 9 demote the signal produced by B to the signal that would be produced by the largest balanced bicluster within B . The size of this balanced cluster is indicated by one such representative, indicted via green hashed lines. Note that even though the signal produced by B has a strength demoted to that of its balanced component, this demoted signal affects the scores of all rows and columns of B (including those that are not included in the hashed bicluster we chose to draw). In the top of Panel-B we show the rough proportion of male-male and male-female loops expected to contribute to the row-score of any male within a balanced bicluster. In the bottom of Panel-B we show the rough proportion of male-male, male-female, female-male and female-female loops expected to contribute to the column-score of any column within a balanced bicluster.

relative-size for the bicluster within category i'). By sorting the set of $\left[Z_{\text{ROW}}^{i'} \right]_j$ (across i') and choosing the I_{req} -largest, we ensure that the final row-score for j is large only if *both* j participates in the bicluster Bi' and the bicluster spans at least I_{req} categories.

A similar story holds for the column sub-scores. Specifically, the column sub-score $\left[Z_{\text{COL}}^{i'} \right]_k$ will, on average, be 0 for any column $k \notin K_B$. On the other hand, if $k \in K_B$ then the average column sub-score will be equal to the scaled column-signal involving Bi and Bi' : namely, $\{(m_{D_i}/M_{D_i})(m_{D_{i'}}/M_{D_{i'}})(n_B/N)(2g-1)\}$. Similar to the row sub-scores, these column sub-scores depend monomially on the ratios m_{D_i}/M_{D_i} , $m_{D_{i'}}/M_{D_{i'}}$ and n_B/N , scaling linearly with each. By sorting the set of $\left[Z_{\text{COL}}^{i'} \right]_k$ (across i, i') and choosing the I_{req}^2 -largest, we ensure that the final col-score for k is large only if *both* k participates in the bicluster B and the bicluster spans at least I_{req} categories.

One important caveat to keep in mind when correcting for categorical-covariates is that choices of I_{req} close to I_{cat} are rather restrictive. If, for example, one were to choose $I_{\text{req}} = I_{\text{cat}}$, then our algorithm would search only for biclusters that span every single category. In this situation a bicluster will only be found if it passes the size-detection thresholds for all the categories (i.e., if $m_{D_i} \gtrsim \sqrt{M_{D_i}}$ for each i). This requirement is unlikely to be met – even by large biclusters – if some of the covariate-categories do not have many representatives (i.e., if M_{D_i} were small for some i). Put another way, choosing $I_{\text{req}} \approx I_{\text{cat}}$ diminishes the power of our algorithm down to that of the smallest category (i.e., as if M were the smallest of the M_{D_i}). Choices of I_{req} in between 1 and I_{cat} are less restrictive, and allow biclusters to be reliably detected even if some of the M_{D_i} are small.

9.1 Interpretation when $I_{\text{cat}} = 2$ and $I_{\text{req}} = 2$

If we consider a binary covariate (e.g., gender), then it is natural to choose $I_{\text{req}} = 2$: a bicluster must extend across both covariate-categories to be considered. If we analyze this situation we see that our algorithm above can be interpreted in a slightly simpler way: each bicluster produces a signal in accordance with the largest balanced bicluster contained within it. This scenario is illustrated in Fig 43.

Carrying forward the notation used above, let's imagine that the first category corresponds to, say, males and the second to females. Let's define $M = M_{D_1} + M_{D_2}$ to be the total number of patients. A bicluster B involving m_{D_1} males

and m_{D2} females will have $m = m_{D1} + m_{D2}$ patients in total. When considering the row-score for male- j in this bicluster, there are two possible kinds of loops which could add a signal: male-male loops and male-female loops (the former involves another row j' from the m_{D1} males in B , whereas the latter involves a row j' from the m_{D2} females in B). If the bicluster B were balanced across gender, then these two kinds of loops should occur with fractions p and q , respectively, where $p = (M_{D1} - 1) / (M - 1)$, and $q = M_{D2} / (M - 1)$ are the fraction of other males and females, respectively (note that we are excluding row- j from the males). However, if B is not balanced then one type of loop will be more abundant than expected, and the other type of loop will be less abundant. In this situation it is natural to construct a row-score which limits the signal of B to the signal produced by the largest balanced bicluster that fits within B – that is, to find out which is the lower of $(m_{D1} - 1) / (M_{D1} - 1)$ or m_{D2} / M_{D2} , and to ignore the surplus of abundant loops coming from the other gender. This can be accomplished by taking, e.g.,

$$[Z_{\text{ROW}}^1]_j = \sum_{\substack{j \text{ fixed, } j' \text{ male, } j' \neq j \\ k' \neq k}} D1_{jk} D1_{j'k} D1_{j'k'} D1_{jk'}, \text{ and } [Z_{\text{ROW}}^2]_j = \sum_{\substack{j \text{ fixed, } j' \text{ female, } \\ k' \neq k}} D1_{jk} D2_{j'k} D2_{j'k'} D1_{jk'}$$

$$[Z_{\text{ROW}}]_j = \min \left\{ [Z_{\text{ROW}}^1]_j \frac{1}{p}, [Z_{\text{ROW}}^2]_j \frac{1}{q} \right\}.$$

Apart from scaling, this is equivalent to our row-score constructed using $I_{\text{req}} = 2$. The row-scores for any given female are similar, and involve female-female and female-male loops. Note that, while our method demotes the signal of B to that of its largest balanced component, this demoted signal affects the row-scores of all rows of B (regardless of gender).

Regarding the column-scores: for any column k there are four kinds of loops which could add a signal: male-male, male-female, female-male and female-female. Ideally, these four different kinds of loops should occur with fractions α_{11} , α_{12} , α_{21} , α_{22} , respectively, where $\alpha_{11} = M_{D1} (M_{D1} - 1) / \bar{M}^2$, $\alpha_{12} = \alpha_{21} = M_{D1} M_{D2} / \bar{M}^2$, and $\alpha_{22} = M_{D2} (M_{D2} - 1) / \bar{M}^2$, with $\bar{M}^2 = M^2 - M_{D1} - M_{D2}$. If B is not balanced we might find that the different kinds of loops will be more or less abundant than expected. We can construct a column-score which limits the signal of B to that produced by the largest balanced bicluster within B via:

$$[Z_{\text{COL}}^{11}]_k = \sum_{\substack{j', j \text{ male, } j' \neq j \\ k \text{ fixed, } k' \neq k}} D1_{jk} D1_{j'k'} D1_{j'k} D1_{j'k}, \text{ and } [Z_{\text{COL}}^{12}]_k = \sum_{\substack{j \text{ male, } j' \text{ female, } \\ k \text{ fixed, } k' \neq k}} D1_{jk} D1_{j'k'} D2_{j'k'} D2_{j'k}$$

$$[Z_{\text{COL}}^{21}]_k = \sum_{\substack{j \text{ female, } j' \text{ male, } \\ k \text{ fixed, } k' \neq k}} D2_{jk} D2_{j'k'} D1_{j'k'} D1_{j'k}, \text{ and } [Z_{\text{COL}}^{22}]_k = \sum_{\substack{j', j \text{ female, } j' \neq j \\ k \text{ fixed, } k' \neq k}} D2_{jk} D2_{j'k'} D2_{j'k'} D2_{j'k}$$

$$[Z_{\text{COL}}]_k = \min \left\{ [Z_{\text{COL}}^{11}]_k \frac{1}{\alpha_{11}}, [Z_{\text{COL}}^{12}]_k \frac{1}{\alpha_{12}}, [Z_{\text{COL}}^{21}]_k \frac{1}{\alpha_{21}}, [Z_{\text{COL}}^{22}]_k \frac{1}{\alpha_{22}} \right\}.$$

Again, apart from scaling, this is equivalent to the column-score we constructed above using $I_{\text{req}} = 2$. An example of this simple correction is given in Figs 44 and 45.

10 Correcting for Continuous Covariates:

For certain applications each patient is associated with a high-dimensional continuous-covariate. For example, in genome-wide association studies, each patient is often equipped with an N_T -dimensional vector of ‘mds-components’. These mds-components are usually formed by applying dimension-reduction techniques (i.e., multi-dimensional-scaling) to the genetic data of the patients. The euclidean distance between patients in this N_T -dimensional mds-space is often used as a proxy for the genetic similarity of those patients’ ancestors [M37, M38]. Consequently, when attempting to control for genetic-ancestry, we are not interested in biclusters involving patients that are concentrated together in mds-space. Instead, we would like to ignore these mds-specific biclusters and focus on the biclusters which involve patients that are widely dispersed across mds-space.

For simplicity we’ll discuss a situation with case-patients only, and we’ll focus on what we’ll later refer to as a ‘2-sided’ covariate correction. In terms of notation, we’ll let $M = M_D$, and refer to the continuous-covariate via:

- The $M \times N_T$ case-covariate matrix T : The entry $T(j, t)$ is the t^{th} -coefficient of the continuous-covariate-vector for the j^{th} -case-patient.

Each row of T contains the $1 \times N_T$ vector of covariate-coefficients for that patient. In Step-0 we’ll binarize both D and T . After binarization, we define the base row- and column-scores in Step-1 as though we had no covariates:

$$[Z_{\text{ROW}}^{\text{base}}]_j = \widetilde{\sum}_{\substack{j \text{ fixed, } j' \in D, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k} D_{j'k'} D_{jk'},$$

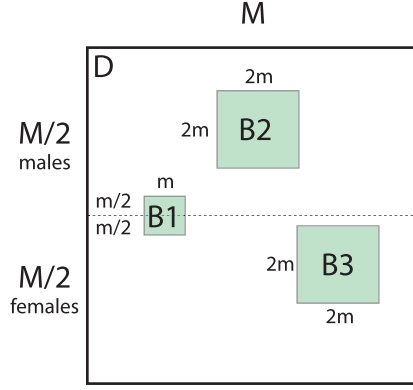


Figure 44: This figure illustrates the setup for the numerical experiments shown in Fig 45. These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving two categorical covariates (i.e. male and female). For each numerical experiment we let D be a large $M \times M$ random matrix with entries chosen independently from a distribution with median 0. Half the patients are male and half are female. For each experiment we implant a small balanced $m \times m$ bicluster $B1$ that is rank-1 with error ε and contains $m/2$ males and $m/2$ females. The rows and columns of $B1$ are chosen at random. We also randomly implant two larger rank-1 error- ε ‘red-herring’ biclusters $B2$ and $B3$ of size $2m \times 2m$, except that we restrict the rows of $B2$ to the males and the rows of $B3$ to the females. The illustration here is slightly misleading; the biclusters $B1, B2$ and $B3$ are not typically disjoint. Because the row- and column-subsets for each bicluster are drawn randomly, there is typically a small overlap between each pair of biclusters. For this simple numerical experiment we assign entries within any overlap to one of the overlapping biclusters at random. For each numerical experiment we calculate the average auc A_1, A_2 and A_3 for biclusters $B1, B2$ and $B3$, respectively. Our goal is to find the first, despite the masking effect of the latter two.

$$[Z_{\text{COL}}^{\text{base}}]_k = \widetilde{\sum}_{\substack{j', j \in D, j' \neq j; \\ k \text{ fixed}, k' \neq k}} D_{jk} D_{j'k'} D_{j'k} D_{jk'}$$

We then define additional subscores for each covariate coefficient:

$$[Z_{\text{ROW}}^{[t]}]_j = \widetilde{\sum}_{\substack{j' \in D, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k'} T_{j't} D_{j'k} D_{jk'} T_{jt}, \text{ for each } t \in \{1, \dots, N_T\},$$

$$[Z_{\text{COL}}^{[t]}]_k = \widetilde{\sum}_{\substack{j', j \in D, j' \neq j; \\ k \text{ fixed}, k' \neq k}} D_{jk} T_{jt} D_{j'k'} D_{j'k} T_{j't} D_{jk'}, \text{ for each } t \in \{1, \dots, N_T\}.$$

Note that, as in sections 8 and 9, we exclude collapsed-loops for these subscores. As before, this exclusion is convenient – ensuring that the subscores have a mean of 0 and depend monomially on the size of any biclusters they participate in.

We then combine these subscores to form the row- and column-scores as follows:

$$[Z_{\text{ROW}}^{[T]}]_j^2 = \frac{1}{\kappa^2} \widetilde{\sum}_t [Z_{\text{ROW}}^{[t]}]_j^2, \quad [Z_{\text{COL}}^{[T]}]_k = \frac{1}{\kappa^2} \widetilde{\sum}_t [Z_{\text{COL}}^{[t]}]_k,$$

$$[Z_{\text{ROW}}]_j^2 = [Z_{\text{ROW}}^{\text{base}}]_j^2 - [Z_{\text{ROW}}^{[T]}]_j^2, \quad [Z_{\text{COL}}]_k = [Z_{\text{COL}}^{\text{base}}]_k - [Z_{\text{COL}}^{[T]}]_k,$$

where $[x] = \max(0, x)$, and κ^2 is a parameter that depends on N_T (we’ll define κ^2 more carefully below).

The main idea behind this construction is to ensure that (i) a bicluster B with rows drawn uniformly from covariate-space produces a signal close to the original signal it would have produced without any covariate-correction, and (ii) a bicluster B with rows drawn from only one side of covariate-space will produce no signal on average. This is accomplished through the covariate-corrected scores $[Z_{\text{ROW}}^{[T]}]_j^2$ and $[Z_{\text{COL}}^{[T]}]_k$. These covariate-corrected scores will be proportional to the norm-squared of the average continuous-covariate vector accumulated across the rows of any biclusters that intersect row- j or column- k , respectively (see Fig 46). These covariate-corrected scores will typically have a small magnitude if the biclusters within the data are balanced with respect to the continuous-covariates, and a large magnitude otherwise.

To see why this might be the case, let’s imagine a situation where B is an $m \times n$ bicluster within D involving row-subset J_B and column-subset K_B , and let’s imagine that the covariates associated with J_B are drawn isotropically from \mathbb{R}^{N_T} . With this construction B is a balanced bicluster; it should have a high score.

Performance of biclustering algorithm with categorical-covariates

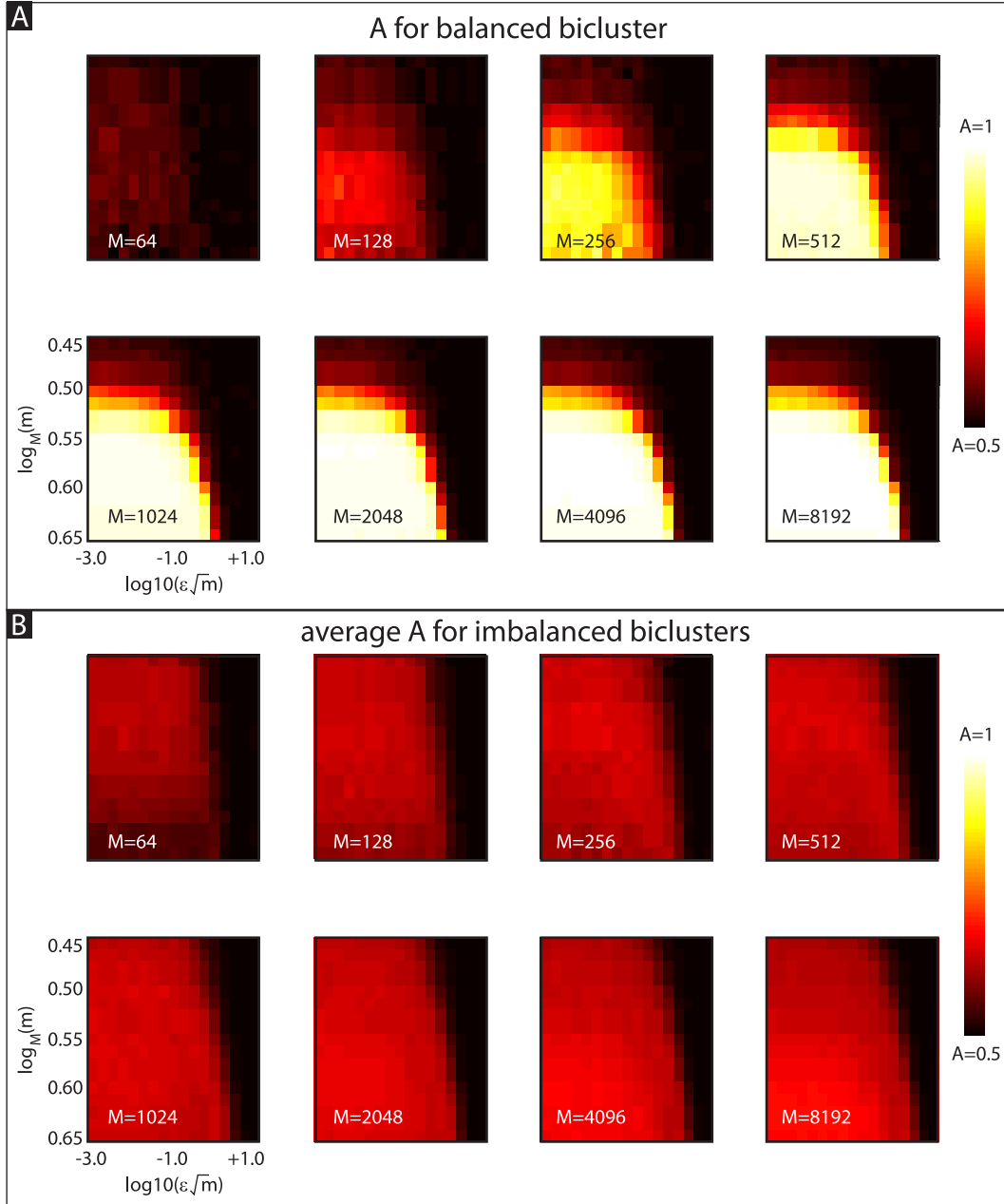


Figure 45: These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving two categorical covariates (i.e. male and female). See Fig 44. In panel-A we show the trial-averaged value of A_1 as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. In panel-B we show the trial-averaged value of $(A_2 + A_3)/2$. While not quite as accurate as the D -only case (e.g., compare panel-A with Fig 26), our algorithm is still generally successful when $\log_{10}(\epsilon\sqrt{m}) \lesssim 0$ and $\log_M(m) \gtrsim 0.5$, provided M is sufficiently large.

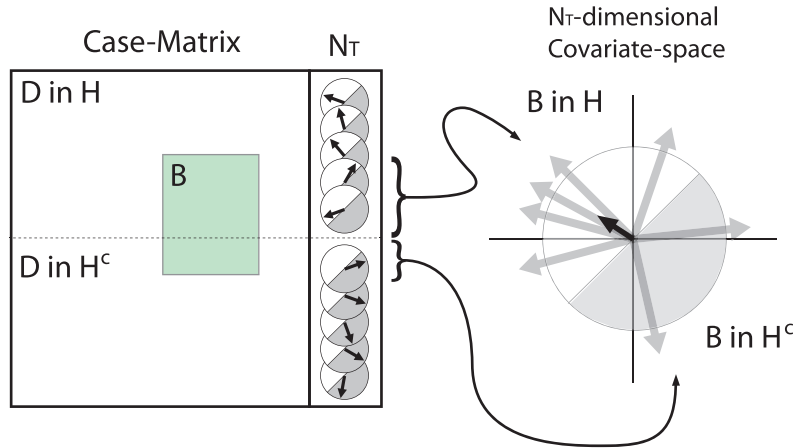


Figure 46: This figure illustrates the main idea behind our continuous-covariate correction described in section 10. For this scenario we assume that we are given a case-matrix D (shown on the left). Moreover, we'll assume that each patient (i.e., row of D) is associated with a continuous-covariate-vector of dimension $N_T = 2$. For illustration, a fraction of the rows of T are represented as unit-vectors in \mathbb{R}^2 . For presentation purposes, we'll imagine that the continuous-covariate-space is divided into two half-spaces, H (indicated by the white semicircle) and H^c (grey semicircle). The matrix D has been organized so that the rows with continuous-covariates drawn from H are placed on top, whereas the rows with continuous-covariates drawn from H^c are placed on the bottom. Within D we've planted a bicluster B . Note that this bicluster isn't perfectly balanced with respect to the continuous-covariate; a disproportionate number of rows of B are drawn from H , with only a minority drawn from H^c . We can measure this imbalance by collecting all the continuous-covariate-vectors associated with the rows of B . A fraction of the covariate-vectors associated with B are illustrated on the right-side of the data-matrix using large grey arrows. By averaging all these covariate-vectors, we obtain the smaller dark black vector: the covariate-average. If B were balanced with respect to the continuous-covariates, we would expect this covariate-average to be close to 0 in magnitude. However, because B is imbalanced, the magnitude of the covariate-average is not quite 0. If B were more imbalanced (e.g., entirely drawn from H or entirely drawn from H^c), then the magnitude of the covariate-average would be even larger. As one may guess, the expected magnitude of this average depends both on the level of imbalance within B , as well as the (fixed) dimension N_T of covariate-space. Our covariate-correction scheme described in section 10 modifies the loop-score to account for the magnitude of this covariate-average, demoting the score of any biclusters that are imbalanced in covariate-space.

We expect the base row-score for each row- $j \in J_B$ to have a signal proportional to $(m/M)(n/N)^2(2g-1)$, where g is the fraction of rank-1 loops in B . Because the covariates associated with J_B are isotropic, the row subscores $[Z_{\text{ROW}}^{[t]}]_j$ will be roughly proportional to the vector

$$\vec{\zeta}_t = \sum_{j' \in J_B} T_{j't},$$

which has coefficients drawn from $\mathcal{N}(0, m) = \rho\sqrt{m}$ (i.e., the number of positive and negative entries $T_{j't}$ should roughly cancel out). Note that for each index t ,

$$\zeta_t^2 = \sum_{j, j' \in J_B} T_{jt}T_{j't},$$

which will be drawn from $N(m, 2m^2) = m + \rho\sqrt{2m}$, implying that $\|\vec{\zeta}\|^2$ will be close to a small multiple of $N_T m$. Because the value of $\|\vec{\zeta}\|^2$ is $O(N_T m)$, rather than $O(N_T m^2)$, the overall row-score for $j \in J_B$ will be on the same order as the base-row-score. Indeed, if the base-score produced by B is sufficiently large then, for $j \in J_B$, we can ignore the contribution of loops not contained within B , and we should expect:

$$\begin{aligned} [Z_{\text{ROW}}^{\text{base}}]_j^2 &\sim \frac{m^2 n^4}{M^2 N^4} (2g-1)^2, \text{ and} \\ [Z_{\text{ROW}}^{[T]}]_j^2 &\sim \frac{1}{\kappa^2} \sum_t \widetilde{\zeta}_t \left[\zeta_t \cdot (1/M)(n/N)^2(2g-1) \right]^2 \sim \frac{1}{\kappa^2} \frac{O(m) \cdot n^4}{M^2 N^4} (2g-1)^2, \text{ implying} \\ [Z_{\text{ROW}}]_j^2 &\sim [Z_{\text{ROW}}^{\text{base}}]_j^2 \text{ for each } j \in J_B. \end{aligned}$$

A similar story holds for the column-scores. The base column-score for each $k \in K_B$ is proportional to the signal $(m/M)^2(n/N)(2g-1)$. Because the covariates associated with B are drawn isotropically from \mathbb{R}^{N_T} , the column-subscores will be roughly proportional to the vector

$$\vec{\xi}_t = \sum_{j, j' \in J_B, j \neq j'} T_{jt}T_{j't} \sim (\zeta_t)^2 - m,$$

which has coefficients drawn from $N(0, 2m^2) = \rho\sqrt{2m}$. Note that $\sum \vec{\xi}_t \sim \|\vec{\zeta}\|^2 - N_T m = O(N_T m)$, rather than $O(N_T m^2)$. Thus if the base-score produced by B is sufficiently large then, for $k \in K_B$, we can ignore the contribution of loops not contained within B , and we should expect:

$$\begin{aligned} [Z_{\text{COL}}^{\text{base}}]_k &\sim \frac{m^2 n}{M^2 N} (2g-1), \text{ and} \\ [Z_{\text{COL}}^{[T]}]_k &\sim \frac{1}{\kappa^2} \sum_t \widetilde{\xi}_t \cdot (1/M^2)(n/N)(2g-1) \sim \frac{1}{\kappa^2} \frac{O(m) \cdot n}{M^2 N} (2g-1), \text{ again implying} \\ [Z_{\text{COL}}]_k &\sim [Z_{\text{COL}}^{\text{base}}]_k \text{ for each } k \in K_B. \end{aligned}$$

Now let's consider a situation where the covariates associated with B are not drawn isotropically from \mathbb{R}^{N_T} , but are instead drawn from a fixed (but otherwise randomly-oriented) half-space in \mathbb{R}^{N_T} . With this construction B will be an imbalanced bicluster; its scores should be low.

To be more precise, let us define the hemisphere in \mathbb{R}^{N_T} as $H = \{\vec{x} \in \mathbb{R}^{N_T} \mid \|\vec{x}\| = 1 \text{ and } x_1 > 0\}$. Letting $Q \in \mathbb{R}^{N_T \times N_T}$ be a rotation of \mathbb{R}^{N_T} , define $\mu(Q)$ to be the uniform measure on Q . For any given Q , let T be an $m_B \times N_T$ random binary-matrix where – prior to binarization – each row is drawn independently from QH . Denote this measure on T by $\mu(T|Q)$. With this construction of T , the base row- and column-scores for B will stay the same, but the vector $\vec{\zeta}$ will be different. Because the covariates of B are now drawn from a half-space, the value $\|\vec{\zeta}\|^2$ will be $O(N_T m^2)$, rather than $O(N_T m)$. Indeed, there exists a constant of proportionality κ^2 such that the expected-value of $\|\vec{\zeta}\|^2$ will equal $\kappa^2 N_T m^2$:

$$\kappa^2(N_T, m) = \frac{1}{N_T m^2} \int \left(\int \|\vec{\zeta}\|^2 d\mu(T|Q) \right) dQ.$$

As shown in Fig 47, this constant κ^2 is only weakly dependent on m , but strongly dependent on N_T ; we can approximate κ^2 via $\kappa^2(N_T) = \kappa^2(N_T, \infty)$, using the values shown in Fig 47.

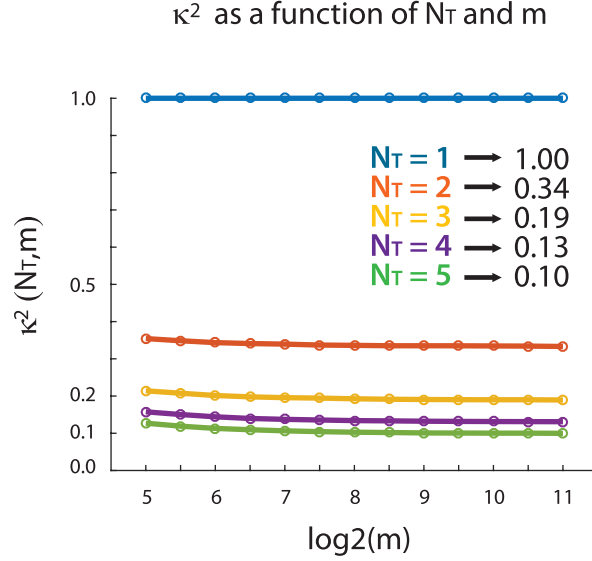


Figure 47: $\kappa^2(N_T, m)$ for the 2-sided continuous-covariate correction, as shown for $m = 32, \dots, 2048$, with different colors corresponding to different values of N_T (higher values of N_T correspond to lower values of κ^2). The large- m values of κ^2 are listed within the legend.

Given N_T we choose $\kappa^2 = \kappa^2(N_T)$ (e.g., if there were 2 continuous-covariate-components per patient we would choose $\kappa^2 \approx 0.34$). This then implies that the sub-scores for $j \in J_B$ or $k \in K_B$ will be:

$$\left[Z_{\text{ROW}}^{[T]} \right]_j^2 \sim \frac{m^2 n^4}{M^2 N^4} (2g-1)^2, \text{ and } \left[Z_{\text{COL}}^{[T]} \right]_k \sim \frac{m^2 n}{M^2 N} (2g-1).$$

Consequently, when the covariates of B are drawn from $\mu(T|Q)$, we expect that on average with respect to $\mu(Q)$ the signal produced by B will be.

$$\left[Z_{\text{ROW}} \right]_j^2 \sim 0 \text{ and } \left[Z_{\text{COL}} \right]_k \sim 0.$$

To summarize so far: we've introduced a method for calculating row- and column-scores which corrects for multi-dimensional continuous-covariates in the following sense. A bicluster B with its covariates drawn isotropically from the covariate-space \mathbb{R}^{N_T} will produce a signal comparable to what that bicluster would have produced in our original algorithm (i.e., a row-signal of $(m/M)(n/N)^2(2g-1)$ and a column-signal of $(m/M)^2(n/N)(2g-1)$). If the covariates associated with B are not isotropically distributed in \mathbb{R}^{N_T} , but are instead concentrated with respect to angle in \mathbb{R}^{N_T} , then the signal produced by B will be demoted. A bicluster B with its covariates drawn from only a half-space of \mathbb{R}^{N_T} will, on average, produce a signal of 0. A bicluster B with its covariates drawn from an even more narrow cone of \mathbb{R}^{N_T} will produce a negative signal. And finally, in the extreme scenario where each covariate of B is identical, the row-signal of B will be $-(1/\kappa^2 - 1)(m/M)(n/N)^2(2g-1)$ and the column-signal $-(1/\kappa^2 - 1)(m/M)^2(n/N)(2g-1)$.

In order for this kind of covariate-correction to be useful, the number of patients m_B in any planted bicluster B must be sufficiently large. Specifically, m_B must be large enough that the averages $\left[Z_{\text{ROW}}^{[T]} \right]_j^2$ and $\left[Z_{\text{COL}}^{[T]} \right]_k$ correctly reflect the covariate-distribution of B . This typically happens when $m_B \gtrsim O(2^{N_T})$ or so. If $m_B \lesssim 2^{N_T}$ then there are unlikely to be very many patients in each orthant of \mathbb{R}^{N_T} ; the averages $\left[Z_{\text{ROW}}^{[T]} \right]_j^2$ and $\left[Z_{\text{COL}}^{[T]} \right]_k$ are unlikely to be accurate estimates of the covariate-distribution of B , and are likely to be large even if B 's covariates are drawn from an isotropic distribution. Thus, if m_B is insufficiently large, even the signals produced by covariate-balanced biclusters will be demoted, and our algorithm will not perform well. Based on this observation, we see that the m_B required for covariate-estimation grows exponentially with N_T , but is not too large for small N_T ; for $N_T = 2$ the required m_B is roughly a few dozen or so (see Figs 48, 49 and 50).

Before we move on, we mention another kind of continuous-covariate correction; one that we'll refer to later as a '1-sided' correction.

$$\left[Z_{\text{ROW}}^{[t] \text{ 1s}} \right]_j = \sum_{\substack{j' \in D, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k} T_{j't} D_{j'k'} D_{jk'}, \text{ for each } t \in \{1, \dots, N_T\},$$

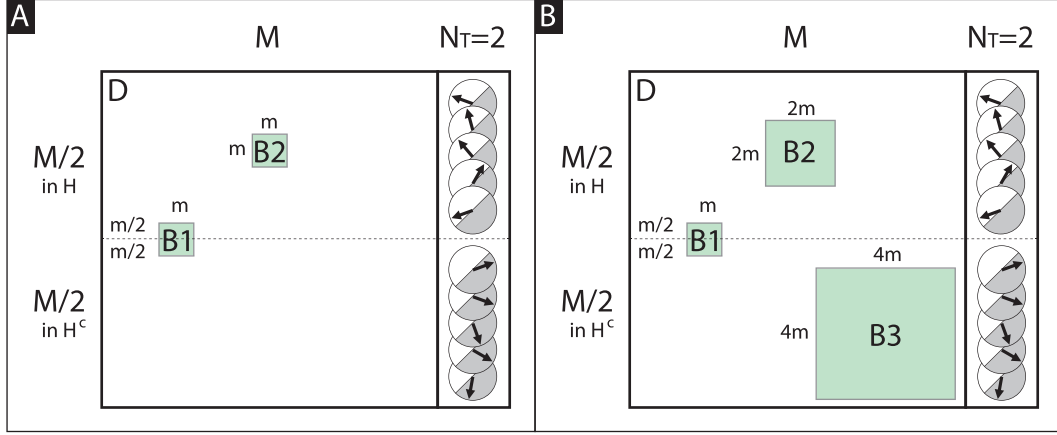


Figure 48: Panels A and B illustrate the setup for the numerical experiments shown in Figs 49 and 50, respectively. These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving continuous covariates with dimension $N_T = 2$. For each numerical experiment we let D be a large $M \times M$ random matrix with entries chosen independently from a distribution with median 0. We let T be an $M \times 2$ matrix of covariates with each row chosen randomly from an isotropic distribution on \mathbb{R}^2 . For illustration, the rows of T are represented as unit-vectors in \mathbb{R}^2 . Once we have chosen T , we select a randomly-oriented half-space H of \mathbb{R}^2 and then determine the subset J_H of rows of D which have covariates-vectors lying in H . In our illustration we depict one such randomly-oriented H with a white semicircle, and the complement H^c with a grey semicircle. We've organized the rows of D so that the rows in J_H are on top and the rows of J_{H^c} are on the bottom. In Panel-A we implant within D a small $m \times m$ bicluster $B1$ that is rank-1 with error- ε . To implant $B1$ we choose the m rows and columns at random. Note that $B1$ will typically be balanced with respect to the covariates; roughly half the rows of $B1$ will come from J_H , and half will come from J_{H^c} . We also implant within D a second rank-1 error- ε 'red-herring' bicluster $B2$ of size $m \times m$. We choose m columns at random from D , and m rows from J_H to implant $B2$. Note that bicluster $B2$ is not balanced with respect to the covariates. Note that our illustration is slightly misleading; $B1$ and $B2$ will not necessarily be disjoint. Rather, because they are randomly chosen, the row- and column-subsets of $B1$ and $B2$ will typically have a small overlap. For the purposes of this simple numerical experiment the entries in this overlap are randomly assigned to either $B1$ or $B2$. In Panel-B we implant $B1$ and $B2$ as before, except that $B2$ is of size $2m \times 2m$. We also implant a rank-1 error- ε $B3$ of size $4m \times 4m$ with rows restricted to J_{H^c} . For each numerical experiment we calculate the average auc A_1 for the bicluster $B1$, as well as the average auc A_2 for bicluster $B2$ (as well as A_3 for $B3$ if it exists). Our goal is to find the first, despite the masking effect of the latter. As shown in Fig 49, an $m \times m$ $B2$ does not compromise the effectiveness of our algorithm all that much. However, as shown in Fig 50, the presence of a larger $B2$ and $B3$ interfere with our algorithm's ability to successfully detect $B1$ when M is insufficiently large.

Performance of biclustering algorithm with continuous-covariates

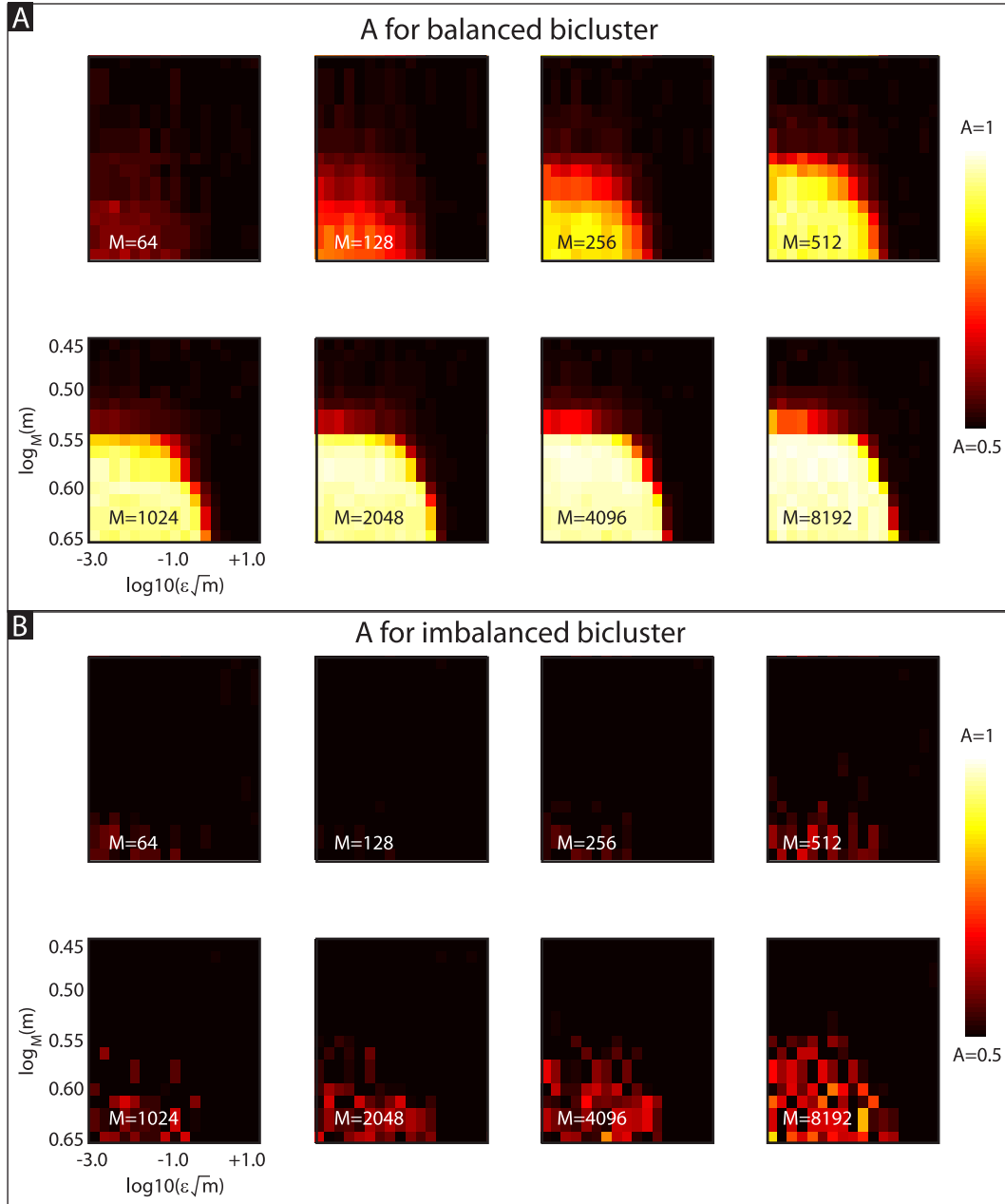


Figure 49: These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving continuous covariates with dimension $N_T = 2$ (see Fig 48A). In panel-A we show the trial-averaged value of A_1 as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. In panel-B we show the trial-averaged value of A_2 . The presence of B_2 slightly impacts our algorithm's performance (compare with Fig 26). Note that, when M is insufficiently large (in this case $\lesssim 64$), our algorithm focuses on B_2 rather than B_1 .

Performance of biclustering algorithm with continuous-covariates

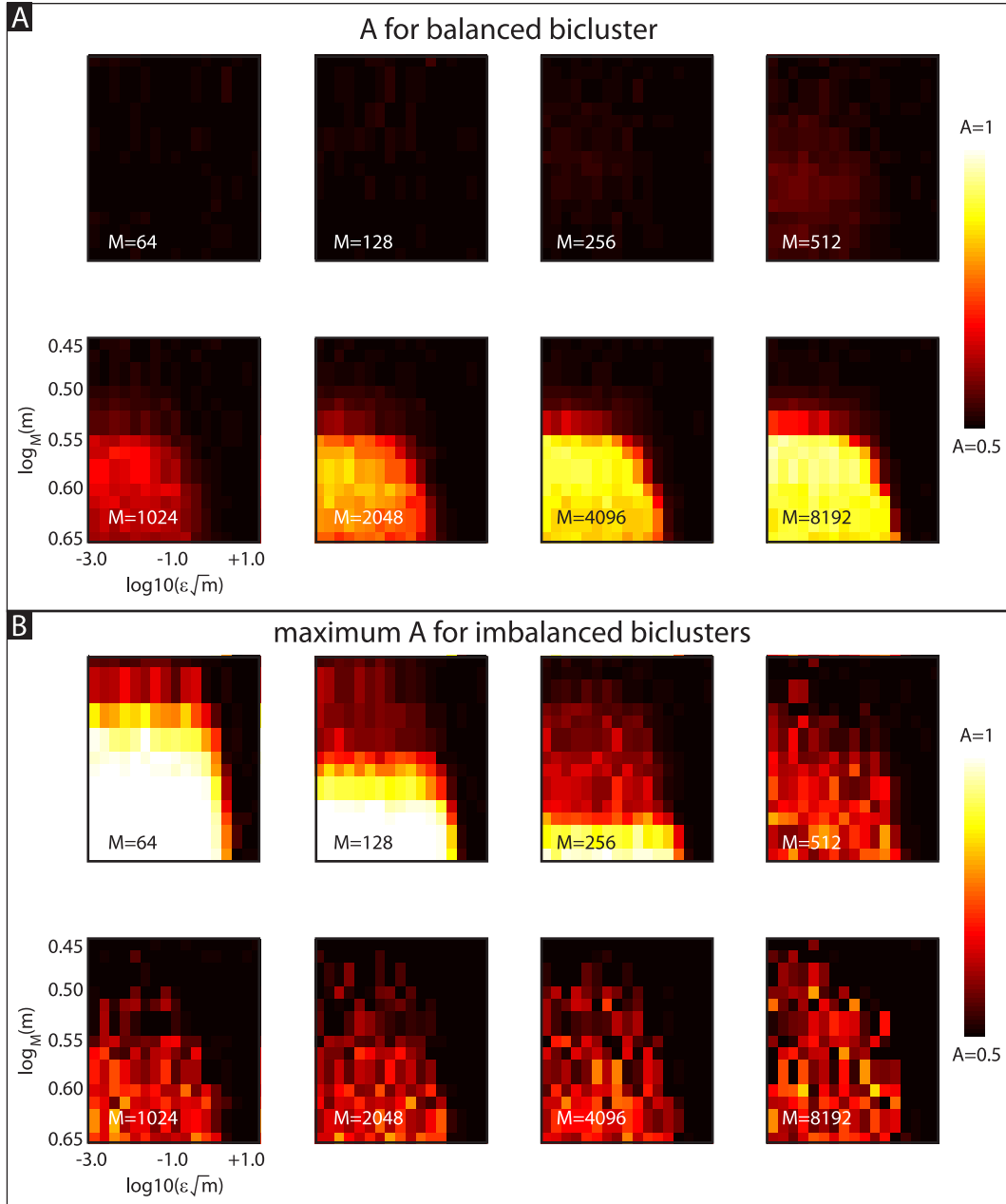


Figure 50: These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving continuous covariates with dimension $N_T = 2$ (see Fig 48B). In panel-A we show the trial-averaged value of A_1 as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. In panel-B we show the trial-averaged value of $\max(A_2, A_3)$ which is usually A_3 . The presence of B_2 and B_3 make it difficult for our bicluster to detect the smaller B_1 when the M is low. Note that, when $M \lesssim 1024$, our algorithm focuses on B_2 or B_3 rather than B_1 .

$$\left[Z_{\text{COL}}^{[t] \text{ 1s}} \right]_k = \widetilde{\sum}_{\substack{j', j \in D, j' \neq j; \\ k \text{ fixed, } k' \neq k}} D_{jk} D_{jk'} D_{j'k'} T_{j't} D_{j'k}, \text{ for each } t \in \{1, \dots, N_T\}.$$

This 1-sided correction only counts the continuous-covariates associated with the j' row-index, ignoring the contribution from $T_{j,t}$. The associated row- and column-scores are:

$$\begin{aligned} \left[Z_{\text{ROW}}^{[T] \text{ 1s}} \right]_j^2 &= \frac{1}{\kappa^2} \widetilde{\sum}_t \left[Z_{\text{ROW}}^{[t] \text{ 1s}} \right]_j^2, & \left[Z_{\text{COL}}^{[T] \text{ 1s}} \right]_k^2 &= \frac{1}{\kappa^2} \widetilde{\sum}_t \left[Z_{\text{COL}}^{[t] \text{ 1s}} \right]_k^2, \\ \left[Z_{\text{ROW}} \right]_j &= \left[Z_{\text{ROW}}^{\text{base}} \right]_j - \left[Z_{\text{ROW}}^{[T] \text{ 1s}} \right]_j, & \left[Z_{\text{COL}} \right]_k &= \left[Z_{\text{COL}}^{\text{base}} \right]_k - \left[Z_{\text{COL}}^{[T] \text{ 1s}} \right]_k. \end{aligned}$$

The 1-sided correction is essentially the same as the 2-sided correction with regards to the row-subscores. However, the 1-sided correction is rather different for the column-subscores. Under the 1-sided correction the column-subscores only capture the norm (rather than the norm-squared) of the averaged continuous-covariate vector accumulated across biclusters that include column- k . This difference won't affect the D -only situation very much, but will have ramifications when we include the X -matrix of controls in section 12.

11 Correcting for Sparsity

Up to this point we've been assuming that the entries of the original data-matrix take on a range of values, both positive and negative. Our notion of low-rank was predicated on this assumption: Indeed, we've spent our time analyzing how our loop-scores function when the underlying data-matrix was drawn from an isotropic gaussian distribution, assuming that any planted-biclusters were drawn from an eccentric (nonisotropic) gaussian distribution. Because binarization is built into our algorithm, our analysis immediately carries forward to the situation where the original data is binary rather than continuous, provided that we assume the underlying data-matrix was drawn from a binarized version of an isotropic gaussian, and that any planted-biclusters were drawn from binarized versions of eccentric gaussians.

All of this analysis depends on one thing: our algorithm assumes that the fraction of negative entries within the data-matrix is comparable to the fraction of positive entries. If the original data-matrix has a surplus of, say, negative entries, then the binarized version of the data-matrix will be 'sparse' (i.e., the fraction of positive entries will be much less than 0.5, and the fraction of negative entries will be much greater than 0.5). Such a sparse matrix is likely to contain large submatrices consisting of mostly negative entries, simply due to their abundance. As we originally described above, our algorithm will typically focus on these large submatrices, even though they are not statistically significant. The reason for this behavior is that the abundance of negative entries gives rise to an abundance of 'all-negative loops' (i.e., rank-1 loops involving only negative entries). The loop-score we describe in section 2 treats all rank-1 loops the same way – each adds one to the total score – regardless of whether or not those loops consist of positive or negative entries (or a mixture of the two). Therefore, the large number of all-negative loops drowns out the other (more significant) rank-1 loops, obscuring the presence of any significant biclusters.

To rectify this behavior we can correct for sparsity. The main idea is to rescore the loops so that rank-1 loops consisting of entries that are otherwise abundant do not add much to the score, whereas rank-1 loops containing entries that are otherwise rare add more to the score. To describe this correction let's assume that our $M \times N$ data-matrix D is binary, with sparsity p (i.e., a fraction p of entries equal to +1, and a fraction $q = 1 - p$ of the entries equal to -1). We set $\alpha = p - q$ to be the column mean of each column of D , and $1/\delta = 4pq$ to be the inverse of the variance of each column. Using this notation we define:

$$\begin{aligned} \left[Z_{\text{ROW}} \right]_j &= \widetilde{\sum}_{\substack{j \text{ fixed, } j' \neq j; \\ k' \neq k}} \left[D_{jk} - \alpha \right] \cdot \delta \cdot \left[D_{j'k} - \alpha \right] \left[D_{j'k'} - \alpha \right] \cdot \delta \cdot \left[D_{jk'} - \alpha \right], \\ \left[Z_{\text{COL}} \right]_k &= \widetilde{\sum}_{\substack{j' \neq j; \\ k \text{ fixed, } k' \neq k}} \left[D_{jk} - \alpha \right] \left[D_{jk'} - \alpha \right] \cdot \delta \cdot \left[D_{j'k'} - \alpha \right] \left[D_{j'k} - \alpha \right] \cdot \delta. \end{aligned}$$

These scores 'normalize' D by subtracting the mean from each entry and then dividing by the standard-deviation. This normalization ensures that, when considering a large random matrix in the large M, N limit, the mean row- and column-score will remain close to 0 and the variance of the row- and column-scores will remain close to $2/(MN^2)$ and $2/(M^2N)$, respectively (regardless of the fixed sparsity-coefficient p). Any rank-1 loop composed of common entries (e.g., the abundant negative entries in our earlier example) will only add a small amount to the total score, whereas a rank-1 loop composed of rare entries (e.g., positive entries) will add a larger amount to the total score.

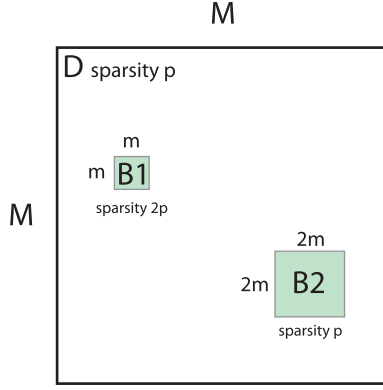


Figure 51: This figure illustrates the setup for the numerical experiments shown in Fig 52. These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving a sparse matrix with sparsity-coefficient $p = 0.0625$. For each numerical experiment we let D be a large $M \times M$ random matrix with entries chosen independently and randomly to be $+1$ with probability $p = 0.0625$, and -1 with probability $q = 1 - p$. Within D we implant an ε -error $m \times m$ bicluster $B1$ with sparsity-coefficient $2p$, and another ε -error $2m \times 2m$ bicluster $B2$ with sparsity-coefficient p . These biclusters are both approximately rank-1, and are generated using the method described in section 5.2, with m , ε and the sparsity-coefficient given as parameters. Both biclusters are implanted randomly. Note that our illustration is slightly misleading; $B1$ and $B2$ will not necessarily be disjoint. Rather, because they are randomly chosen, the row- and column-subsets of $B1$ and $B2$ will typically have a small overlap. For the purposes of this simple numerical experiment the entries in this overlap are randomly assigned to either $B1$ or $B2$. For each numerical experiment we calculate the average auc A_1 for the bicluster $B1$, as well as the average auc A_2 for bicluster $B2$. Our goal is to find both $B1$ and $B2$, even in the presence of the abundant negative entries within D . Because of the tradeoff between size, noise and sparsity, there will be combinations of m , ε and p for which $B2$ is prioritized over $B1$, and vice-versa.

To be more specific, of the 16 different possible loops within a binary-matrix, 8 will be rank-1 and 8 will be rank-2 (recall Fig 20). The 8 rank-1 loops fall into three categories: (a) four negative entries, (b) two negative and two positive entries, and (c) four positive entries. These categories contribute, respectively, p^2/q^2 , 1, and q^2/p^2 to the summands above. The 8 rank-2 loops fall into two categories: (d) three negative entries and one positive entry, and (e) three positive entries and one negative entry. These categories contribute, respectively, $-p/q$ and $-q/p$ to the summands above.

To relate this to the planted-bicluster problem, let's consider an $M \times N$ data-matrix D with sparsity-coefficient $p < 0.5$. Let's implant within D a rank-1 error- ε $m \times n$ bicluster B with a potentially different sparsity-coefficient \tilde{p} with $p < \tilde{p} < 0.5$. We'll generate B using the algorithm described in section 5.2. If $\varepsilon\sqrt{m}$ is not too large, the bicluster B will produce a row-signal of approximately $(m/M)(n/N)^2\eta$ and a column-signal of approximately $(m/M)^2(n/N)\eta$, where η is given by:

$$\eta \sim g \left\{ \xi_{4,0} \frac{p^2}{q^2} + \xi_{2,2} + \xi_{0,4} \frac{q^2}{p^2} \right\} - (1-g) \left\{ \xi_{4,0} \frac{p}{q} + \xi_{2,2} \frac{1}{2} \left(\frac{p}{q} + \frac{q}{p} \right) + \xi_{0,4} \frac{q}{p} \right\}, \text{ with}$$

$$\xi_{4,0} = (\hat{p}^4 + \hat{q}^4), \quad \xi_{2,2} = 4(\hat{p}^3\hat{q} + \hat{p}^2\hat{q}^2 + \hat{p}\hat{q}^3), \quad \text{and} \quad \xi_{0,4} = 2\hat{p}^2\hat{q}^2,$$

and \hat{p} and $\hat{q} = 1 - \hat{p}$ given by the construction in section 5.2 (i.e., using sparsity-coefficient \tilde{p}). The terms $\xi_{4,0}$, $\xi_{2,2}$ and $\xi_{0,4}$ represent the distribution of loops from categories (a), (b) and (c), respectively, from within the outer product uv^T used in the construction of B . In the limit $\varepsilon\sqrt{m} \rightarrow 0$ and $p \ll 0.5$, the signal-factor η can be approximated by:

$$\eta \sim \left\{ \xi_{4,0} \frac{p^2}{q^2} + \xi_{2,2} + \xi_{0,4} \frac{q^2}{p^2} \right\}, \text{ with } \xi_{4,0} \sim 1 - 2\tilde{p} + \frac{1}{2}\tilde{p}^2, \quad \xi_{2,2} \sim 2\tilde{p} - \tilde{p}^2, \quad \text{and} \quad \xi_{0,4} \sim \frac{1}{2}\tilde{p}^2, \text{ implying}$$

$$\eta \sim \left(1 - 2\tilde{p} + \frac{1}{2}\tilde{p}^2 \right) \frac{p^2}{q^2} + 2\tilde{p} - \tilde{p}^2 + \frac{1}{2}\tilde{p}^2 \frac{q^2}{p^2} \sim \frac{1}{2} \left(\frac{\tilde{p}}{p} \right)^2.$$

The overall signal produced by B depends on the size of B and the noise (as before), but also on the sparsity coefficient \tilde{p} of B , which influences \hat{p} and η . The signal-factor η typically grows with \tilde{p} and, ultimately, there is a tradeoff: a bicluster $B1$ which is small but has a higher \tilde{p} may produce a signal that is just as great as another bicluster $B2$ that is large but has a lower \tilde{p} . For example, the bicluster $B1$ may have more rank-1 loops containing rare positive-entries; their presence causes the loops of $B1$ to count for more than the typical loops of $B2$. An example along these lines is provided in Figs 51 and 52.

Sparsity arises in practice when attempting to analyze genotyped data; single-nucleotide-polymorphisms (SNPs) can involve minor-allele-frequencies (MAFs) that are quite small (e.g., 0.1 or smaller), giving rise to sparsity-coefficients that

Performance of biclustering algorithm with sparsity

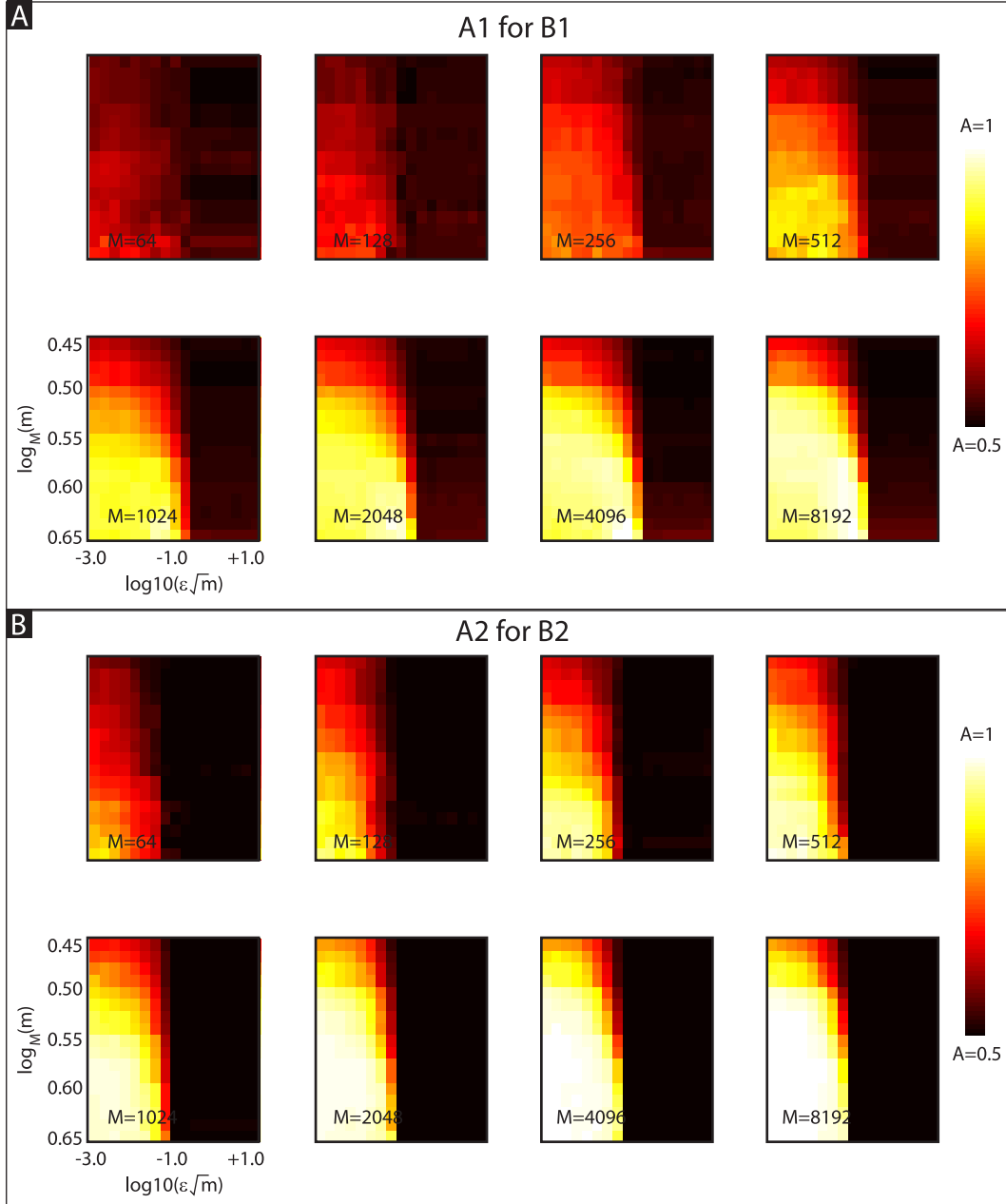


Figure 52: These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving a sparse matrix (see Fig 51). In panel-A we show the trial-averaged value of A_1 as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. In panel-B we show the trial-averaged value of A_2 . Note that our algorithm typically finds both B_1 and B_2 , despite the abundance of negative entries in D . Depending on the parameters, either B_1 or B_2 can be prioritized.

are very far from 0.5 (e.g., ≤ 0.01 or ≥ 0.99). Moreover, the different columns k of the data-matrix often correspond to different SNPs with different MAFs and hence different sparsity-coefficients p_k . We can generalize our sparsity-correction to account for this scenario by defining:

$$\begin{aligned}
[Z_{\text{ROW}}]_j &= \sum_{\substack{j \text{ fixed, } j' \neq j; \\ k' \neq k}} [D - 1\alpha^\top]_{jk} \Delta_{kk} [D^\top - \alpha 1^\top]_{kj'} [D - 1\alpha^\top]_{j'k'} \Delta_{k'k'} [D^\top - \alpha 1^\top]_{k'j}, \\
[Z_{\text{COL}}]_k &= \sum_{\substack{j' \neq j; \\ k \text{ fixed, } k' \neq k}} [D^\top - \alpha 1^\top]_{kj} [D - 1\alpha^\top]_{jk'} \Delta_{k'k'} [D^\top - \alpha 1^\top]_{k'j'} [D - 1\alpha^\top]_{j'k} \Delta_{kk},
\end{aligned}$$

where $\vec{\alpha} \in \mathbb{R}^N$ is the $N \times 1$ vector of means $\alpha_k = p_k - q_k$, $\Delta \in \mathbb{R}^{N \times N}$ is the diagonal matrix with entries $\Delta_{kk}^{-1} = 4p_k q_k$, and $\vec{1} \in \mathbb{R}^M$ is the $M \times 1$ vector of all ones. Both these row- and column-scores can be computed using two binary matrix-matrix multiplications as well as $O(1)$ matrix-vector multiplications and $O(\max(M, N))$ vector-vector dot-products (see section 12 for details).

12 Putting it all together

So far we've discussed modifications to our algorithm that account for cases-vs-controls, categorical- and continuous-covariates, as well as sparsity. One of our goals was to ensure that these modifications were not mutually exclusive; they can all be combined into one algorithm which accounts for all of the above. When combining these modifications we end up calculating a variety of sub-scores which contribute in various ways to our final row- and column-scores. Within the context of our methodology, we have chosen to perform the following steps in order:

1. Correct for sparsity.
2. Correct for continuous covariates.
3. Given I_{req} , correct for categorical covariates.
4. Correct for cases versus controls.

Based on our experience, we believe that the sparsity correction should be performed first, because the sparsity correction alters the weight of each loop. After that, the correction for continuous-covariates should come before the correction for categorical-covariates; this ensures that each category refers to its own distribution of continuous-covariates. Finally, the correction for categorical-covariates should come before the correction for cases-vs-controls; this ensures that biclusters which are balanced across case-categories are not unfairly penalized by an imbalance across control-categories. Two examples which have motivated our choices are given in Fig 53.

The main user-specified input-parameter is I_{req} , which is used in Step-3 to dictate the number of covariate-categories a bicluster must extend across in order to be considered by the algorithm. Technically speaking, Step-2 also involves a parameter κ^2 , which dictates how broadly a bicluster must be distributed in continuous-covariate-space in order to be considered. While κ^2 could in principle be modified by the user, we typically set κ^2 so that, on average, biclusters produce no signal when they are drawn from only half of covariate-space (see Fig 47).

Before we describe our general procedure, we briefly review our notation. We'll consider a situation where the patients are segregated into I_{cat} different covariate-categories (e.g., study, batch-number, recording location, etc). We'll use the post-script i to track the category-number. We'll denote by M_{D_i} and M_{X_i} the number of case- and control-patients in category- i . Each patient will also be associated with N different genetic-measurements. Each of these genetic-measurements will, in general, give rise to its own sparsity-coefficient p_k (for $k = 1, \dots, N$) calculated across the entire patient population. The genetic-data for the case- and control-patients in category- i is contained in the data-matrices D_i and X_i , of size $M_{D_i} \times N$ and $M_{X_i} \times N$, respectively. The matrix-entry $D_i(j, k)$ contains the k^{th} -genetic-measurement for the j^{th} -case-patient from category- i . The matrix-entry $X_i(j, k)$ contains the same for the j^{th} -control-patient. In addition to the genetic-data, each patient will also be associated with an N_T -dimensional vector of continuous-covariates. These continuous-covariates are contained in covariate-matrices T_{D_i} and T_{X_i} , of size $M_{D_i} \times N_T$ and $M_{X_i} \times N_T$, respectively. The row-vector $T_{D_i}(j, t)$ for $t = 1, \dots, N_T$ contains the continuous-covariates for the j^{th} case-patient in category- i . The row-vector $T_{X_i}(j, t)$ for $t = 1, \dots, N_T$ contains the same for the j^{th} -control-patient. In the expressions below we'll use the dummy-variable 'Q' below to denote either D or X ; e.g., M_{Q_i} could be either M_{D_i} or M_{X_i} . We'll also use the notation 1_{Q_i} to refer to the $M_{Q_i} \times 1$ vector of all ones.

After binarizing all the matrices involved we calculate the sparsity coefficients:

$$p_k = \sum_{\substack{i, j; \\ Q=\{D, X\}}} [Q_i(j, k)], \text{ where } [x] = \max(0, x).$$

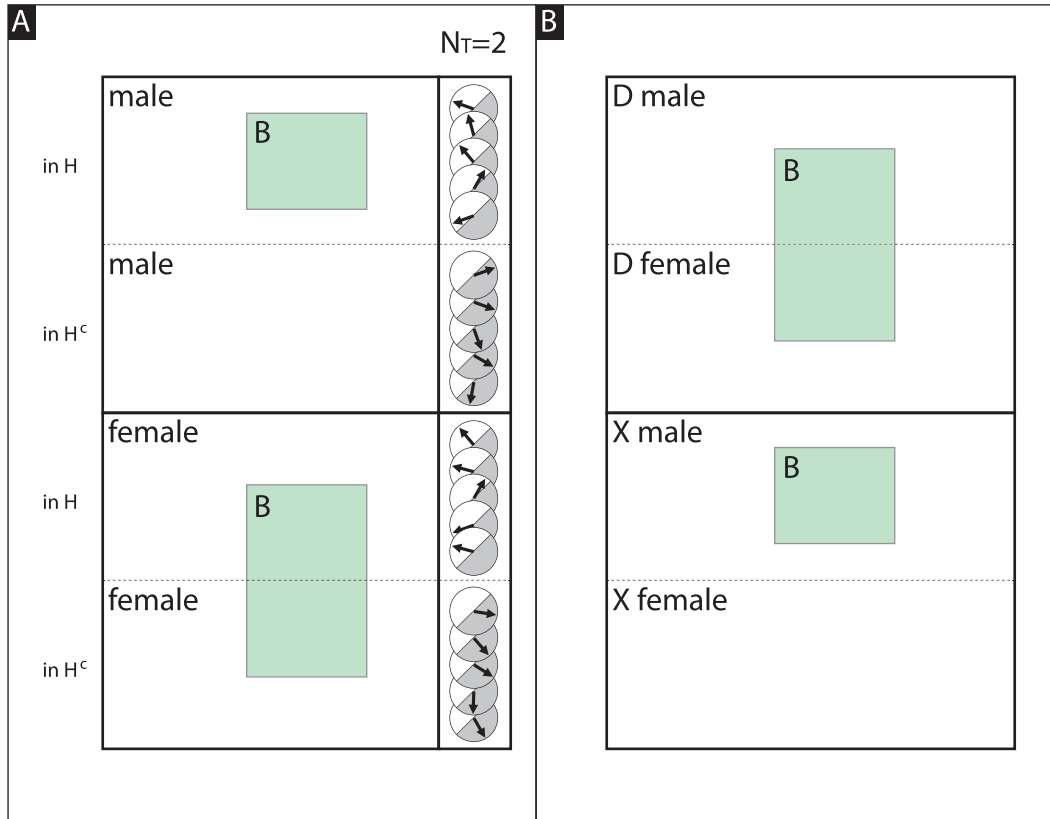


Figure 53: Examples illustrating the order of corrections for our full algorithm. In Panel-A we show a situation with both continuous- and categorical-covariates. The continuous-covariates are drawn from an $N_T = 2$ dimensional space (right side). There are two covariate-categories, male and female. The bicluster B shown extends across many males as well as many females. While the females in B have continuous-covariates drawn isotropically from \mathbb{R}^{N_T} , the males in B all have continuous-covariates drawn from the same half-space $H \in \mathbb{R}^{N_T}$ (indicated by the white semicircle). If we were to correct for categorical-covariates before correcting for continuous-covariates, the signal produced by B would be high, which is incorrect in this case. If, instead, we correct for continuous-covariates before correcting for categorical-covariates, the signal produced by B will be correctly reduced to zero. In Panel-B we show a situation with both categorical-covariates as well as cases-vs-controls. The bicluster B extends across both male-cases and female-cases, but only across male-controls. If we were to correct for cases-vs-controls before correcting for categorical-covariates, the score for B would be reduced to zero, even though B is balanced across the cases in a manner not exhibited across the controls. If, instead, we correct for categorical-covariates before correcting for cases-vs-controls, the signal produced by B will be high, as it should be.

Using these sparsity coefficients we define the probabilities $q_k = 1 - p_k$, the means $\alpha_k = p_k - q_k$, and the variances $\delta_k^{-1} = 4p_k q_k$. These means and variances allow us to define the $N \times 1$ vector $\vec{\alpha}$ and the $N \times N$ diagonal matrix Δ with diagonal-entries δ_k . We then compute the base sub-scores for each combination of $Q = D$ and $Q' \in \{D, X\}$, for each pair of categories i, i' :

$$\begin{aligned} \left[Z_{\text{ROW}}^{Q, Q'; i, i'; \text{base}} \right]_j &= \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ fixed in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i' \\ k' \neq k}} [Qi - 1_{Qi} \alpha^\top]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k} \\ &\quad \times [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k'} \Delta_{k'k'} [Qi - 1_{Qi} \alpha^\top]_{j,k'}, \\ \left[Z_{\text{COL}}^{Q, Q'; i, i'; \text{base}} \right]_k &= \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i' \\ k \text{ fixed}, k' \neq k}} [Qi - 1_{Qi} \alpha^\top]_{j,k} [Qi - 1_{Qi} \alpha^\top]_{j,k'} \Delta_{k'k'} \\ &\quad \times [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k'} [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k} \Delta_{kk}, \end{aligned}$$

as well as the covariate-dependent sub-scores for each $t = 1, \dots, N_T$:

$$\begin{aligned} \left[Z_{\text{ROW}}^{Q, Q'; i, i'; [t]} \right]_j &= \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ fixed in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i' \\ k' \neq k}} [Qi - 1_{Qi} \alpha^\top]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k} [T_{Q'i'}]_{j',t} \\ &\quad \times [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k'} \Delta_{k'k'} [Qi - 1_{Qi} \alpha^\top]_{j,k} [T_{Qi}]_{j,t}, \\ \left[Z_{\text{COL}}^{Q, Q'; i, i'; [t]} \right]_k &= \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i' \\ k \text{ fixed}, k' \neq k}} [Qi - 1_{Qi} \alpha^\top]_{j,k} [T_{Qi}]_{j,t} [Qi - 1_{Qi} \alpha^\top]_{j,k'} \Delta_{k'k'} \\ &\quad \times [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k'} [T_{Q'i'}]_{j',t} [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k} \Delta_{kk}. \end{aligned}$$

Combining these sub-scores allows us to correct for continuous-covariates within each combination of Q, Q', i, i' :

$$\left[Z_{\text{ROW}}^{Q, Q'; i, i'; [T]} \right]_j = \frac{1}{\kappa^2} \widetilde{\sum}_t \left[Z_{\text{ROW}}^{Q, Q'; i, i'; [t]} \right]_j^2, \text{ and } \left[Z_{\text{COL}}^{Q, Q'; i, i'; [T]} \right]_k = \frac{1}{\kappa^2} \widetilde{\sum}_t \left[Z_{\text{COL}}^{Q, Q'; i, i'; [t]} \right]_k,$$

where $\kappa^2 = \kappa^2(N_T)$ as defined in section 10. We then define sub-scores for each Q, Q', i, i' .

$$\left[Z_{\text{ROW}}^{Q, Q'; i, i'} \right]_j^2 = \left[Z_{\text{ROW}}^{Q, Q'; i, i'; \text{base}} \right]_j^2 - \left[Z_{\text{ROW}}^{Q, Q'; i, i'; [T]} \right]_j^2, \text{ and } \left[Z_{\text{COL}}^{Q, Q'; i, i'} \right]_k = \left[Z_{\text{COL}}^{Q, Q'; i, i'; \text{base}} \right]_k - \left[Z_{\text{COL}}^{Q, Q'; i, i'; [T]} \right]_k.$$

We correct for categorical covariates by choosing the I_{req} -largest element across i' for each row-sub-score, and the I_{req}^2 -largest element across i, i' for each column-sub-score (see section 9):

$$\left[Z_{\text{ROW}}^{Q, Q'; i} \right]_j = \left[Z_{\text{ROW}}^{Q, Q'; i, \#I_{\text{req}}} \right]_j, \text{ and } \left[Z_{\text{COL}}^{Q, Q'} \right]_k = \left[Z_{\text{COL}}^{Q, Q'; \#I_{\text{req}}^2} \right]_k.$$

Finally, we correct for cases-vs-controls by subtracting the control-sub-scores from the case-sub-scores (see section 8):

$$\left[Z_{\text{ROW}} \right]_j = \left[Z_{\text{ROW}}^{D, D; i} \right]_j - \left[Z_{\text{ROW}}^{D, X; i} \right]_j \text{ for } j \in Di, \text{ and } \left[Z_{\text{COL}} \right]_k = \left[Z_{\text{COL}}^{D, D} \right]_k - \left[Z_{\text{COL}}^{D, X} \right]_k.$$

An example illustrating this combination of corrections is shown in Figs 54 and 55. This example illustrates the performance of our algorithm on a planted-bicluster problem involving both an $M \times N$ case-matrix D as well as a $2M \times N$ control-matrix X , where $N = 3M$. Within D and X , the case- and control-patients (i.e., rows) are randomly assigned to one of $I_{\text{cat}} = 3$ categories, with $M_{Di} = M/3$ and $M_{Xi} = 2M/3$ respectively. In addition, we endow each patient with a randomly generated continuous-covariate vector of dimension $N_T = 2$. The genes (i.e., columns) have varying degrees of sparsity; one-third of the columns have a sparsity-coefficient $p = 0.125$, one-third have $p = 0.25$, and one-third have $p = 0.5$.

Within these case- and control-matrices we implant several rank-1 ε -error biclusters, each with sparsity $p = 0.25$. The size of these implanted biclusters will depend on m and n , where n is related to m by ensuring that $\log_N(n) = \log_M(m)$ (i.e., if $m = \sqrt{M}$, then $n = \sqrt{N}$).

Case-specific and balanced: The first bicluster we implant is an $m \times n$ bicluster B_0 restricted to the cases. The bicluster B_0 is implanted within D by choosing m rows and columns at random; consequently, B_0 is a case-specific bicluster that is typically balanced across all 3 covariate-categories and continuous-covariates.

Case-specific and somewhat balanced: The next three biclusters we implant, B_1 B_2 and B_3 , are each of size $3m/2 \times 3n/2$ and are restricted to the cases. Each of these bicluster has its rows and columns chosen at random from D , with the restriction that B_1 excludes category-1, B_2 excludes category-2, and B_3 excludes category-3; each of these three case-specific biclusters is balanced with respect to the continuous-covariates, but only extends across 2 of the 3 covariate-categories.

Case-nonspecific: The next bicluster we implant, B_4 , is of size $9m/2 \times 3n/2$, and extends across both the cases and controls. This bicluster has $1/3$ of its rows drawn randomly from D and $2/3$ drawn from X ; B_4 will be balanced with respect to the continuous- and categorical-covariates, but will not be case-specific (extending into the controls).

Case-specific but imbalanced: The final two biclusters we implant, B_5 and B_6 , are each of size $3m/2 \times 3n/2$, and are restricted to the cases. Each of these biclusters has its rows and columns chosen at random from D , with the restriction that B_5 be restricted to rows that have continuous-covariates drawn from a randomly oriented half-space $H \in \mathbb{R}^{N^r}$, and B_6 be restricted to rows that have continuous-covariates drawn from the complement H^c . Consequently, each of these two case-specific biclusters will be balanced with respect to the categorical-covariates, but not the continuous-covariates.

All of our biclusters have columns drawn randomly; each will typically overlap all of the different sparsity-domains in D and/or X . Note that the illustration in Fig 54 is misleading; the biclusters will not necessarily have disjoint columns (nor strongly overlapping rows). Rather, because they are randomly chosen, the row- and column-subsets of each bicluster will typically have a small overlap. For the purposes of this numerical experiment we implant each bicluster in the order described above: any overlapping entries will be overwritten by the next bicluster in the list (with the imbalanced biclusters taking precedence).

For each numerical experiment we calculate the average auc A_0 for the bicluster B_0 , as well as the average auc A_1 for bicluster B_1 and so forth. In our first numerical experiment shown on the left of Fig 55 we set $I_{\text{req}} = 3$; We are looking for biclusters that span *all* of the covariate-categories, and we are not interested in finding B_1 B_2 or B_3 . With this choice of $I_{\text{req}} = 3$, our goal is to find B_0 despite the masking effects of the other biclusters. In our second numerical experiment shown on the right of Fig 55 we set $I_{\text{req}} = 2$; We are looking for biclusters that span at least 2 of the 3 covariate-categories. With $I_{\text{req}} = 2$ we are satisfied with either B_0 B_1 B_2 or B_3 , but we still don't want to find B_4 B_5 or B_6 , since these are either case-nonspecific or imbalanced with respect to the continuous-covariates.

2-sided versus 1-Sided continuous-covariate correction: As we alluded to earlier, it is possible to correct for continuous-covariates in step-2 by using a '1-sided'-correction, rather than the 2-sided correction described above. These 1-sided corrections involve altering the covariate-dependent sub-scores:

$$\begin{aligned} \left[Z_{\text{ROW}}^{Q,Q'; i,i'; [t] \text{ 1s}} \right]_j &= \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed;} \\ j \text{ fixed in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i' \\ k' \neq k}} [Qi - 1_{Qi}\alpha^\top]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'}\alpha^\top]_{j',k} [T_{Q'i'}]_{j',t} \\ &\quad \times [Q'i' - 1_{Q'i'}\alpha^\top]_{j',k'} \Delta_{k'k'} [Qi - 1_{Qi}\alpha^\top]_{j,k'}, \\ \left[Z_{\text{COL}}^{Q,Q'; i,i'; [t] \text{ 1s}} \right]_k &= \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed;} \\ j \text{ in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i' \\ k \text{ fixed, } k' \neq k}} [Qi - 1_{Qi}\alpha^\top]_{j,k} [Qi - 1_{Qi}\alpha^\top]_{j,k'} \Delta_{k'k'} \\ &\quad \times [Q'i' - 1_{Q'i'}\alpha^\top]_{j',k'} [T_{Q'i'}]_{j',t} [Q'i' - 1_{Q'i'}\alpha^\top]_{j',k} \Delta_{kk}. \end{aligned}$$

Note that these covariant-dependent subscores depend on $[T_{Q'i'}]_{j',t}$, but not $[T_{Qi}]_{j,t}$. These 1-sided-subscores are then combined:

$$\begin{aligned} \left[Z_{\text{ROW}}^{Q,Q'; i,i'; [T] \text{ 1s}} \right]_j^2 &= \frac{1}{\kappa^2} \widetilde{\sum}_t \left[Z_{\text{ROW}}^{Q,Q'; i,i'; [t] \text{ 1s}} \right]_j^2 \\ \left[Z_{\text{COL}}^{Q,Q'; i,i'; [T] \text{ 1s}} \right]_k^2 &= \frac{1}{\kappa^2} \widetilde{\sum}_t \left[Z_{\text{COL}}^{Q,Q'; i,i'; [t] \text{ 1s}} \right]_k^2, \end{aligned}$$

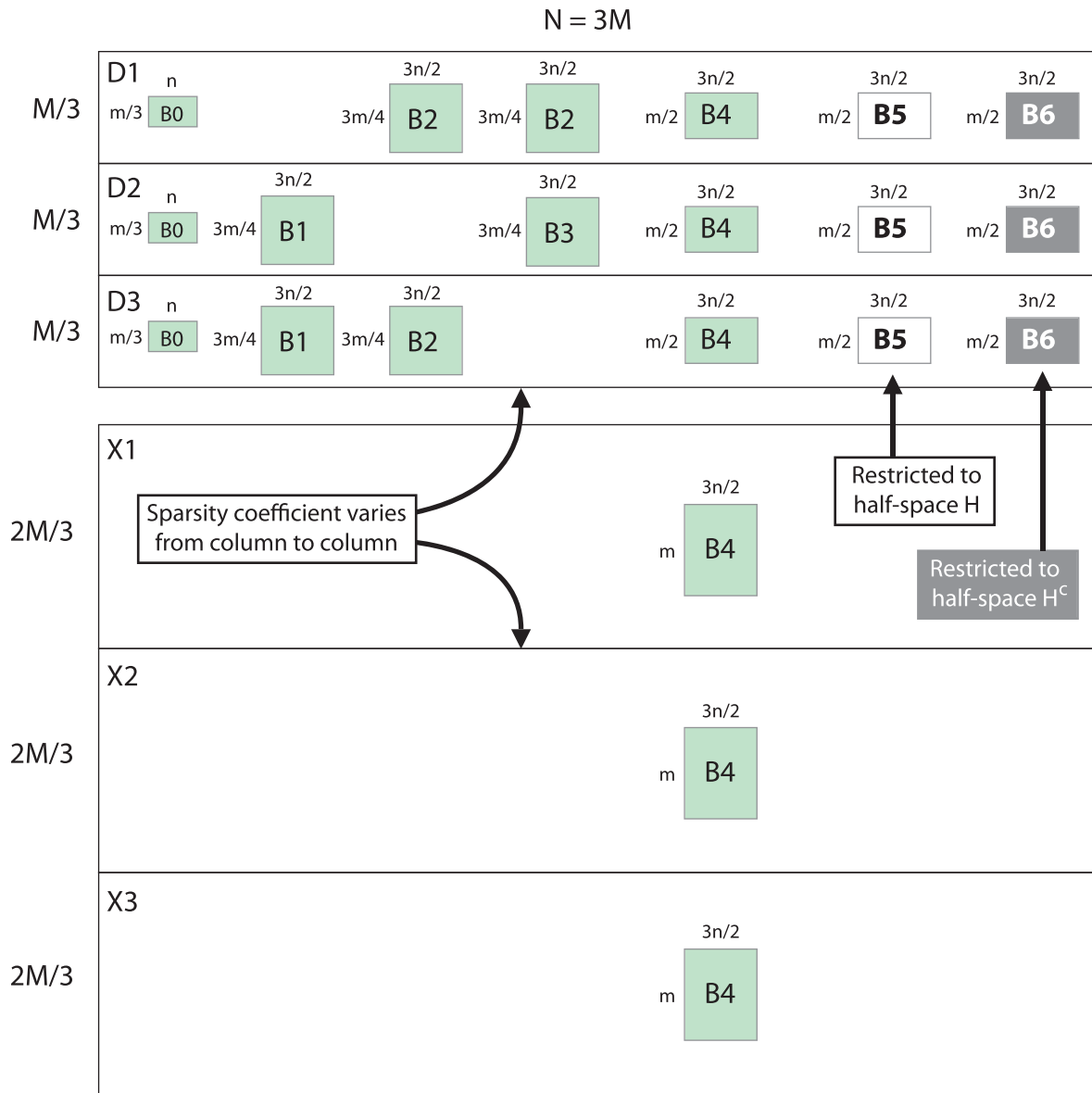


Figure 54: This figure illustrates the setup for the numerical experiments shown in Fig 55. See text for description.

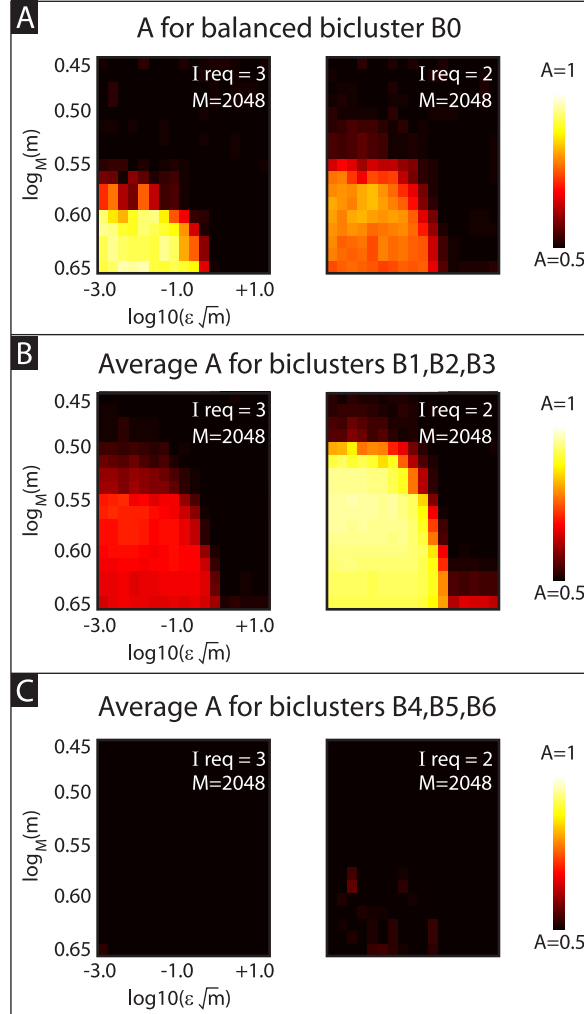


Figure 55: These numerical experiments illustrate the performance of our algorithm on the planted-bicluster problem described in Fig 54. In panel-A we show the trial-averaged value of A_0 as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. In panel-B we show the trial-averaged value of A_1, A_2 and A_3 . In panel-C we show the trial-averaged value of A_4, A_5 and A_6 . The left side of each panel displays results with $I_{\text{req}} = 3$, the right side with $I_{\text{req}} = 2$. Note that, when $I_{\text{req}} = 3$ the goal was to find B_0 , as this was the only case-specific bicluster that was balanced across all the continuous-covariates and covariate-categories. On the other hand, when $I_{\text{req}} = 2$ the goal was to find the most dominant of B_0, B_1, B_2 and B_3 , as each of these case-specific biclusters were balanced across all the continuous-covariates while extending across at least 2 covariate-categories. Note that our algorithm typically achieves these goal, despite the presence of the other biclusters B_4, B_5 and B_6 .

and contribute to the sub-scores for each Q, Q', i, i' as follows:

$$\begin{aligned} \left[Z_{\text{ROW}}^{Q, Q'; i, i'} \right]_j &= \left[Z_{\text{ROW}}^{Q, Q'; i, i'; \text{base}} \right]_j - \left[Z_{\text{ROW}}^{Q, Q'; i, i'; [T] \text{ 1s}} \right]_j \\ \left[Z_{\text{COL}}^{Q, Q'; i, i'} \right]_k &= \left[Z_{\text{COL}}^{Q, Q'; i, i'; \text{base}} \right]_k - \left[Z_{\text{COL}}^{Q, Q'; i, i'; [T] \text{ 1s}} \right]_k. \end{aligned}$$

As mentioned earlier in section 10, these 1-sided corrections don't change the row-scores much, but do alter the column-scores. Under the 1-sided correction the column-subscores $Z_{\text{COL}}^{Q, Q'; i, i'; [T] \text{ 1s}}$ now capture the norm (rather than the norm-squared) of the averaged continuous-covariate-vector accumulated across any biclusters intersecting column- k .

With regards to case-specific biclusters, both the 2-sided and 1-sided continuous-covariate corrections accomplish the same general goal. That is to say, both correction schemes reduce the score of case-specific biclusters that are imbalanced with respect to the space of continuous-covariates, while leaving unaffected the scores of case-specific biclusters that are balanced in covariate-space (at least to first order). However, these two correction-schemes differ in how they treat certain kinds of biclusters that straddle both cases and controls. The 2-sided correction penalizes biclusters that include a comparable number of cases and controls, regardless of how those cases and controls are distributed in covariate-space. The 1-sided correction also penalizes biclusters that include a comparable number of cases and controls, but behaves differently in one important situation: The 1-sided correction does not penalize biclusters that include both cases and controls as long as (i) the cases are balanced in covariate-space, and (ii) the controls are imbalanced in covariate-space. An illustration of this difference is given in Fig 56.

These two different strategies for continuous-covariate correction are useful in different applications. For example, the 1-sided correction is useful if we want to discover a covariate-independent case-specific signature from amongst a heterogenous group of controls. This kind of correction may be appropriate if we expect many different regions in covariate-space to each give rise to their own biclusters (which we should then need to correct for). On the other hand, the 2-sided correction is more conservative, and is more useful for highlighting case-specific biclusters that are robust to changes in the control population. This kind of correction may be appropriate if we want to isolate biclusters that remain significant when compared with arbitrary groups of controls (rather than merely covariate-balanced groups of controls); this kind of goal can be relevant for developing robust risk-prediction models.

Role of binarization: In the prescription above we choose to binarize T . This binarization allows for a more efficient calculation (see next section), but somewhat compromises the accuracy of the covariate-correction. More specifically, after binarization, the covariate-dependent subscores $\left[Z_{\text{ROW}}^{Q, Q'; i, i'; [T]} \right]_j$ (resp. $\left[Z_{\text{COL}}^{Q, Q'; i, i'; [T]} \right]_k$) only approximate the magnitude (resp. magnitude-squared) of the averaged-covariate-vector associated with any biclusters including row- j (resp. column- k). Once we binarize T , then this averaged-covariate-vector will depend not only on whether or not the relevant biclusters are imbalanced with respect to any half-space H , but also on the orientation of that half-space. For example, assuming the existence of a bicluster B that is fully imbalanced with respect to H , it is possible for H to be oriented in such a way that the covariate-correction does not fully demote the scores associated with B (e.g., the normal vector defining H can lie along a coordinate axis in covariate-space).

To avoid this scenario, and to ensure that all imbalanced biclusters are appropriately penalized, we could have included multiple versions of T , each randomly rotated (in covariate-space) before binarization. With this redundancy, we could calculate the covariate-dependent sub-scores (referred to in the previous paragraph) for each version of T , taking e.g., the highest magnitude across the versions. One issue with this approach is that there is a trade-off between speed and accuracy; the more versions of T we include, the more accurate an estimate we can produce, but the longer it takes to produce this estimate. At some point it becomes more efficient to simply skip the T -binarization step altogether, and just include the full covariate-vector for T .

For our GWAS example at the beginning (see section 1.4), as well as for our numerical-examples throughout, we've seen that these more sophisticated correction techniques are unnecessary; the simplest covariate-correction has sufficed. That is to say, we've binarized T , and only used a single version (with no additional random rotations).

13 Notes regarding computation

As we touched on earlier, the binarization step in our algorithm sets the stage for the efficient storage and manipulation of matrices and vectors. Many of the laborious computations involved in calculating the row- and column-scores can be accomplished without the use of floating-point operations. As an example, consider the row-sub-score $\left[Z_{\text{ROW}}^{Q, Q'; i, i'; [t]} \right]_j$, with $i = i'$, and $Q = Q' = D$. In the following equations we'll refer to the $M_{D_i} \times 1$ vector 1_{D_i} as simply 1 , and suppress the post-scripts i and i' to ease the notation. We'll use matlab conventions for referring to indices: $D_{j,:}$ and $D_{:,k}$ refer to the j^{th} -row and k^{th} -column of D , respectively. We'll also denote the entry-wise product as \cdot , and the entry-wise power as \wedge . I.e., $[Q \cdot Q']_{ij} = Q_{ij}Q'_{ij}$, and $[Q \wedge 2]_{ij} = Q_{ij}^2$.

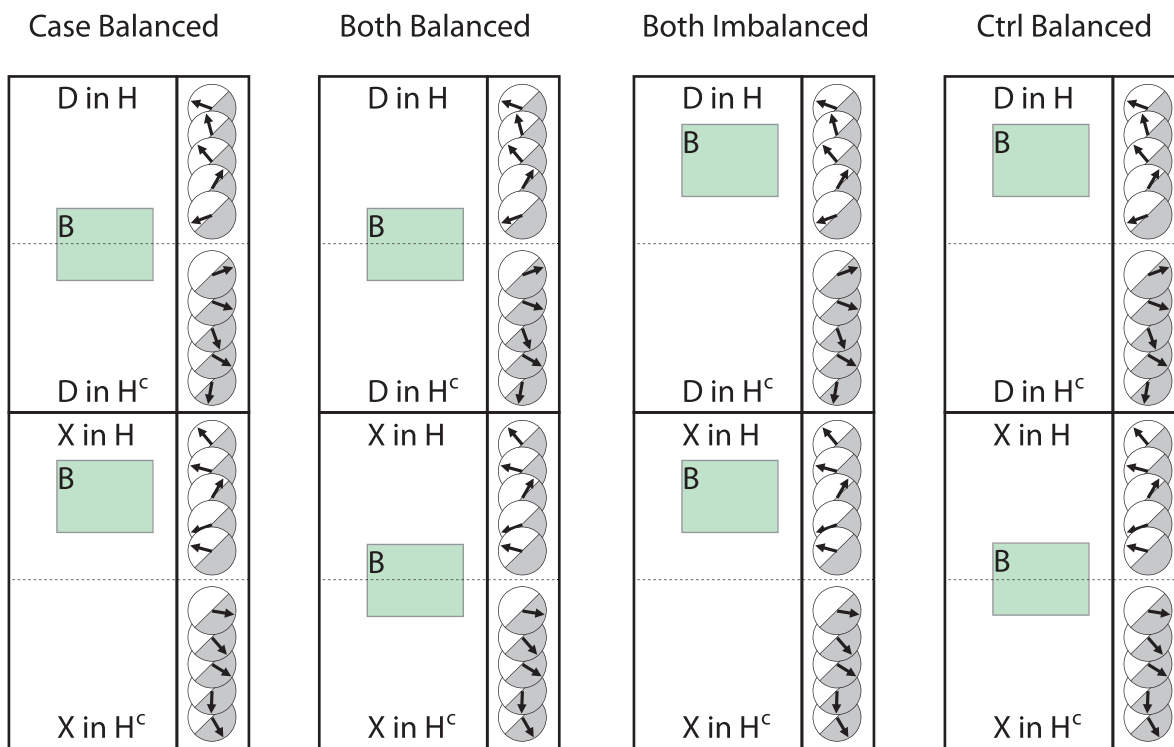


Figure 56: This figure illustrates some of the differences between the 2-sided and 1-sided continuous-covariate correction. In each situation we illustrate a data-matrix containing cases D and controls X . We assume that this data-matrix contains a bicluster B which includes both a case-specific and control-specific component of comparable size. We'll focus on the differences that result when these components of B occupy different regions in covariate-space. For ease of illustration, we've divided covariate-space into a half-space H (white), and its complement H^c (grey). Each of the four panels illustrates a different arrangement; the case- and control-specific components of B are either balanced or imbalanced with regards to the continuous-covariate. Panel 1 (Case Balanced): When the case-component is balanced but the control-component is imbalanced we see the greatest difference between the 2-sided and 1-sided scheme; the former fully demotes the column-score of B (reducing it to 0), whereas the latter does not penalize B at all. Panel 2 (Both Balanced): When both components of B are balanced, both the 2-sided and the 1-sided schemes fully demote the column-score (reducing it to 0). Panel 3 (Both Imbalanced): When both components of B are imbalanced, both the 2-sided and the 1-sided schemes fully demote the column-score (reducing it to 0). Panel 4 (Control Imbalanced): When the control-component is balanced but the case-component is imbalanced, both the 2-sided and the 1-sided schemes greatly lower the column-score, reducing it to a negative value.

The sub-score $\left[Z_{\text{ROW}}^{D,D; i, i'; [t]} \right]_j$ can be calculated by first computing the $M_D \times M_D$ correlation-matrix ‘ C ’, defined via:

$$C = (D - 1\alpha^\top) \Delta (D^\top - \alpha 1^\top) = D\Delta D^\top - 1\alpha^\top \Delta D^\top - D\Delta \alpha 1^\top + (\alpha^\top \Delta \alpha) 11^\top,$$

then computing

$$\begin{aligned} \left[Z_{\text{ROW}}^{\text{full sum}; [t]} \right]_j &= C_{j,:} \cdot \text{diag}(T_{:,t}) \cdot C_{:,j} = C_{j,:}^2 \cdot T_{:,t}, \\ \left[Z_{\text{ROW}}^{j'=j; [t]} \right]_j &= T_{j,t} \cdot C_{jj}^2, \end{aligned}$$

$$\begin{aligned} \left[Z_{\text{ROW}}^{k'=k; [t]} \right]_j &= (D - 1\alpha^\top)_{j,:}^2 \Delta^2 (D^\top - \alpha 1^\top)^2 T_{:,t} \\ &= (11^\top - 2[D \cdot * 1\alpha^\top] + 1\alpha^2)_{j,:} \Delta^2 (11^\top - 2[D^\top \cdot * \alpha 1^\top] + \alpha^2 1^\top) T_{:,t}, \end{aligned}$$

$$\left[Z_{\text{ROW}}^{j'=j \& k'=k; [t]} \right]_j = T_{j,:} \cdot [1_j 1^\top \Delta^2 1; -4 \cdot D_{j,:} \text{diag}(\alpha) \Delta^2 1; +6 \cdot 1_j \alpha^2 \Delta^2 1; -4 \cdot D_{j,:} \text{diag}(\alpha^3) \Delta^2 1; +1_j \alpha^4 \Delta^2 1],$$

and forming the combination:

$$\left[Z_{\text{ROW}}^{D,D; i, i'; [t]} \right]_j = \frac{T_{j,t}}{(M-1)N(N-1)} \left\{ \left[Z_{\text{ROW}}^{\text{full sum}; [t]} \right]_j - \left[Z_{\text{ROW}}^{j'=j; [t]} \right]_j - \left[Z_{\text{ROW}}^{k'=k; [t]} \right]_j + \left[Z_{\text{ROW}}^{j'=j \& k'=k; [t]} \right]_j \right\}.$$

All the terms in these expressions can be rearranged to only involve matrix-vector products, with the exception of $D\Delta D^\top$ in the calculation of C .

The matrix D is binary, and therefore a matrix-matrix product such as DD^\top can be calculated using a binary matrix-matrix product. This binary matrix-matrix product does take $O(M^2N)$ operations, but can be computed rapidly by appealing to bitwise operations rather than floating-point operations. These bitwise operations typically allow for multiple matrix entries to be processed simultaneously. For example, if $\vec{u} = D_{j,:}$ and $\vec{v} = D_{j',:}$ are two binary row-vectors from D , then the dot-product $\vec{u}^\top \cdot \vec{v}$ can be calculated by first breaking up \vec{u} and \vec{v} into disjoint chunks, each comprising R adjacent bits:

$$\vec{u}_r = [\vec{u}_{rR-R+1}, \dots, \vec{u}_{rR}], \vec{v}_r = [\vec{v}_{rR-R+1}, \dots, \vec{v}_{rR}] \text{ for } r = 1, \dots, N/R,$$

and then using bitwise operations on each pair of chunks \vec{u}_r, \vec{v}_r :

$$\vec{u}^\top \vec{v} = N - 2 \sum_{r=1, \dots, N/R} \text{popcount}(\text{xor}(\vec{u}_r, \vec{v}_r)).$$

In the above calculation we assume that R is the number of bits that can be conveniently loaded at once (e.g., $R = 128 * (16 - 1) = 1920$), and that N is a multiple of R (or that \vec{u} and \vec{v} have been extended in length and appropriately masked or padded). The xor operation refers to a bitwise ‘exclusive-or’, and the popcount operation sums the number of nonzero bits.

Our situation is slightly more complicated than the simple example above because the diagonal matrix Δ is not binary; rather than simply forming $\vec{u}^\top \vec{v}$, we have to form $\vec{u}^\top \Delta \vec{v}$, which cannot be easily computed using only bitwise operations. Fortunately, in practice the size of Δ is usually very large (e.g., $N \gtrsim 10^5$), and the variances $\delta_k^{-1} = 4p_k q_k$ are bounded above by 1 and usually bounded below by a constant greater than 0. We take advantage of this observation by sorting the columns of D so that the diagonal entries of Δ are monotonically ordered. We then approximate each block of R entries of Δ by a single constant equal to their average:

$$\bar{\delta}_r = \frac{1}{R} (\delta_{rR-R+1} + \dots + \delta_{rR}),$$

and then calculate:

$$\vec{u}^\top \Delta \vec{v} = \sum_r \bar{\delta}_r \{N_r - 2 \cdot \text{popcount}(\text{xor}(\vec{u}_r, \vec{v}_r))\},$$

where N_r is the number of column-indices in chunk r . This procedure allows us to generate $D\Delta D^\top$, and ultimately C , by using $O(M^2N/R)$ bitwise-operations, rather than $O(M^2N)$ floating-point multiplications.

The column-sub-score $\left[Z_{\text{COL}}^{D,D; i, i'; [t]} \right]_k$ has a similar structure, and can be calculated by first computing

$$\begin{aligned} \left[Z_{\text{COL}}^{\text{full sum}; [t]} \right]_k &= \left(D_{k,:}^\top - \alpha_k 1^\top \right) \text{diag}(T_{:,t}) C \text{diag}(T_{:,t}) (D_{:,k} - 1 \cdot \alpha_k^\top), \\ \left[Z_{\text{COL}}^{k'=k; [t]} \right]_k &= \left[\left(1_k 1^\top - 2D_{k,:}^\top \cdot * \alpha_k 1^\top + \alpha_k^2 1^\top \right) T_{:,t} \right] \cdot \left[\left(1_k 1^\top - 2D_{k,:}^\top \cdot * \alpha_k 1^\top + \alpha_k^2 1^\top \right) T_{:,t} \right], \end{aligned}$$

$$\begin{aligned} \left[Z_{\text{COL}}^{j'=j; [t]} \right]_k &= \left(\mathbf{1}_k \mathbf{1}^\top - 2D_{k,:}^\top * \alpha_k \mathbf{1}^\top + \alpha_k^2 \mathbf{1}^\top \right) \cdot \text{diag}(T_{:,t}) \cdot (\mathbf{1}\mathbf{1}^\top - 2D * \mathbf{1}\alpha^\top + \mathbf{1}\alpha^2\mathbf{1}^\top) \Delta \mathbf{1}, \\ \left[Z_{\text{COL}}^{j'=j \& k'=k; [t]} \right]_k &= \Delta_{k,k} \cdot \left[\mathbf{1}_k \mathbf{1}^\top - 4 \cdot D_{k,:}^\top * \alpha_k \mathbf{1}^\top + 6 \cdot \alpha_k^2 \mathbf{1}^\top - 4 \cdot D_{k,:}^\top * \alpha_k^3 \mathbf{1}^\top + \alpha_k^4 \mathbf{1}^\top \right] \cdot T_{:,t}^2, \end{aligned}$$

and then forming the combination:

$$\left[Z_{\text{COL}}^{D,D; i,i'; [t]} \right]_k = \frac{\Delta_{k,k}}{(N-1)M(M-1)} \left\{ \left[Z_{\text{COL}}^{\text{full sum}; [t]} \right]_k - \left[Z_{\text{COL}}^{k'=k; [t]} \right]_k - \left[Z_{\text{COL}}^{j'=j; [t]} \right]_k + \left[Z_{\text{COL}}^{j'=j \& k'=k; [t]} \right]_k \right\}.$$

As in the case with the row-sub-score, all the expressions within this column-sub-score involve matrix-vector products, with the exception of $D\Delta D^\top$ in C , which requires a matrix-matrix product.

Re-Binarization of C : We pause to remark on the calculation of

$$\left[Z_{\text{COL}}^{\text{temp}; [t]} \right]_k := D_{k,:}^\top \text{diag}(T_{:,t}) C \text{diag}(T_{:,t}) D_{:,k} \text{ within } \left[Z_{\text{COL}}^{\text{full sum}; [t]} \right]_k.$$

In practice N is typically much larger than any M_{Q_i} ; i.e., for genome-wide-association-studies (GWAS) the total number of genetic-measurements (i.e., SNPs) can be on the order of 10^{5-6} or more, while the total number of patients is only on the order of 10^{3-4} . This implies that, even though calculating $D_{k,:}^\top \text{diag}(T) C \text{diag}(T_{:,t}) D_{:,k}$ for each k is technically $O(NM)$ floating-point operations, this calculation can often be the bottleneck for our entire algorithm. In these cases it is advantageous to take advantage of the fact that both D and T are binary-matrices. Although C is not a binary-matrix itself, we can represent C using a collection of b binary-matrices (one such binary-matrix for each significant bit of C we wish to preserve). This kind of representation allows us to calculate $Z_{\text{COL}}^{\text{temp}; [t]}$ using b binary matrix-matrix calculations, each requiring $O(NM^2/R)$ bitwise-operations. We find that this procedure is often more efficient than the direct calculation of $Z_{\text{COL}}^{\text{temp}; [t]}$, and is sufficiently precise so long as we preserve sufficiently many bits of C (i.e., if b is sufficiently high). When the sparsity-coefficients p_k are all close to 0.5, such as in gene-expression-analysis, then 8 bits of C are typically sufficient (i.e., we choose $b \gtrsim 8$). In cases where the sparsity coefficients p_k extend from close to 0 up to 0.5, such as in GWAS, more bits of C are necessary, and we typically choose $b \gtrsim 24$. This re-binarization can be applied not only to the matrix C that we introduced at the beginning of section 13, but also to the other correlation-matrices involved in the calculation of $\left[Z_{\text{ROW}}^{Q,Q'; i,i'; [t]} \right]_j$ and $\left[Z_{\text{COL}}^{Q,Q'; i,i'; [t]} \right]_k$ which have a similar structure to C .

More rows than columns: Some applications involve data-matrices D and X with more rows than columns (e.g., $M_Q > N$). In these situations the calculations above are not efficient; they require $O(M_Q^2 N)$ work, when $O(M_Q N^2)$ work would be preferable. To arrange the calculation so that only $O(M_Q N^2)$ work is required, we can first calculate the $N \times N$ matrix E :

$$E = (D^\top - \alpha \mathbf{1}^\top) \text{diag}(T_{:,t}) (D - \mathbf{1}\alpha^\top),$$

and then compute

$$\begin{aligned} \left[Z_{\text{ROW}}^{\text{full sum}; [t]} \right]_j &= (D - \mathbf{1}\alpha^\top) \cdot \Delta \cdot E \cdot \Delta \cdot (D^\top - \alpha \mathbf{1}^\top), \text{ and} \\ \left[Z_{\text{COL}}^{\text{full sum}; [t]} \right]_k &= E_{k,:} \cdot \Delta \cdot E_{:,k} = E_{k,:}^2 \cdot \text{diag}(\Delta). \end{aligned}$$

Parallelization and Timing: The full algorithm described in section 12 involves both case- and control-matrices, as well as I_{cat} covariate categories and N_T continuous-covariate coefficients. There are many different ways to organize this calculation, with different considerations pertaining to different computing architectures. Our current implementation at the time of this writing uses different nodes to process the different trials (e.g., the original data as well as label-shuffled-trials from H0x). For each trial running on a given node, we use different processors within the node to calculate the scores associated with different combinations of Q' , i , i' , and t . This implementation is certainly not always optimal (e.g., in the D -only case), but is reasonably efficient for many applications. For example, the GWAS data-set shown in Example-3 was processed over a weekend; the computation used 65 nodes (one for the original-data, and 64 others for the label-shuffled-trials from H0x), each with 16 processors (Intel Xeon Processor E5-2650 v2). The total computation-time per-node was $\lesssim 12$ hours, with γ set to 0.05 and $b = 32$ bits retained for each correlation-matrix. If, instead, we had only retained $b = 8$ bits (which is sufficient for many problems) and not corrected for the $N_T = 2$ continuous covariates, then the total computation-time per-node would have been reduced by a factor of ~ 12 . If necessary, a further reduction in computation-time could be obtained by increasing γ .

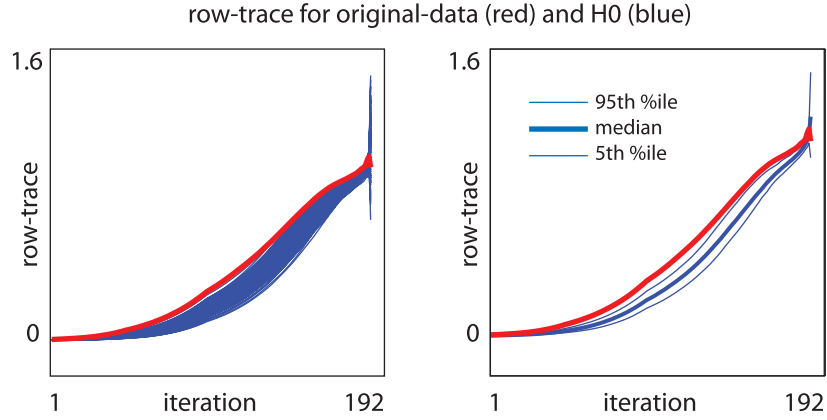


Figure 57: Row-traces for the bicluster shown in Example-1b from section 1.2. This bicluster was found by running our loop-counting algorithm on the data shown in Fig 5. For this example we corrected for cases-vs-controls; we compare our original-data to the distribution we obtain under the null-hypothesis H_0 . On the left we show the row-trace as a function of iteration for the original-data (red) as well as each of the 256 random shuffles (blue). On the right we replot this same trace data, showing the 5th, 50th and 95th percentile (across iterations) of the H_0 distribution. Because we are not correcting for any covariates, the column-traces are identical to the row-traces.

14 Notes regarding application

Our biclustering algorithm produces three lists as output. The first lists the rows and columns of the data-matrix in the order in which they were eliminated. The second lists the average row-score taken across the remaining data-matrix at each iteration. The third lists the average column-score taken across the remaining data-matrix at each iteration. We refer to these average row- and column-scores as row- and column-traces; they are both equal to the trace of DD^TDD^T in the D -only situation, but will in general be different when correcting for cases-vs-controls and covariates. In practice it is often critical to determine

- How to interpret the algorithm’s output,
- How significant this output is,
- Whether or not a bicluster was found, and what the boundaries of the bicluster are.
- How to find the second, third, and subsequent biclusters.

The answers to these questions are, in general, application dependent. We comment on some of our experiences below.

14.1 Interpreting the output:

We have run our algorithm in the context of gene-expression analysis and genome-wide-association-studies (i.e., GWAS). In both cases we were comparing case-patients to control-patients, at times correcting for various covariates. In the context of gene-expression-analysis we had the luxury of normalizing the data so that the sparsity-coefficient for each column was $p_k = 0.5$. This choice of normalization meant that we did not need to correct for sparsity; each $\alpha_k = 0$ and $\delta_k = 1$. Because each loop contributed either a $+1$ or a -1 to our total score, the row- and column-traces produced by loops within D were bounded between -1 and $+1$. Similarly, the traces produced by loops entering X were also bounded between ± 1 . The full trace produced by our algorithm was thus bounded by ± 2 , and was usually in the interval $[0, 1]$. We could interpret the output in this situation relatively easily: If, after sufficiently many iterations, the row- or column-traces grew large (i.e., came close to $+1$), then the remaining rows and columns should form a highly correlated low-rank bicluster. An illustration along these lines is shown in Figs 57 and 58, which corresponds to our Example-1b for gene-expression-analysis. A similar illustration is given in Figs 59 and 60, which corresponds to our Example-2 for gene-expression-analysis (which involves categorical covariates). Similarly, Figs 61 and 62 correspond to Example-1a.

For GWAS the same interpretation is not valid; the GWAS data is itself binary because each allele is either dominant or recessive. This means that we do not have the luxury of normalizing the data to alter the sparsity-coefficients; typically the sparsity-coefficients for genotyped-data range from close to 0 all the way up to 0.5. This range of sparsity-coefficients makes interpreting the output of our algorithm more difficult. It is now possible for any loop to contribute a large magnitude to the total scores. The row- and column-traces are no longer bounded by a small number like ± 1 or ± 2 , and often rise to be much higher towards the end of the iterations when the few remaining columns have extreme sparsity-coefficients. In

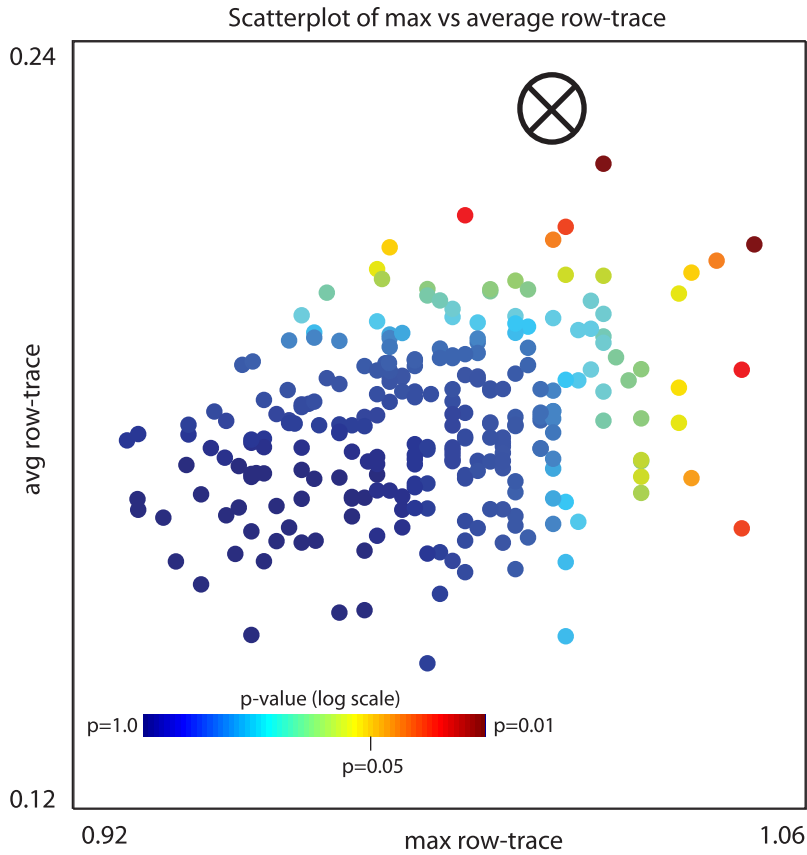


Figure 58: We show a scatterplot of the data shown in Fig 57. Each row-trace shown on the left in Fig 57 is plotted as a single point in 2-dimensional space; the horizontal-axis corresponds to the maximum row-trace and the vertical-axis corresponds to the average row-trace (taken across the iterations). The original-data is indicated with a ‘ \otimes ’, and each of the random shuffles with a colored ‘ \bullet ’. The p -value for any point \vec{w} in this plane is equal to the fraction of label-shuffled-traces that have either an x -position larger than x_w or a y -position larger than y_w , where x_w and y_w are the x - and y -percentiles associated with the most extreme coordinate of \vec{w} (details given in section 14.2). Each random shuffle is colored by its p -value determined by the label-shuffled-distribution. By comparing the original-trace with the shuffled-distribution we can read off a p -value for the original-data of $\lesssim 0.008$.

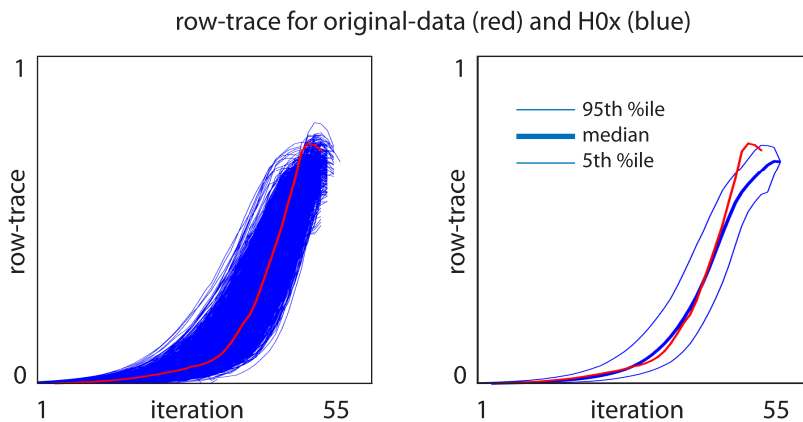


Figure 59: Row-traces for the bicluster shown in Example 2 in section 1.3. This bicluster was found by running our loop-counting algorithm on the data shown in Fig 8. For this example we corrected for cases-vs-controls as well as the categorical-covariates of gender and ethnicity. Because we corrected for categorical-covariates, we compare our original-data to the distribution we obtain under the null-hypothesis H_{0x} . On the left we show the row-trace as a function of iteration for the original-data (red) as well as each of the 2048 random shuffles (blue). Note that the different traces terminate at different iteration-numbers. This is due to the fact that different trials eliminate the various covariate-categories at different iteration-numbers; for each trace we only include iterations for which all covariate-categories remain. On the right we replot this same trace data, showing the 5th, 50th and 95th percentile (across iterations) of the H_{0x} distribution. While not exactly identical to the row-traces, the column-traces are very similar (not shown).

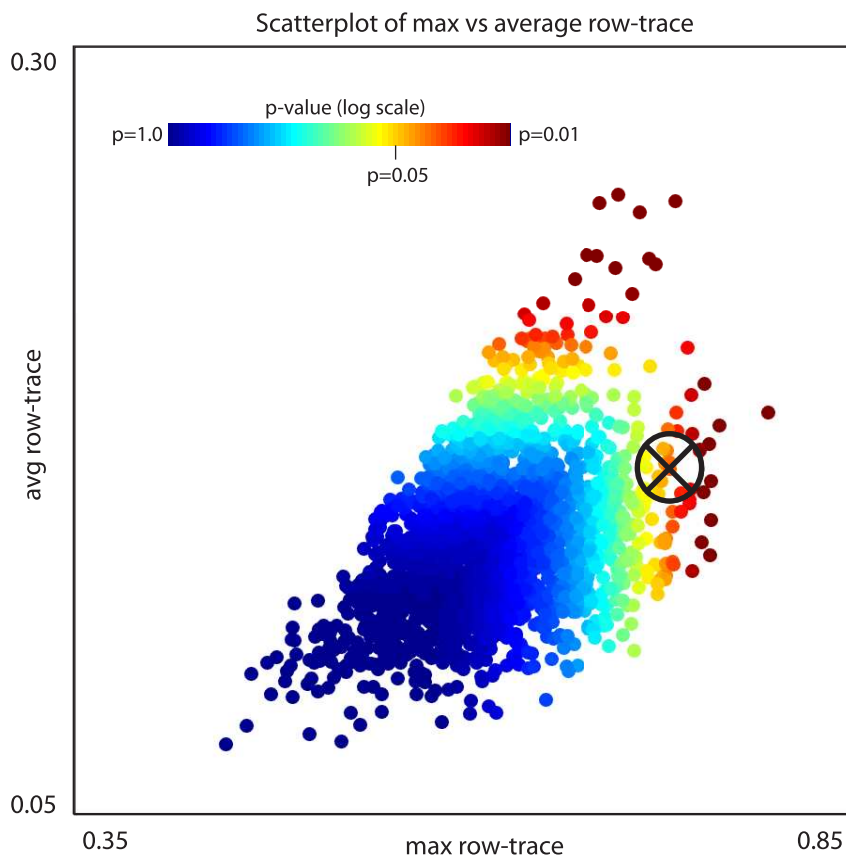


Figure 60: We show a scatterplot of the data shown in Fig 59. The format of this figure is similar to that of Fig 58. By comparing the original-trace with the shuffled-distribution we can read off a p-value for the original-data of ~ 0.027 .

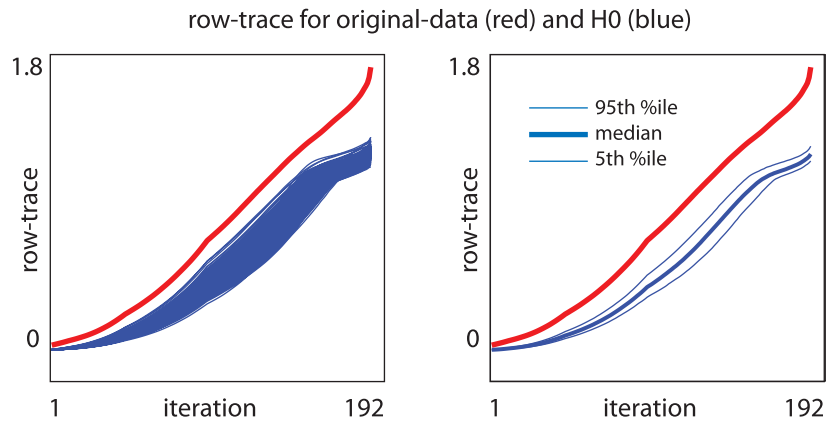


Figure 61: Row-traces for the bicluster shown in Example-1a from section 1.2. This bicluster was found by running our half-loop algorithm on the data shown in Fig 2. For this example we corrected for cases-vs-controls; we compare our original-data to the distribution we obtain under the null-hypothesis H_0 . On the left we show the row-trace as a function of iteration for the original-data (red) as well as each of the 512 random shuffles (blue). On the right we replot this same trace data, showing the 5th, 50th and 95th percentile (across iterations) of the H_0 distribution. Because we are not correcting for any covariates, the column-traces are identical to the row-traces.

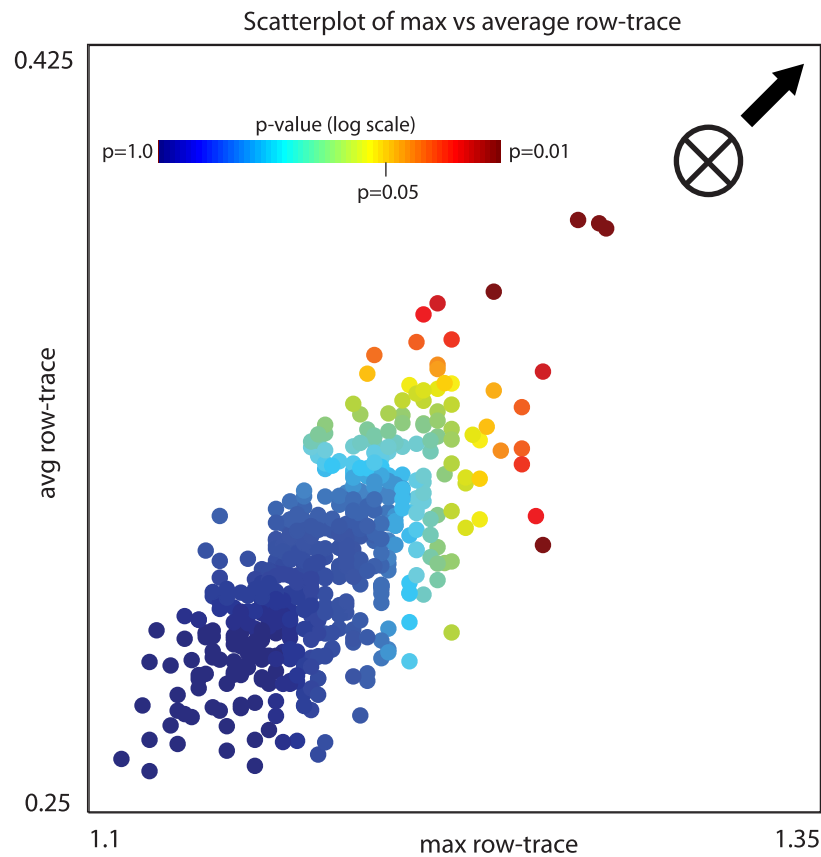


Figure 62: We show a scatterplot of the data shown in Fig 61. The format for this scatterplot is similar to that shown in Fig 58, except that the original-trace lies far to the upper-right of the distribution drawn from H_0 (see arrowhead). By comparing the original-trace with the shuffled-distribution we can read off a p-value for the original-data of $\lesssim 1/512$ (i.e., bounded above by the reciprocal of the number of samples drawn from H_0).

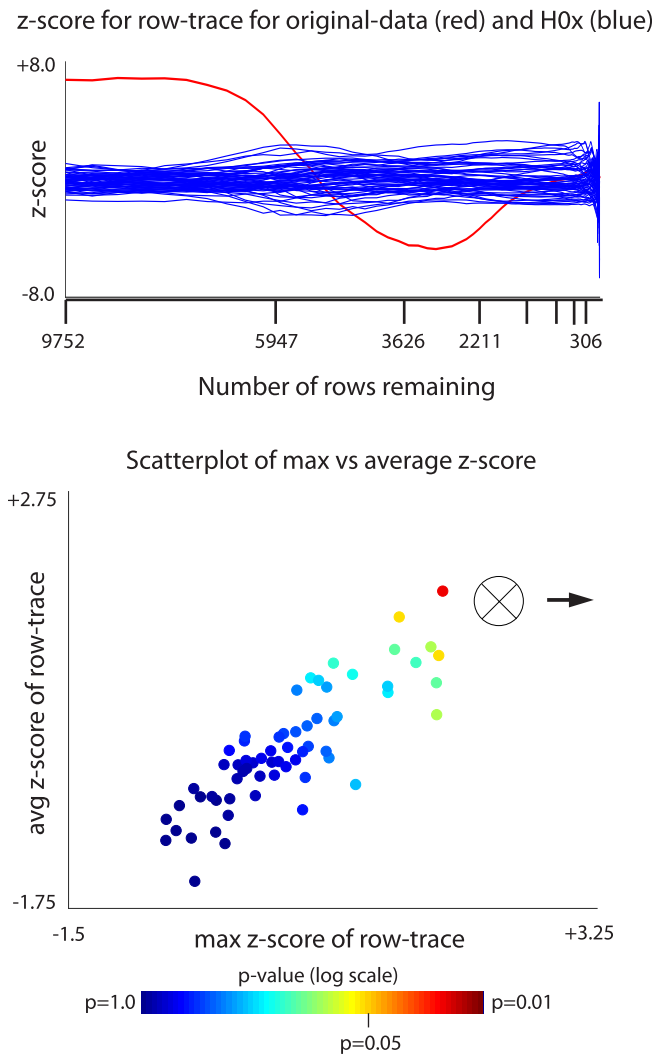


Figure 63: Z-scores for the bicluster shown in section 1.4. This bicluster was found by running our algorithm on the GWAS data-set referenced in 1.4. For this example we corrected for cases-vs-controls, as well as continuous-covariates and sparsity. Because we corrected for continuous-covariates, we compare our original-data to the distribution obtained under H0x. Because the data-set had many columns with extreme sparsity-coefficients, the magnitude of our row- and column-traces is not particularly meaningful by itself. A more meaningful measurement is the z-score of the row- and column-traces (calculated with respect to the label-shuffled distribution). On top we plot this z-score as a function of the number of rows remaining. The original-trace is shown in red, and the 64 random shuffles from H0x are shown in blue. On the bottom we show a scatterplot of the maximum z-score (horizontal) and average z-score (vertical) taken across the iterations. The format for this scatterplot is similar to that shown in Fig 58, except that the original-trace lies far to the right of the distribution drawn from H0x (see arrowhead). By comparing the original-trace to the shuffled-distribution, we can read off a p-value of $\lesssim 1/64$ (i.e., bounded above by the reciprocal of the number of samples drawn from H0x).

this situation the most natural way to determine whether a low-rank structure has been found is to compare the traces for the original data to traces obtained after label-shuffling the data (see the next subsection).

14.2 Determining significance:

Our typical hypothesis (say, H1) is that there exists some disease-related structure in the case-patients that is not exhibited by the control-patients. This is in contrast to the null-hypothesis (H0) in which the case- and control-labels are actually arranged randomly and have no disease-related structure. Under this null-hypothesis the traces we observe after running our algorithm on the original data should be similar to the traces we would find if we were to shuffle the case-control labels randomly (across patients). To draw a sample from this null-hypothesis H0, we shuffle the case-control labels of the patients indiscriminantly while retaining the same number of cases and controls. If we are correcting for covariates, then we restrict our null-hypothesis slightly (H0x) so that the random-shuffles respect the covariates. For example, if we were correcting for gender as a categorical-covariate, we would shuffle the labels of case-males only with control-males, and shuffle the labels of case-females only with control-females. If we were correcting for a continuous-covariate, we would shuffle the case-control labels of patients in each orthant of covariate-space.

For each label-shuffled trial we rerun our algorithm, and collect the output. Each trial does not depend on any of the other trials, and they can all be processed in parallel. This library of label-shuffled traces produces a distribution associated with H0 (or H0x). We use this label-shuffled distribution to calculate a p-value for the traces produced by the original data.

Example-1 and Example-2:

The row- and column-traces each have one value per iteration. Two of the most straightforward measurements for each trace are (i) the maximum value, and (ii) the average value, with the latter computed using a measure where each trace-value contributes a weight proportional to the number of rows or columns eliminated during that iteration. When considering gene-expression-data these two measurements are useful and have a natural interpretation. On one hand, (i) traces that reach a high maximum during a late iteration correspond to small but tightly-correlated biclusters. On the other hand, (ii) traces that may not have an exceptionally high maximum value but are consistently above the median for many consecutive iterations correspond to larger but more loosely-correlated biclusters. The biclusters shown in Example-1a, Example-1b and Example-2 illustrate the first phenomenon (i), as shown in Figs 61, 57 and 59.

We can compare our original-data with the label-shuffled-distribution to obtain a p-value for our original-trace. We have done this using the 2-dimensional scatterplots shown in Figs 62, 58 and 60. For illustration, consider Fig 60. Each of the $L = 2048$ label-shuffled-traces has an x - and a y -position on this scatterplot. We'll denote the position of trace- l by (x_l, y_l) . Let's also denote by $F(x)$ the fraction of label-shuffled traces with an $x_l < x$, and by $G(y)$ the fraction of label-shuffled traces with a $y_l < y$. Let's further denote by $x(F)$ the x -position of the F^{th} -percentile of the x_l samples (i.e., $x(F)$ is the inverse of $F(x)$). Let's also denote by $y(G)$ the y -position of the G^{th} -percentile of the y_l samples (i.e., $y(G)$ is the inverse of $G(y)$). Now we can calculate the p-value of any location $\vec{w} = (w_x, w_y)$ in the x, y -plane as follows.

1. Calculate $F(w_x)$ and $G(w_y)$,
2. Let $H = \max(F(w_x), G(w_y))$ be the more extreme of the two.
3. Calculate $x(H)$ and $y(H)$,
4. find the set $\Omega = \{l \text{ such that either } x_l \geq x(H) \text{ or } y_l \geq y(H)\}$.
5. Set the p -value for \vec{w} to be the fraction of label-shuffled-traces in Ω : i.e., $p\text{-value} = p(w_x, w_y) = |\Omega|/L$.

We remark that this method of calculating p -values is 'self-consistent'; the distribution of p -values across the label-shuffled traces will be uniform by construction. Specifically, the collection of $p(x_l, w_l)$ taken across l will be uniformly distributed in $[0, 1]$ and the empirical cdf of these p -values will be a diagonal line with slope 1. The more standard bonferroni-correction $p = \max(F(w_x), G(w_y))$ is an overestimate and will not produce a uniform distribution of p -values across the label-shuffled traces when F and G are correlated.

Note that the label-shuffled null-hypothesis H0x is 'nested-within' (and more restrictive than) the null-hypothesis H0. Specifically, the collection of label-permutations allowed under H0x is a subset of the label-permutations allowed under H0. The label-shuffles which mix patients in different covariate categories (allowed under H0) are typically less likely to exhibit the same structures as the label-shuffles allowed under H0x. As a result, a smaller fraction of the trials drawn from H0 are likely to 'look like the real data', as compared with H0x. Consequently, the p-value obtained from the H0-distribution will typically be smaller (i.e., indicating a more significant result) than the more conservative p-value obtained using the more reasonable H0x-distribution.

Example-3:

When considering GWAS data the maximum and average of the trace are not useful measurements because of the varying sparsity-coefficients in the data. Because of these varying sparsity-coefficients, the trace can take on values far outside the interval $[-1, 1]$, as is typical when there are only a few iterations remaining. Standard measurements such as the max- and average-trace are strongly influenced by the values of the trace during the final few iterations. To account for this bias, we transform each trace by replacing its value at each iteration with its z-score calculated with respect to the distribution of the label-shuffled-traces at that iteration. The max and average of the transformed-traces have an analogous interpretation to the original measures (i) and (ii) described earlier, except that the correlations revealed should be interpreted relative to the label-shuffled distribution, rather than in an absolute sense. Traces that reach a high z-score during a late iteration correspond to biclusters which, although small, are more tightly-correlated than the typical small bicluster seen at that iteration. Traces that have a high average z-score correspond to biclusters that are larger, but somewhat less correlated. The bicluster shown in Example-3 illustrates this latter phenomenon, as shown in Fig 63.

14.3 Delineating a bicluster:

Example-1 and Example-2:

One way to visualize the structure highlighted by the biclustering algorithm is to calculate a ‘shape-matrix’ S . To create S we first order the rows and columns of the original data-matrix in terms of their importance (i.e., the longest retained rows and columns come first). With this ordering we define S_{jk} as the average trace of the ‘corner-submatrix’ obtained by considering the j most important rows and the k most important columns. This calculation can be performed for each j and k , revealing the subsets of rows and columns for which the trace is high. The decay of S_{jk} as j and k increase will indicate the shape of the structure found. We illustrate some shape-matrices in Figs 64 and Figs 65, which correspond to our Example-1a and Example-1b. The shape-matrix in Fig 66 corresponds to our Example-2, while the shape-matrices in Fig 67 correspond to some other biclusters we have found in other gene-expression data-sets.

We note that, towards the end of our iteration, there are not always sufficiently many patients remaining to represent all the covariate-categories (e.g., in Fig 66 $I_{\text{cat}} = 4$). When the number of remaining covariate-categories drops (e.g., when the last noncaucasian female is eliminated), our algorithm reduces the number of covariate-categories I_{cat} involved in the score-calculation, and reduces I_{req} as necessary. This abrupt change in I_{cat} and I_{req} results in the discontinuities of S in the j -direction seen in Fig 66B. When defining p -values and delineating the bicluster we only consider iterations where all the initial covariate-categories are represented.

To actually delineate a bicluster we invoke a user-specified threshold $\tau \in (0, 1)$; in this case $\tau = 0.7$. Given τ , We find the j, k for which S_{jk} is equal to the fraction τ of its maximum, and jk is as large as possible. In other words, we find the largest corner-submatrix with average-trace τS_{max} . As an example, the biclusters shown in Figs 64, 65 and 66 were delineated using this technique.

This procedure (i.e., picking out a corner-submatrix) does indeed do a reasonable job of delineating rectangular biclusters when they exist, and is not very sensitive to τ when the rectangular biclusters are sharply defined. Nevertheless, this kind of delineation does not always tell the whole story; corner-submatrices are not always the best indicator of the various sorts of structure that can be revealed by S in practice. For example, as seen in Figs 64 and 67, our algorithm can often reveal patterns within gene-expression data that are not quite rectangular. These shape-matrices may indicate some of the more complicated underlying structures discussed in section 7.3. Thus, in summary, it is important to note that bicluster-delineation will always be somewhat arbitrary; in practice there will rarely be a clear-cut boundary to any bicluster. Because of this arbitrariness, we have structured our methodology so that the details regarding bicluster-delineation do not affect the bicluster’s p -value (which only depends on the traces, and not the threshold τ).

Example-3:

The S -matrix described above is useful when the goal is to pick out a small sharply-correlated bicluster – i.e., when the traces output by our algorithm are significantly high towards the end of the iterations. However, if the output-traces are not high towards the end of the iterations, then this S -matrix may not be particularly meaningful. This situation arises in Example-3, where the output-traces are significantly higher than the H0x-distribution for the first several iterations, but not towards the end (see Fig 63). If we were to try and use the S -matrix to delineate this bicluster, then we would need to compare the S_{jk} produced by the original-data with the S_{jk} produced by the random-shuffles from H0x; the former would only be significantly greater than the latter when j is in the thousands and k is in the hundreds-of-thousands – clearly not a sharp delineation of any kind. Thus, for Example-3, we choose to delineate the bicluster by using the last peak of the output-trace relative to the H0x-distribution (i.e., the last peak of the z-score). This final peak occurs with 115 case-patients and 706 alleles remaining (i.e., the submatrix shown in Fig 11).

14.4 Finding secondary biclusters:

So far we have been concerned with finding a single bicluster hidden within a large data-matrix. In practice of course, many large data-sets are home to multiple biclusters, some of which are distinct and some of which overlap to varying degrees.

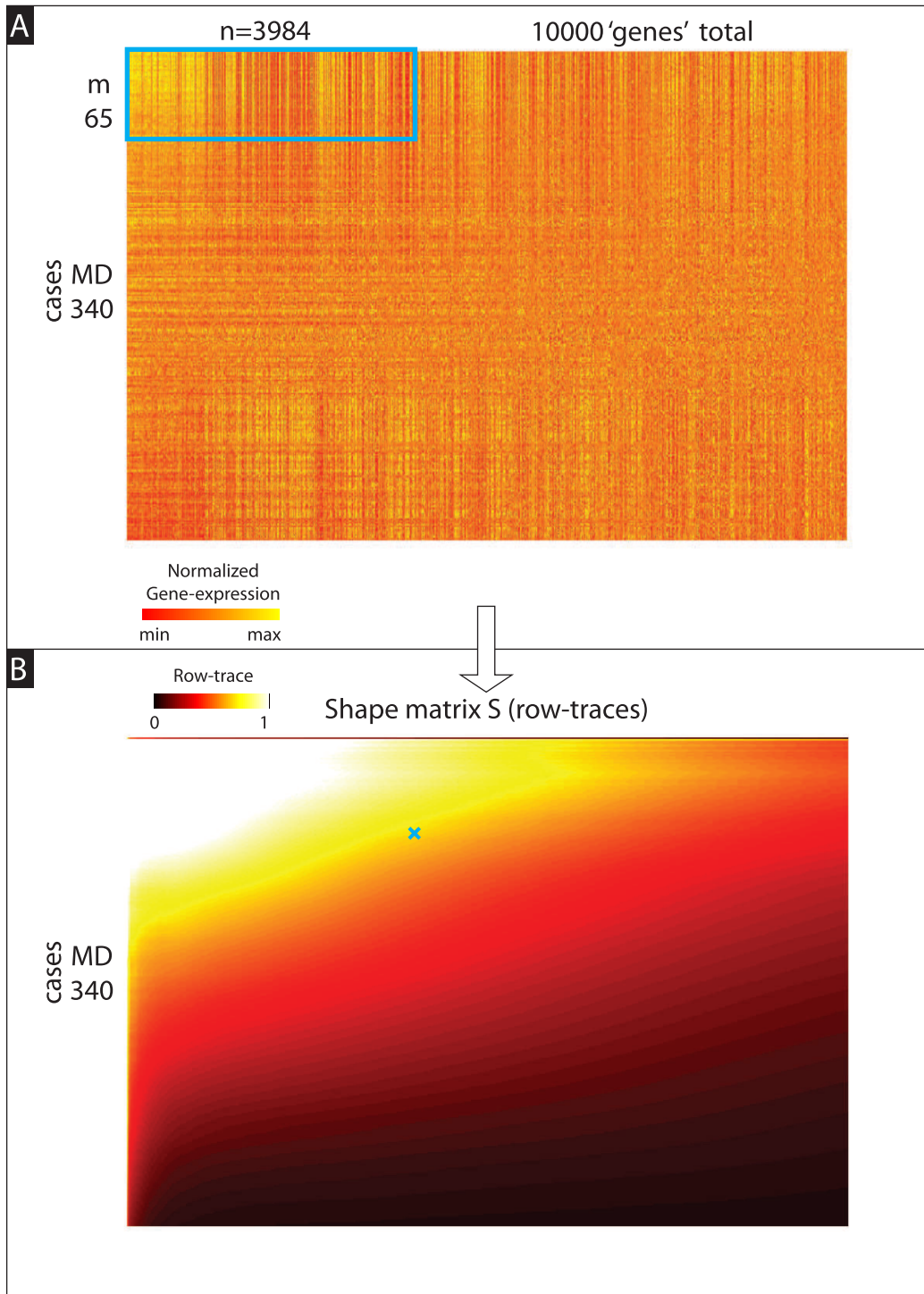


Figure 64: Illustration of ‘shape-matrix’ S for Example-1a shown in Fig 3. In Panel-A we show a subset of the D -matrix rearranged according to the output of our algorithm, with the most important rows and columns listed first. We show all $M_D = 340$ patients, but only the 10000 most important of the $N = 11711$ genes. The upper left corner holds the bicluster shown in Fig 3A, as indicated by the cyan box. In Panel-B we show the shape-matrix S , which records the row-trace for each corner-submatrix of the output shown in Panel-A. Specifically, given the rearranged version of D shown in Panel-A, we calculate S_{jk} to be the row-trace associated with rows $\{1, \dots, j\}$ and columns $\{1, \dots, k\}$. The entry of S for $j = 65$ and $k = 3984$ corresponds to the submatrix within the cyan box, and is indicated with a cyan ‘x’. This shape-matrix reveals a rather smeared-out bicluster; while there are many rows and columns that clearly participate, the shape of the bicluster is not obviously rectangular. Consequently, the delineation of a ‘corner-submatrix’ will depend somewhat sensitively on the methodology used (e.g., the particular value of τ we use to choose the corner).

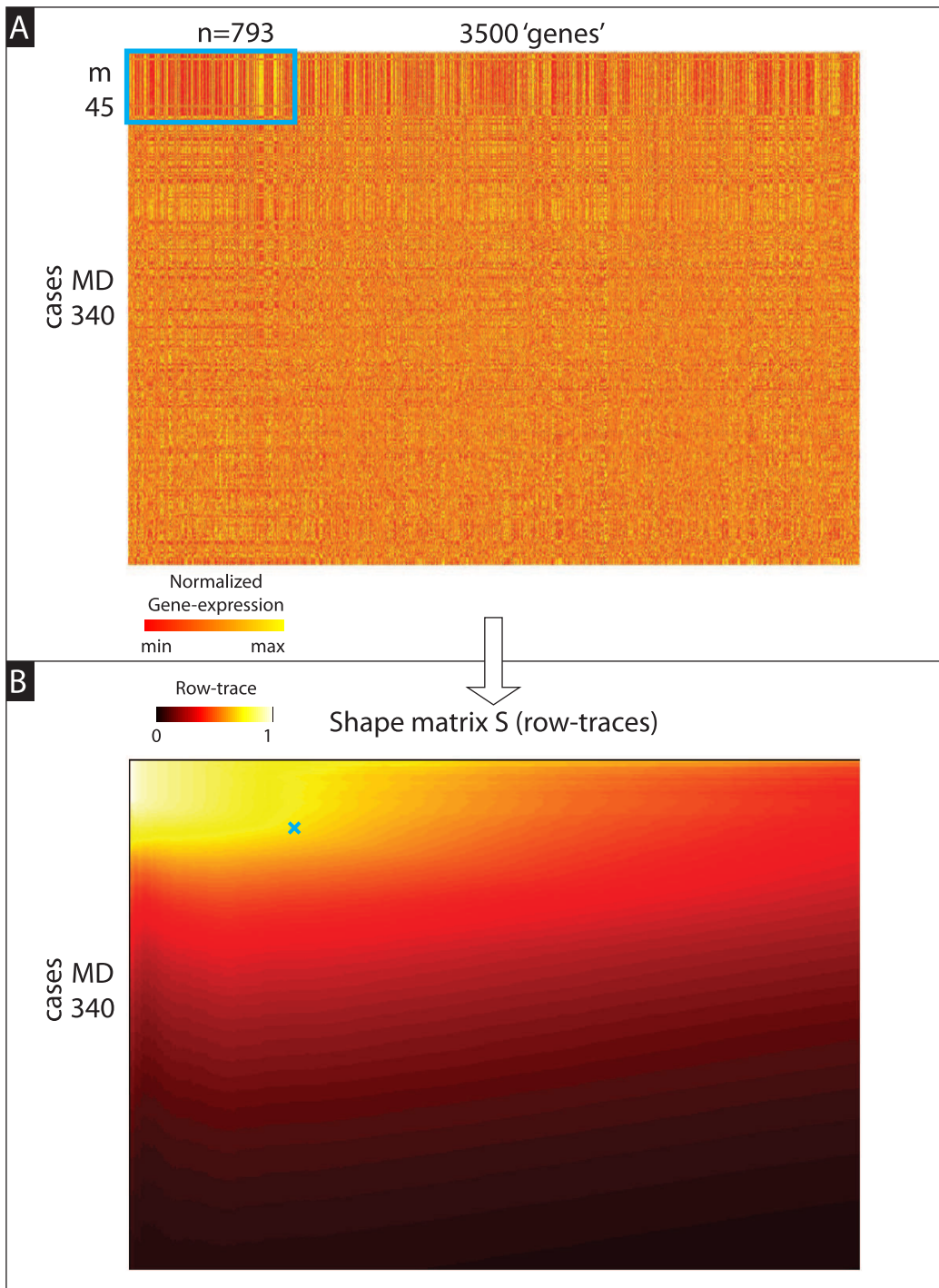


Figure 65: Illustration of 'shape-matrix' S for Example-1b shown in Fig 6. The format of this figure is similar to that of Fig 64. We show all $M_D = 340$ patients, but only the 10000 most important of the $N = 16738$ genes. The entry of S for $j = 45$ and $k = 793$ corresponds to the submatrix within the cyan box, and is indicated with a cyan 'x'. This shape-matrix reveals a rather rectangular bicluster; the delineation of this bicluster is not very sensitive to our threshold of $\tau = 0.7$.

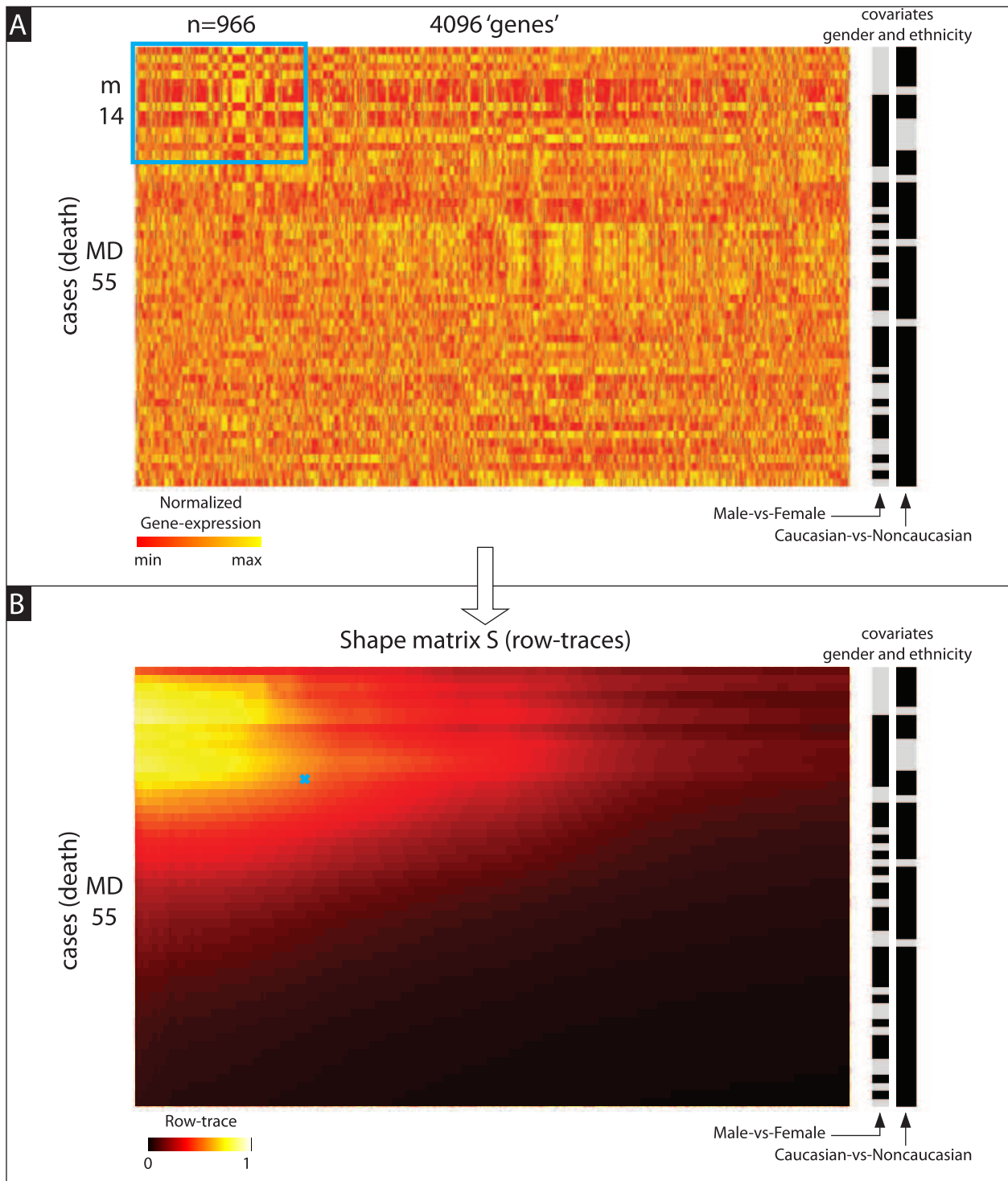


Figure 66: Illustration of 'shape-matrix' S for Example-2 shown in Fig 9. The format of this figure is similar to that of Fig 64. We show all $M_D = 55$ patients, but only the 4096 most important of the $N = 17942$ genes. The entry of S for $j = 14$ and $k = 966$ corresponds to the submatrix within the cyan box, and is indicated with a cyan 'x'. This shape-matrix reveals a rather rectangular bicluster; the delineation of this bicluster is not very sensitive to our threshold of $\tau = 0.7$.

Various shape matrices for other biclusters

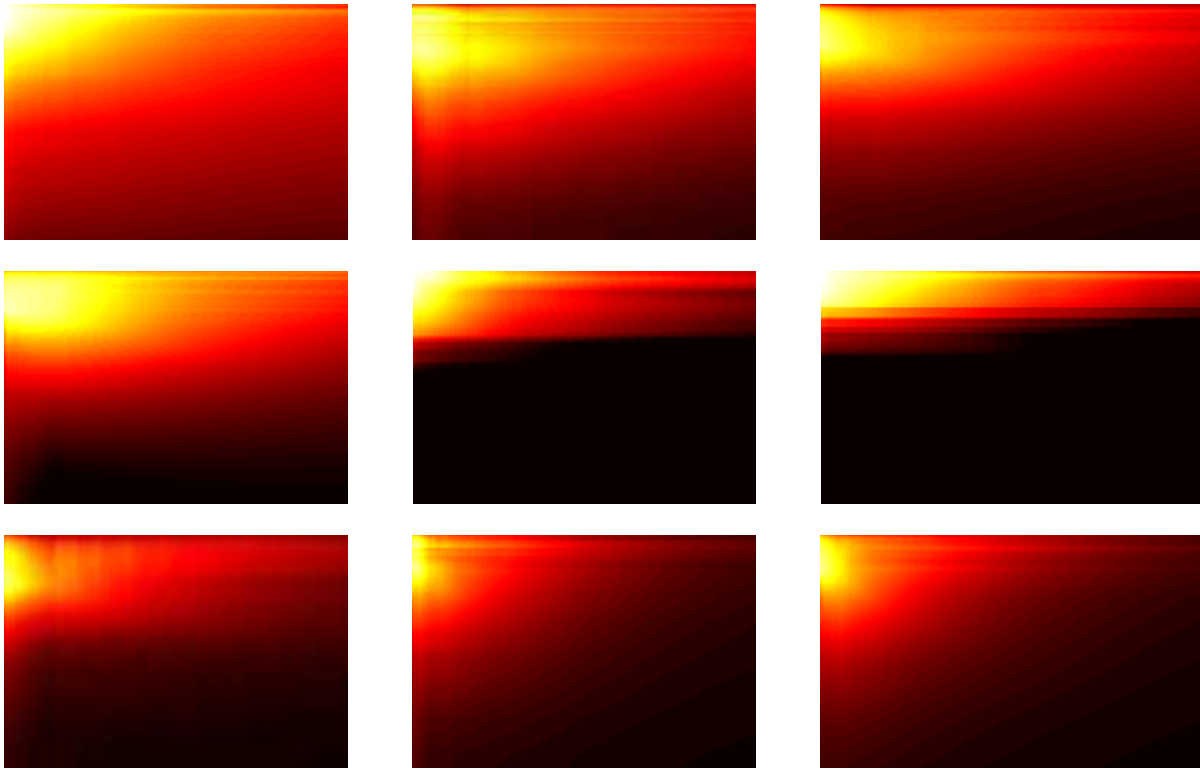


Figure 67: Illustration of some other shape-matrices we obtained when running our algorithm on various gene-expression data-sets. Some of these shape-matrices suggest rectangular biclusters, but others suggest more ‘smeared-out’ structures, some similar to the kinds of structures discussed in section 7.3.

What does our algorithm do when the data-set has multiple biclusters, and how do we find these multiple biclusters?

As we’ve discussed above in section 4, our algorithm usually finds the most ‘dominant’ bicluster within the data; that is, the bicluster that produces the greatest signal (i.e., the largest and/or most tightly correlated bicluster). This first most dominant bicluster will often be highlighted by the corners of the shape matrix ‘S’ described in section 14.3. In some cases secondary biclusters will also be highlighted by the shape matrix, trailing the first (see Fig 33).

Situations as clean as Fig 33 don’t usually arise in practice, however, and we’ve found it useful to search for secondary biclusters using repeated passes of our algorithm. More specifically, after running our algorithm for the first time (and delineating the most dominant bicluster), we then rerun our algorithm from scratch, scrambling the data within the first detected bicluster to destroy any signal it might create. This second pass of our algorithm will then locate the second most dominant bicluster. After this we can perform a third pass, scrambling the first two biclusters and so forth.

Note that when we scramble the data within the biclusters we have found so far, we should respect the constraints compatible with our null hypothesis (e.g., H_0 , H_{0x} , etc.). For example, if we were considering categorical-covariates we should scramble the data within a bicluster by randomly interchanging/replacing matrix-entries within (but not across) covariate-categories. This procedure ensures that we destroy (most of) the signal associated with the previously discovered biclusters without introducing any large new spurious signals.

We remark that secondary passes of our algorithm only rarely produce traces that are as high as those produced by the first pass (after all, the secondary biclusters typically have a weaker signal than the first). Nevertheless, we currently believe that it is best to assess the p-value of secondary biclusters by comparing their traces with the original shuffled-distribution of traces used to assess the most dominant bicluster (rather than with a new shuffled-distribution associated with second passes of our algorithm). We adopt this methodology because it is conservative: we use the same standard to assess both secondary and primary biclusters, even though this standard may underestimate the significance of secondary biclusters.

The results of this approach for Example-2 are shown in Fig 68. Note that the first trace is the most pronounced. This phenomenon is typical for the gene-expression data sets we have analyzed; while certainly not impossible, it is rare for a secondary bicluster to be more significant than a previously found bicluster.

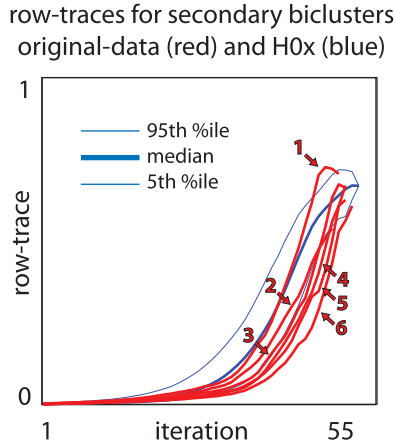


Figure 68: Here we show the row-traces for the secondary biclusters associated with Example-2 from section 1.3. These biclusters were found by successively rerunning our algorithm on the data shown in Fig 8 after scrambling any previously found biclusters. In the background we indicate the distribution of traces drawn from the shuffled-distribution (blue lines). These are identical to those shown in Fig 59. In the foreground we indicate the row-traces for the various passes of our algorithm (red numbers and lines). The first pass (corresponding to the most dominant bicluster) was shown previously in Fig 59).

15 Additional Applications

We have focused so far on the detection of low-rank biclusters. The simplest situation involves locating a ‘rank-1’ bicluster B hidden within an $M \times N$ data-matrix D . Because the bicluster B can be approximated with a single outer-product $u \cdot v^T$, we have chosen to consider loops (i.e., 2×2 submatrices) within D . The low-rank structure of B ensures that loops within B are likely to be rank-1 (rather than rank-2). Moreover, the rows and columns of D that intersect B will contribute to many more rank-1 loops than the rows and columns that don’t. Using these observations, our algorithms accumulates the loops within D ; exposing B in the process. We’ve also discussed how to search for low-rank biclusters within a data-matrix involving cases and controls, considering along the way how to account for sparsity as well as categorical- and continuous-covariates.

Our methods can be easily extended to tackle many related problem. For example, we can treat ‘genetic controls’, look for ‘rank-0’ biclusters (i.e., differentially-expressed biclusters), and even look for ‘triclusters’. We discuss these topics briefly below.

15.1 Accounting for genetic-controls:

In some applications the goal is to find biclusters that are not merely specific to a subset of case-rows, but also to a subset of case-columns. Such a situation may arise in gene-expression analysis when the goal is to find biclusters restricted to one set of genes that do not extend to encompass the gene-expression measurements taken from another set of genes. As an example, let’s return to the Example-2 in section 1.3. The particular bicluster we already found in section 1.3 involved many genes and pathways that are already known to affect colon-cancer; these genes are involved in many pathways related to DNA-replication, DNA-repair, mitosis, etc. Instead of focusing on these well-known pathways, we might want to deliberately ignore them and search for signals within other genes; hopefully discovering genes that participate in less well understood pathways.

One way to go about this is to classify some of the genes in our data-set as ‘control-genes’; i.e., genes that we deliberately want to avoid when searching for low-rank structure. For this example we’ll use as genetic-controls a list of genes that are already well-recognized to affect colon-cancer (see the discussion towards the end of sec 1.3). To recap: we generate our set of control-genes by first choosing three of the most well documented genes which influence colon-cancer: MLH1, MSH2 and MSH6. Our second step involves using ‘Seek’ (see [M42]) to generate a list of genes that are commonly co-expressed alongside these three cancer-related genes. We then define our control-genes to be all the genes in this list which have a combined co-expression value (with MLH1, MSH2 and MSH6) of at least 0.75. The control-gene set generated this way comprises 1659 of our original 17941 genes. As we alluded to above, this control-gene set is *strongly* enriched for DNA-replication, DNA-repair, mitosis, etc. (e.g., p-values on the order of $1e-20$ or below); these are the pathways we want to ignore. We’ll classify the remaining 16282 genes as ‘case-genes’, and search for a bicluster within these case-genes that does not exhibit a low-rank structure that is shared by a significant fraction of the control-genes.

We can divide our measurements into the following data-matrices:

- the $M_D \times N_D$ case-matrix D : The entry $D_{j,k}$ is the k^{th} case-gene measurement from the j^{th} -case patient.
- the $M_X \times N_D$ control-matrix X : The entry $X_{j,k}$ is the k^{th} case-gene measurement from the j^{th} -control patient.
- the $M_D \times N_Y$ control-matrix Y : The entry $Y_{j,k}$ is the k^{th} control-gene measurement from the j^{th} -case patient.
- the $M_X \times N_Y$ control-matrix W : The entry $W_{j,k}$ is the k^{th} control-gene measurement from the j^{th} -control patient.

Our goal will be to find co-expressed subsets of genes within the case-population; we would like to find a low-rank bicluster within D . As before, we are most interested in biclusters which are case-patient-specific and don't extend to include many control-patients from X . In addition, we would also like to focus on biclusters which are case-gene-specific, and don't extend to include many gene-expression measurements from the control-genes that are commonly co-expressed with MLH1, MSH2 and MSH6 (i.e., Y). Put another way, we are interested in finding co-expressed subsets of genes that are (as best as possible) isolated to the case-patient-population and restricted to the 'undiscovered' subset of case-genes.

In order to accomplish this goal, we can extend the techniques used above, counting loops as before. This time, however, we'll also count loops that go through D and Y , as well as loops that go through all four data-matrices D , X , W and Y . Similar to the situation with both cases and controls (see section 8), we'll flip the sign on loops passing through X , and also flip the sign of loops passing through Y . Consequently, we'll end up positively counting loops passing through W . One way of understanding this choice of signs is that (i) we want to reward low-rank loops that are fully contained in D , and (ii) penalize low-rank loops that extend from D to X or from D to Y . The reason we want to reward low-rank loops that pass through all four data-matrices (i.e., through W) is that we want to ensure that biclusters are not unfairly penalized when the data-set has ambient correlations. Put another way, we would like the average row- and column-scores of random data (with no planted biclusters) to be 0. We choose the sign associated with the W -loops so that this average score remains 0 even when the random data is drawn from a non-isotropic gaussian (rather than an isotropic gaussian).

The row-scores built using this intuition take the following form (see Fig 69):

$$\begin{aligned} [Z_{\text{ROW}}^{DDDD}]_j &= \widetilde{\sum}_{\substack{j \text{ fixed, } j' \in D, j' \neq j; \\ k' \neq k}} D_{jk} D_{j'k} D_{j'k'} D_{jk'}, & [Z_{\text{ROW}}^{DXXXD}]_j &= \widetilde{\sum}_{\substack{j \text{ fixed, } j' \in X; \\ k' \neq k}} D_{jk} X_{j'k} X_{j'k'} D_{jk'} \\ [Z_{\text{ROW}}^{DDYY}]_j &= \widetilde{\sum}_{\substack{j \text{ fixed, } j' \in D, j' \neq j; \\ k \in D, k' \in Y}} D_{jk} D_{j'k} Y_{j'k'} Y_{jk'}, & [Z_{\text{ROW}}^{DXWY}]_j &= \widetilde{\sum}_{\substack{j \text{ fixed, } j' \in X; \\ k \in D, k' \in Y}} D_{jk} X_{j'k} W_{j'k'} Y_{jk'}, \end{aligned}$$

and are combined as follows:

$$[Z_{\text{ROW}}]_j = [Z_{\text{ROW}}^{DDDD}]_j - [Z_{\text{ROW}}^{DXXXD}]_j - [Z_{\text{ROW}}^{DDYY}]_j + [Z_{\text{ROW}}^{DXWY}]_j.$$

The column-scores, on the other hand, take the following form:

$$\begin{aligned} [Z_{\text{COL}}^{DDDD}]_k &= \widetilde{\sum}_{\substack{j', j \in D, j' \neq j; \\ k \text{ fixed, } k' \neq k}} D_{jk} D_{j'k} D_{j'k'} D_{jk'}, & [Z_{\text{COL}}^{DDXX}]_k &= \widetilde{\sum}_{\substack{j \in D, j' \in X; \\ k \text{ fixed, } k' \neq k}} D_{jk} D_{j'k} X_{j'k'} X_{j'k}. \\ [Z_{\text{COL}}^{DYYD}]_k &= \widetilde{\sum}_{\substack{j', j \in D, j' \neq j; \\ k \in D \text{ fixed, } k' \in Y}} D_{jk} Y_{j'k'} Y_{j'k} D_{j'k}, & [Z_{\text{COL}}^{DYWX}]_k &= \widetilde{\sum}_{\substack{j \in D, j' \in X; \\ k \in D \text{ fixed, } k' \in Y}} D_{jk} Y_{j'k'} W_{j'k'} X_{j'k}. \end{aligned}$$

and are combined as follows:

$$[Z_{\text{COL}}]_k = [Z_{\text{COL}}^{DDDD}]_k - [Z_{\text{COL}}^{DDXX}]_k - [Z_{\text{COL}}^{DYYD}]_k + [Z_{\text{COL}}^{DYWX}]_k.$$

A simple example illustrating this scenario is shown in Figs 70 and 71, where we consider a variation on the planted-bicluster problem involving both a planted case-specific bicluster and various planted non-specific biclusters involving both the patient- and genetic-controls.

When we actually carry out this method on the data from section 1.3 we do indeed find a significant bicluster, shown in Figs 72-75. Not only is this bicluster significant with respect to the label-shuffled hypothesis, but it is also enriched strongly for many different pathways. The most strongly enriched pathways include substrate-specific channel-activity (p=3e-8), passive-transmembrane-transporter-activity (p=1e-7) and cation-transport (p=1.5e-6); all of which are distinct from the most prominent pathways affiliated with the control-genes (as desired).

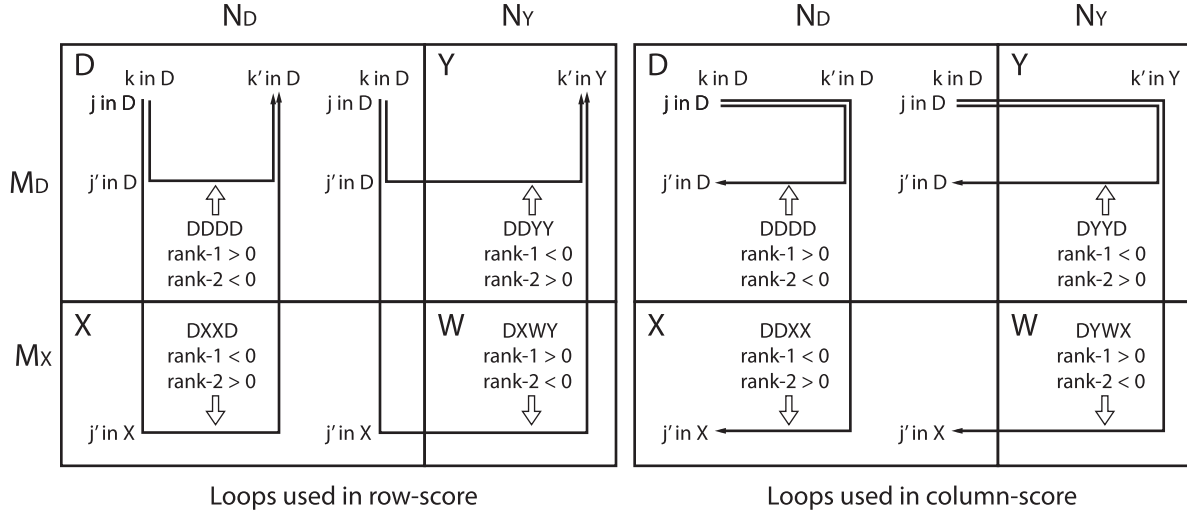


Figure 69: Illustration of the different kinds of loops that contribute to the row-score in the presence of genetic-controls. In the left panel we show some loops which contribute to the row-score; on the right we show some loops which contribute to the column-score. In each panel we illustrate an $M_D \times N_D$ case-matrix D (upper left) and an $M_X \times N_D$ control-matrix X (bottom left). In addition, we illustrate an $M_D \times N_Y$ matrix of genetic-controls Y (upper right) and an $M_X \times N_Y$ matrix of genetic-controls for the control-patients W (bottom right). Some of the loops are contained entirely within D , some extend from D to X , some extend from D to Y , and some extend to include D , X , W and Y . We count the first and last type of loop positively if they are rank-1, and negatively if they are rank-2. We count the second and third type of loop positively if they are rank-2, and negatively if they are rank-1.

15.2 Searching for ‘rank-0’ (i.e., differentially-expressed) biclusters:

As mentioned above, our algorithms accumulate the loops within D . While conceptually straightforward, the accumulation of these loops requires a matrix-matrix multiplication, limiting the efficiency of our approach (i.e., requiring at least $O(\max(M, N)MN)$ work).

If, instead of attempting to find generic rank-1 biclusters, we restrict our search to a particular kind of rank-1 bicluster, we can overcome this limitation and make our algorithms even more efficient (e.g., requiring only $O(MN)$ work). Specifically, we can greatly reduce the run-time of our algorithm if we decide to search only for special rank-1 biclusters $B \sim 1 \cdot v^T$; i.e., where the column-vector ‘ u ’ has been replaced by a vector of all-ones. These biclusters are characterized as follows: each column of the bicluster is either high (i.e., mostly above the median for that column) or low (i.e., mostly below the median for that column). This means that, after binarization, the ‘high’ columns of the bicluster are mostly ‘+1’, and the ‘low’ columns are mostly ‘-1’. Put another way, the rows/columns of the bicluster clump together near a single point in high-dimensional space; the bicluster is effectively rank-0. These biclusters can also be thought of as ‘differentially-expressed’: the genes in the biclusters are each differentially-expressed when comparing the patients in the bicluster to the rest of the population.

To detect these rank-0 biclusters we don’t actually need to consider full loops; simpler half-loops will suffice (see Fig 76). The accumulation of half-loops is substantially easier than the accumulation of full loops, and does not require a matrix-matrix multiplication. Consequently, our methodology can be applied to the case of rank-0 biclusters with a reduced computational cost of only $O(MN \log(\max(M, N)))$ work.

All of our corrections involving sparsity, categorical- and continuous-covariates work just as well here. The formulae in section 12 can be carried forward, replacing all the loops with the associated half-loops. Specifically, we redefine our base-scores as follows:

$$\left[Z_{\text{ROW}}^{Q, Q'; i, i'; \text{base}} \right]_j = \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ fixed in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i' \\ k \in Q}} [Qi - 1_{Qi} \alpha^T]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'} \alpha^T]_{j',k},$$

$$\left[Z_{\text{COL}}^{Q, Q'; i, i'; \text{base}} \right]_k = \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i' \\ k \text{ fixed,}}} [Qi - 1_{Qi} \alpha^T]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'} \alpha^T]_{j',k},$$

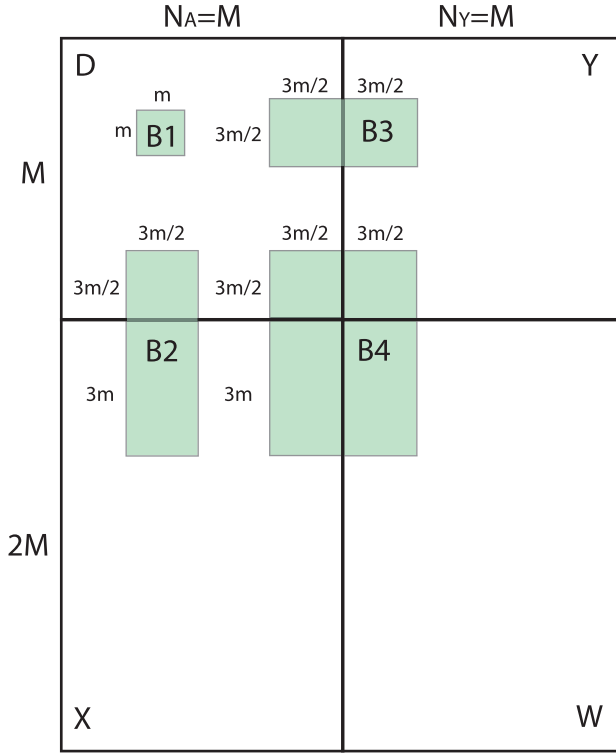


Figure 70: This figure illustrates the setup for the numerical experiments shown in Fig 71. These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving both case- and control-patients, along with case- and control-genes. The case-genes associated with the case- and control-patients are stored in matrices D and X , respectively. The control-genes associated with the case- and control-patients are stored in matrices Y and W , respectively. For each numerical experiment we let D be a large $M \times M$ random matrix with entries chosen independently from a distribution with median 0. We construct X , Y , and W in a similar fashion, except that X , Y and W are $2M \times M$, $M \times M$ and $2M \times M$, respectively. For each experiment we implant within the case-matrix D a small $m \times m$ bicluster $B1$ that is rank-1 with error ε . The rows and columns of $B1$ are chosen at random. We also implant a second rank-1 error- ε ‘red-herring’ bicluster $B2$ within both the case-matrix D as well as the control-matrix X . To implant $B2$ we first randomly choose $3m/2$ case-genes K and $3m/2$ case-patients J^D from D , as well as $3m$ control-patients J^X within X . We then construct a $3 \cdot 3m/2 \times 3m/2$ bicluster $B2$ that is rank-1 with error ε , and implant the top third of $B2$ into the chosen case-patients J^D and the bottom two-thirds of $B2$ into the chosen controls J^X (in each case using the chosen case-genes K). We implant another bicluster $B3$ in a similar fashion, except that $B3$ extends across both D and Y , and the total size of $B3$ is $3m/2 \times 2 \cdot 3m/2$. To implant $B3$ we first randomly choose $3m/2$ case-genes K_D from D and another $3m/2$ control-genes K_Y from Y , as well as $3m/2$ case-patients J from D . After constructing the rank-1 error ε matrix $B3$, we implant the left half of $B3$ into the chosen case-patients J and case-genes K_D , while implanting the right half of $B3$ into the chosen case-patients J and control-genes K_Y . Finally, we implant a fourth bicluster $B4$ which extends across D , X , W and Y . The bicluster $B4$ is of total size $3 \cdot 3m/2 \times 2 \cdot 3m/2$, and we create and implant it in a manner analogous to biclusters $B2$ and $B3$ above. We remark that our illustration is somewhat misleading, as the biclusters $B1$, $B2$, $B3$, and $B4$ are not typically disjoint. Because the row- and column-subsets for these biclusters are chosen randomly, there is typically a small overlap between them. For the purposes of this numerical experiment we implant each bicluster in the order described above: any overlapping entries will be overwritten by the next bicluster in the list (with biclusters $B2$, $B3$, $B4$ taking precedence). For each numerical experiment we calculate the average auc A_1 for $B1$, A_2 for $B2$, and so forth. Our goal is to find $B1$, despite the masking effect of the larger biclusters.

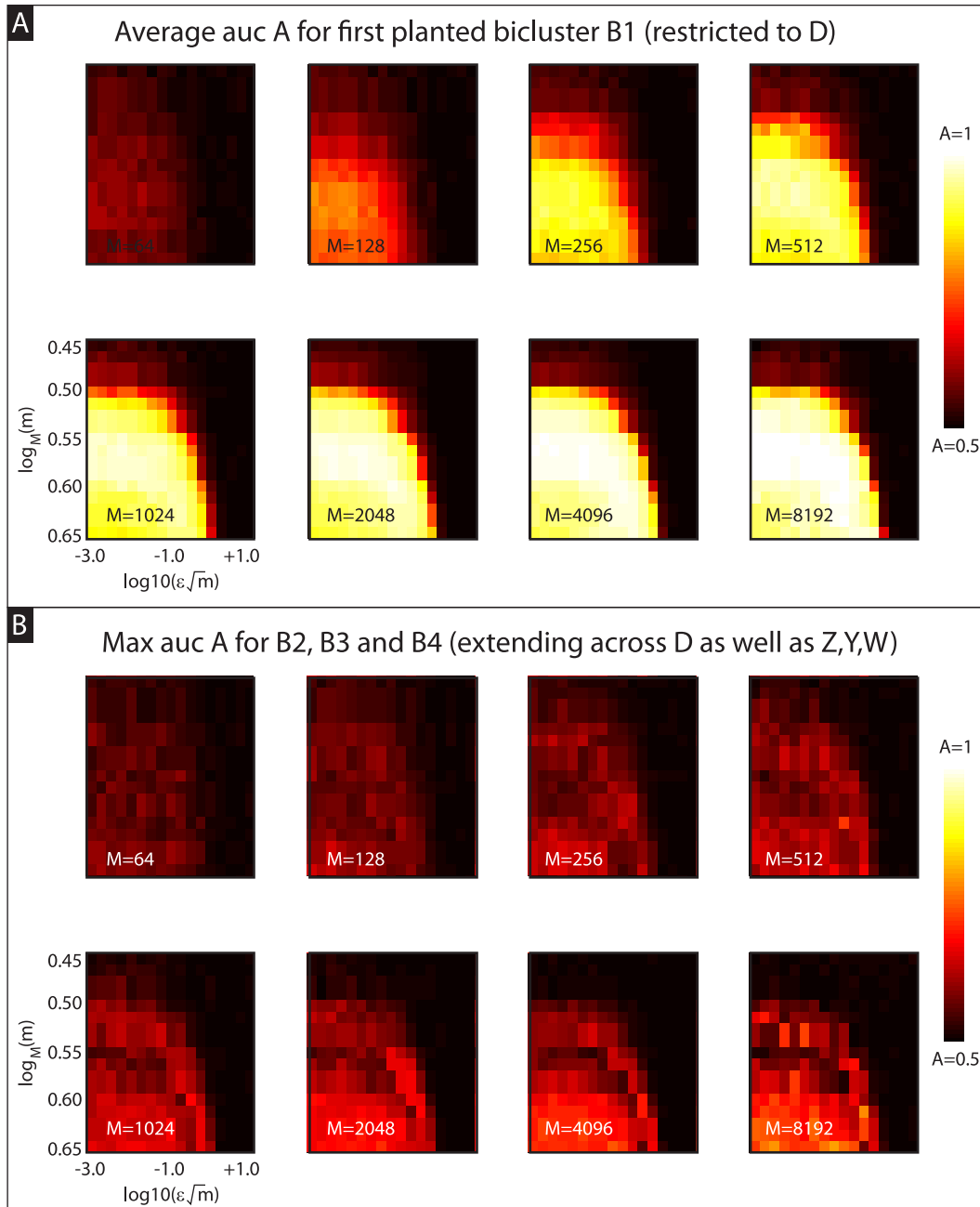


Figure 71: These numerical experiments illustrate the performance of our algorithm on a simple planted-bicluster problem involving both case- and control-patients as well as case- and control-genes (see Fig 70). In panel-A we show the trial-averaged value of A_1 associated with the case-specific bicluster $B1$ as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. In panel-B we show the maximum of the trial-averaged values of A_2 , A_3 and A_4 associated with the other biclusters. While the presence of genetic-controls makes it more difficult for our algorithm to detect $B1$ (e.g., compare panel-A with Fig 42), our algorithm is still generally successful when $\log_{10}(\epsilon\sqrt{m}) \lesssim 0$ and $\log_M(m) \gtrsim 0.5$.

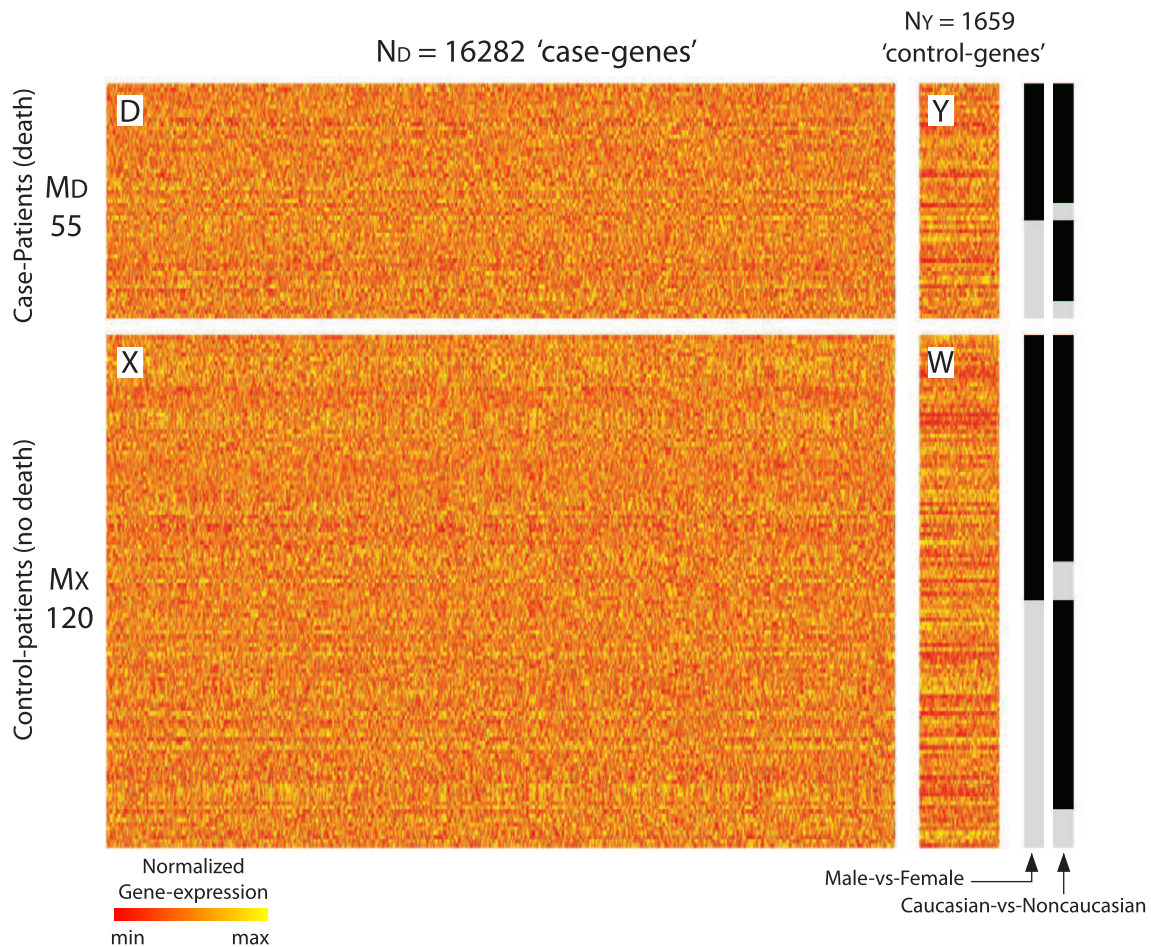


Figure 72: This figure illustrates the GSE17536 gene-expression data-set used in section 15.1. Each row corresponds to a patient, and each column to a ‘gene’ (i.e., gene-expression measurement): the color of each pixel codes for the intensity of a particular measurement of a particular patient (see colorbar to the bottom). We divide the 175 patients into $M_D = 55$ case-patients and $M_X = 120$ control-patients; colorectal cancer was determined to be the significant cause of death for the former, but not the latter. We also divide the 17941 genes into $N_D = 16286$ case-genes and $N_Y = 1659$ control-genes; the control-genes are those that are significantly co-expressed alongside genes MLH1 MSH2 and MSH6, whereas the case-genes are the remainder. We use these patient- and gene-classifications to delineate the case-matrix D , as well as the control-matrices X , Y and W . The covariates (gender and ethnicity) for each patient are shown to the far right, (i.e., grey vs black).

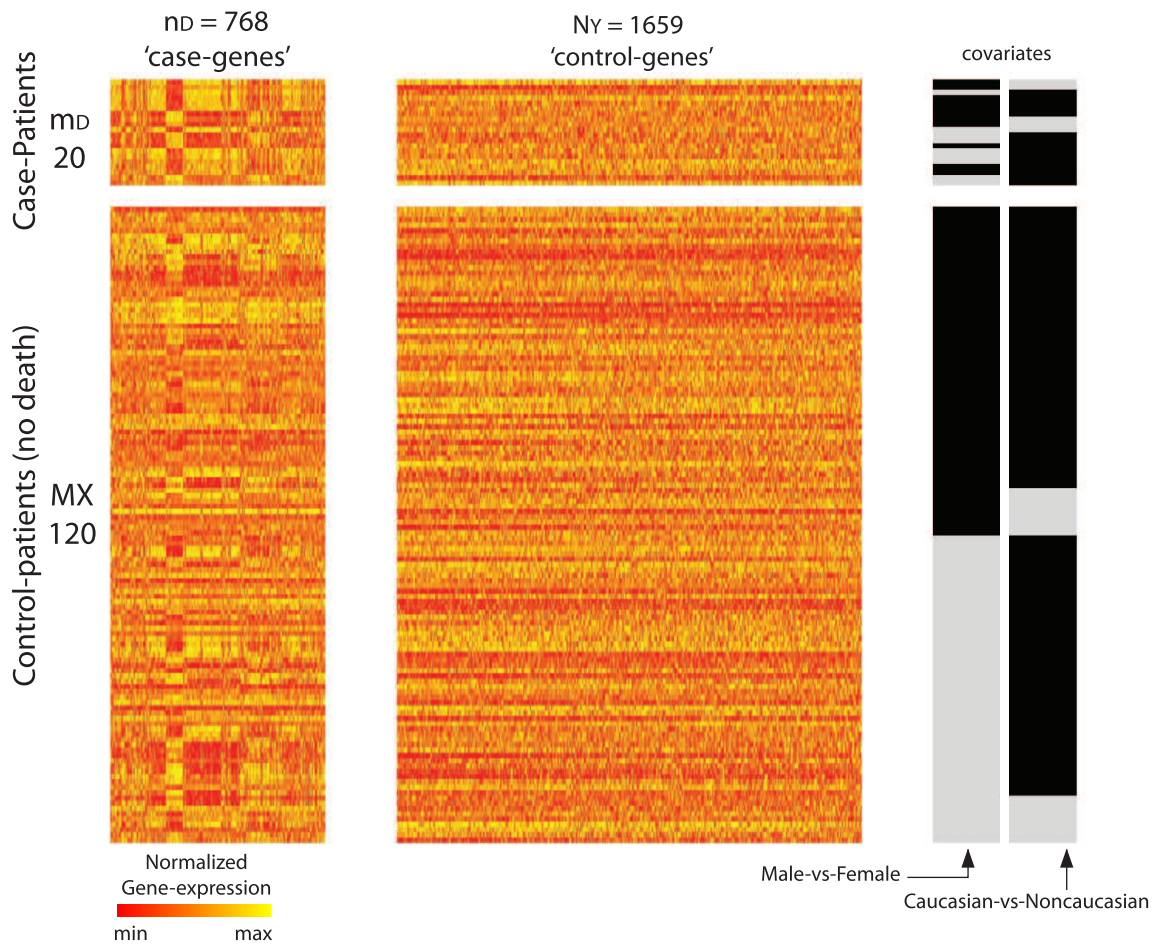


Figure 73: This figure illustrates the bicluster discovered after applying the loop-counting algorithm described in section 15.1 to the data-set shown in Fig 72. We've shown only the m_D case-patients and n_D genes involved in the bicluster, listed in the order given by the output of the algorithm (the patients and genes placed in the upper-left corner are those that have been retained the longest).

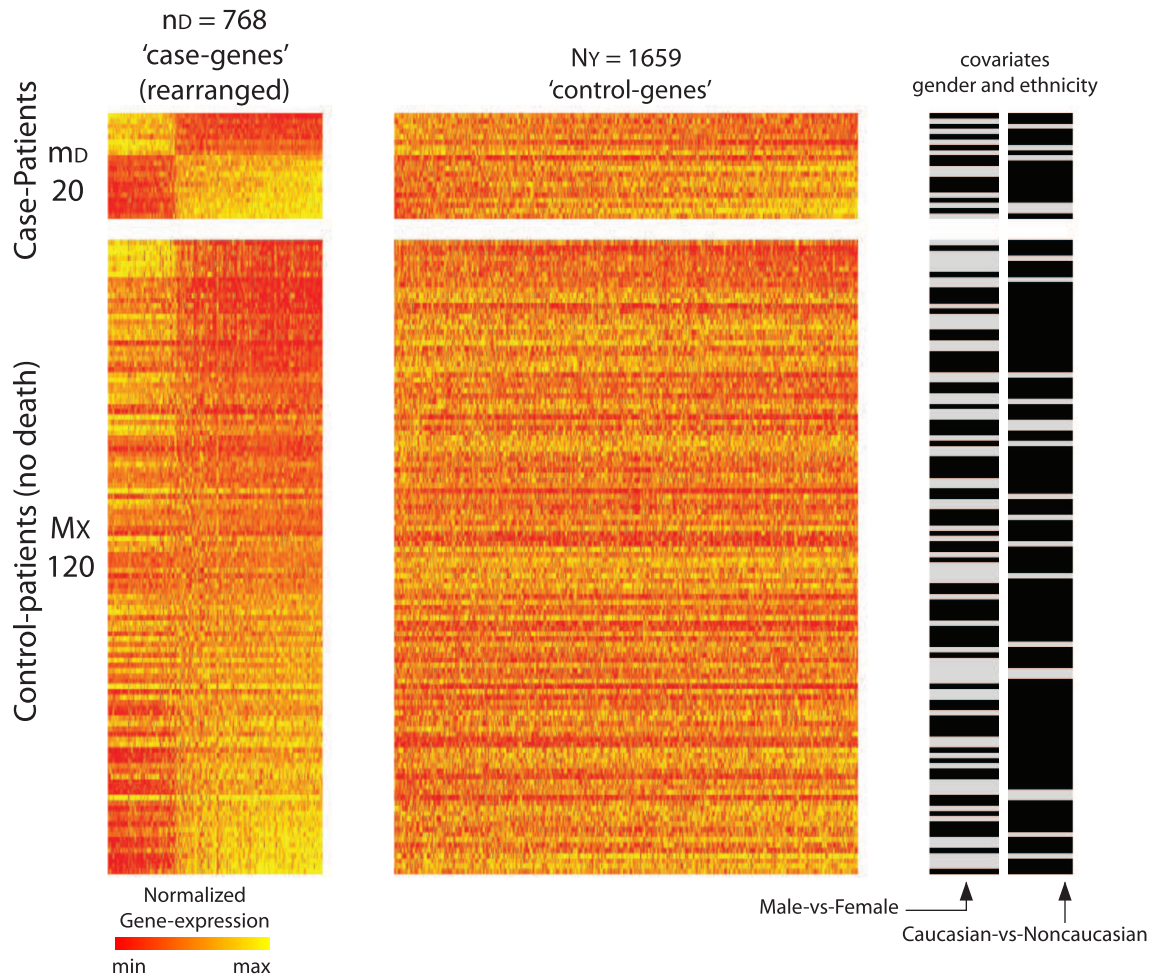


Figure 74: This figure illustrates the same bicluster shown in Fig 73, except that we've rearranged the rows and columns to highlight the co-expression pattern more clearly. The control-patients (and control-genes) are also rearranged in accordance with the dominant right (and left, respectively) singular vectors of the bicluster. As can be seen, there is a striking pattern of correlation across the 20 case-patients and 768 case-genes within the bicluster. Moreover, this co-expression pattern is not exhibited across a comparable fraction of the control-patients or control-genes.

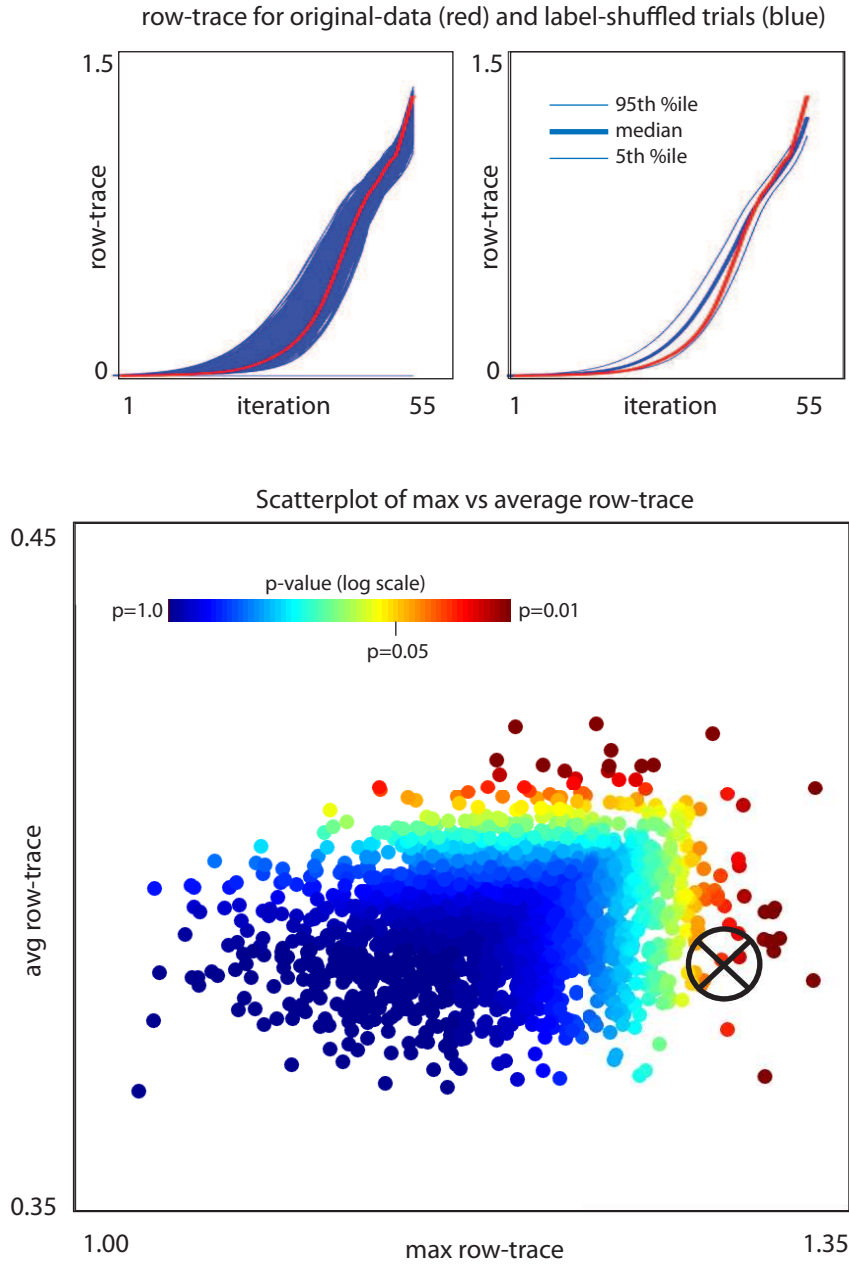


Figure 75: On top we show row-traces associated with the bicluster shown in Fig 74. On the left we show the row-trace as a function of iteration for the original-data (red) as well as each of the 2048 random shuffles (blue) generated under the null hypothesis. On the right we replot this same trace data, showing the 5th, 50th and 95th percentile (across iterations) of the shuffled-distribution. On the bottom we show a scatterplot of maximum- and average-row-traces (taken across the iterations) for both the original data and the shuffled-trials. Each row-trace from the graph above is plotted as a single point in 2-dimensional space; the horizontal-axis corresponds to the maximum row-trace and the vertical-axis corresponds to the average row-trace (taken across the iterations). The original-data is indicated with a ‘ \otimes ’, and each of the random shuffles with a colored ‘ \bullet ’. The color of each ‘ \bullet ’ indicates the p-value associated with that position in x,y -space. By comparing the original-trace with the shuffled-distribution we can read off a p-value of 0.021 for the bicluster drawn from the original-data.

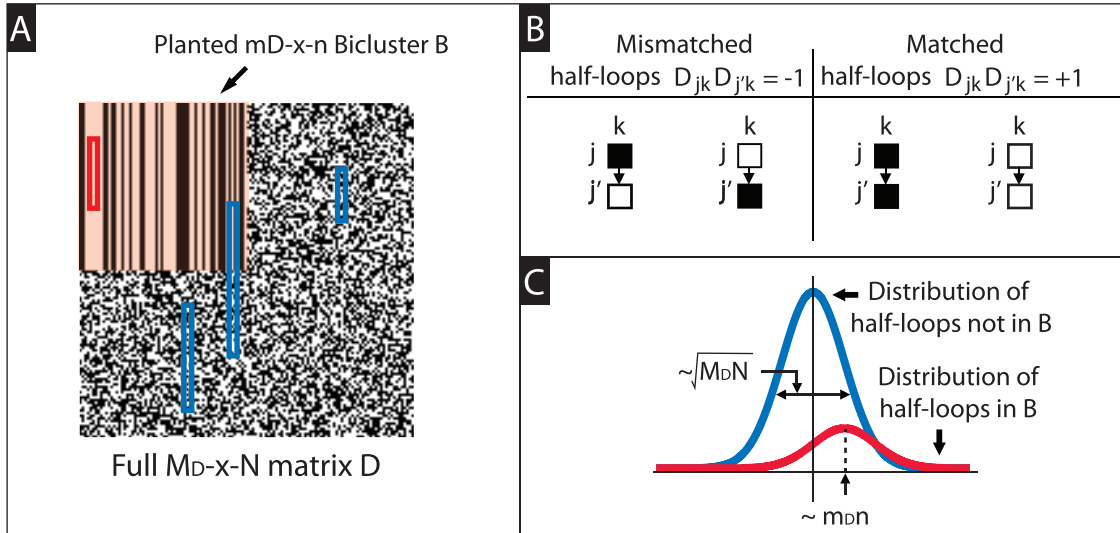


Figure 76: Detecting a rank-0 bicluster within a case-matrix alone. In Panel-A we show a large $M \times N$ binarized matrix D (black and white pixels correspond to values of ± 1 , respectively). In the upper left corner of D we've inserted a large rank-0 bicluster B (shaded in pink). Our algorithm considers all 2×1 submatrices (i.e., 'half-loops') within D . Several such half-loops are highlighted via the thin blue rectangles (the top and bottom of each rectangle pick out a 2×1 submatrix). Generally speaking, the entries of a half-loop are equally likely to be matched or mismatched. Some half-loops, such as the half-loop shown in red, are entirely contained within B . These half-loops are more likely to be matched than mismatched. In Panel-B we show some examples of mismatched and matched half-loops. Given a half-loop with row-indices j, j' and column-index k , the parity of the half-loop is determined by the sign of $D_{jk}D_{j'k}$. Our algorithm accumulates a 'half-loop-score' for each row j and each column k . The half-loop-score for a particular row j is given by $\sum_{j',k} D_{jk}D_{j'k}^T = [DD^T \mathbf{1}]_j$. The half-loop-score for a column k is given by $[\mathbf{1}^T D]_k \cdot [D^T \mathbf{1}]_k$. In Panel-C we show the distribution of half-loop-scores we might expect from the rows within D . The blue-curve corresponds to the distribution of scores expected from the rows/cols of D that are not in B , whereas the red-curve corresponds to the distribution of scores expected from the rows/cols of B .

and redefine the (2-sided) covariate-dependent sub-scores for each $t = 1, \dots, N_T$ as follows:

$$\left[Z_{\text{ROW}}^{Q,Q'; i,i'; [t]} \right]_j = \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ fixed in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i'; \\ k \in Q}} [T_{Qi}]_{j,t} [Qi - 1_{Qi} \alpha^\top]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k} [T_{Q'i'}]_{j',t},$$

$$\left[Z_{\text{COL}}^{Q,Q'; i,i'; [t]} \right]_k = \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i'; \\ k \text{ fixed,}}} [T_{Qi}]_{j,t} [Qi - 1_{Qi} \alpha^\top]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k} [T_{Q'i'}]_{j',t}.$$

After substituting these new definitions, the subsequent combination of sub-scores remains unchanged.

We pause to remark that we can also redefine our 1-sided covariate-dependent sub-scores as follows:

$$\left[Z_{\text{ROW}}^{Q,Q'; i,i'; [t] \text{ 1s}} \right]_j = \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ fixed in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i'; \\ k \in Q}} [Qi - 1_{Qi} \alpha^\top]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k} [T_{Q'i'}]_{j',t},$$

$$\left[Z_{\text{COL}}^{Q,Q'; i,i'; [t] \text{ 1s}} \right]_k = \widetilde{\sum}_{\substack{Q, i, Q', i' \text{ fixed}; \\ j \text{ in } Qi, j' \text{ in } Q'i'; \\ j' \neq j \text{ if } Q=Q' \text{ and } i=i'; \\ k \text{ fixed,}}} [Qi - 1_{Qi} \alpha^\top]_{j,k} \Delta_{kk} [Q'i' - 1_{Q'i'} \alpha^\top]_{j',k} [T_{Q'i'}]_{j',t}.$$

After this redefinition, we use the same subsequent combination of sub-scores as discussed in section 12 (i.e., the combination appropriate for a 1-sided covariate-correction).

One advantage to the half-loop algorithms described above is that they are very easy to analyze. In fact, much of our previous analysis remains unchanged; the main difference being that we replace the probability $g_{l,\varepsilon,m}$ with the new probability \hat{g}_ε . Recall that we previously defined the former to be the probability that a loop drawn from a low-rank submatrix is rank-1, whereas we now define the latter to be the probability that a half-loop drawn from a rank-0 submatrix is of matched sign. While $g_{l,\varepsilon,m}$ depends on both the noise-level ε as well as the submatrix size m , the probability \hat{g}_ε only depends on the level of noise, and not on the size m . To generate a rank-0 submatrix with noise level ε , we draw each entry independently and randomly to be either ± 1 with a probability chosen so that $g_{1,\varepsilon,m}$ equals \hat{g}_ε .

A second advantage to the half-loop algorithms is that they can be used to search for differentially-expressed biclusters where each gene is either high across the cases and low across the controls, or vice-versa. These structures are – by definition – rank-1 structures that extend across both the case- and control-populations; they were deliberately ignored by the control-correction described in section 8, but can easily be found using the half-loop algorithms above.

A simple example illustrating this approach is shown in Figs 77 and 78. A more complicated example is shown in Figs 79 and 80. These two examples are structured almost identically to the numerical experiments described in Figs 24 and 54, the only difference being that we implant rank-0 biclusters, rather than merely low-rank biclusters. As these examples illustrate, our half-loop algorithm is not quite as good as our full loop-counting algorithm. Nevertheless, it is an order of magnitude faster, a feature that may be of interest when analyzing large data-sets.

When we actually carry out our half-loop method on the data from section 1.4, we do indeed find a rank-0 bicluster, shown in Figs 81-85. This rank-0 bicluster is strongly enriched for many different pathways, the most prominent of which include: neurogenesis ($p=1.6e-9$), neuron-differentiation ($p=5.2e-9$), axon-guidance ($p=3.4e-6$), enzyme-linked receptor-protein signaling pathway ($p=6e-5$), ion-channel-binding ($1.6e-4$), peptidyl-tyrosine phosphorylation ($5e-4$) and ephrin-receptor-activity ($8e-4$). As in the example shown in section 1.4, this bicluster was found within the neurotypical patients, and may highlight genes that play a protective role with regards to the psychiatric disorder associated with this data-set.

15.3 Searching for triclusters:

There are often situations where the data-paradigm we've assumed – involving N measurements taken across M patients – doesn't suffice. For example, in a clinical study there may be M patients, each of which are subjected to P different kinds of treatments (e.g., therapy regimes, medications, etc). For each of these P treatments, N variables may be measured for each patient. Within this paradigm, the data isn't best represented as a 2-dimensional array (e.g., a matrix), but rather as a 3-dimensional array (i.e., a box or a cube) comprising both rows and columns, as well as 'layers'. In formal terms, we

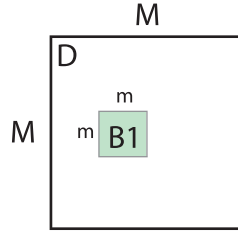


Figure 77: This figure illustrates the setup for the numerical experiments shown in Fig 78. The setup for these experiments is almost identical to the setup shown in Fig 24, with the exception that the planted bicluster is not merely low-rank, but rather rank-0. A rank-0 bicluster is characterized by its size, as well as by the probability \hat{g}_ε that a half-loop drawn from the bicluster will be of matched sign. We generate our rank-0 biclusters by choosing each entry independently and randomly so that the probability \hat{g}_ε is equal to the $g_{1,\varepsilon,m}$ we would associate with a rank-1 bicluster.

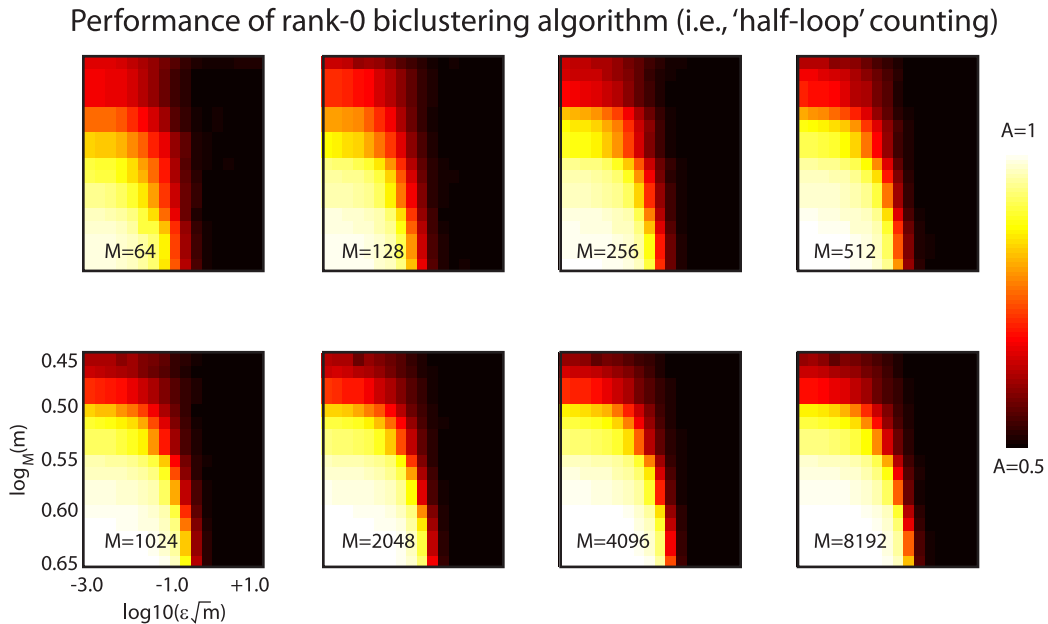


Figure 78: This figure illustrates the numerical experiments described in Fig 77. We show the trial-averaged value of A as a function of $\varepsilon\sqrt{m}$ and $\log_M(m)$. Note that, while our half-loop algorithm does indeed detect the bicluster when $\varepsilon\sqrt{m} \lesssim 1/10$ and $m \gtrsim \sqrt{M}$, it does not do as good a job as our full loop-counting algorithm (compare with Fig 26).

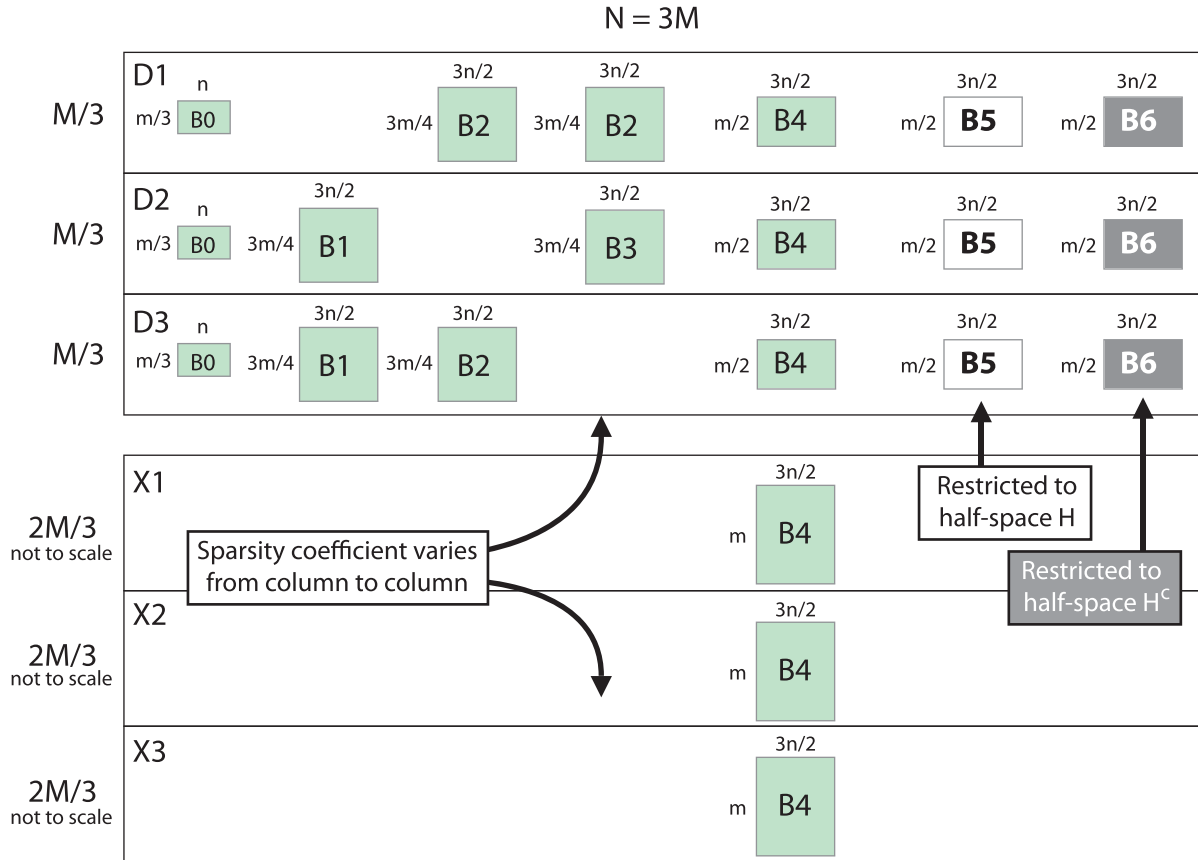


Figure 79: This figure illustrates the setup for the numerical experiments shown in Fig 80. The setup for these experiments is almost identical to the setup shown in Fig 54, with the exception that all the planted biclusters are not merely low-rank, but rather rank-0. As in Fig 78, we generate our rank-0 biclusters by choosing each entry independently and randomly so that the probability \hat{g}_ε is equal to the $g_{1,\varepsilon,m}$ we would associate with a rank-1 biclster.

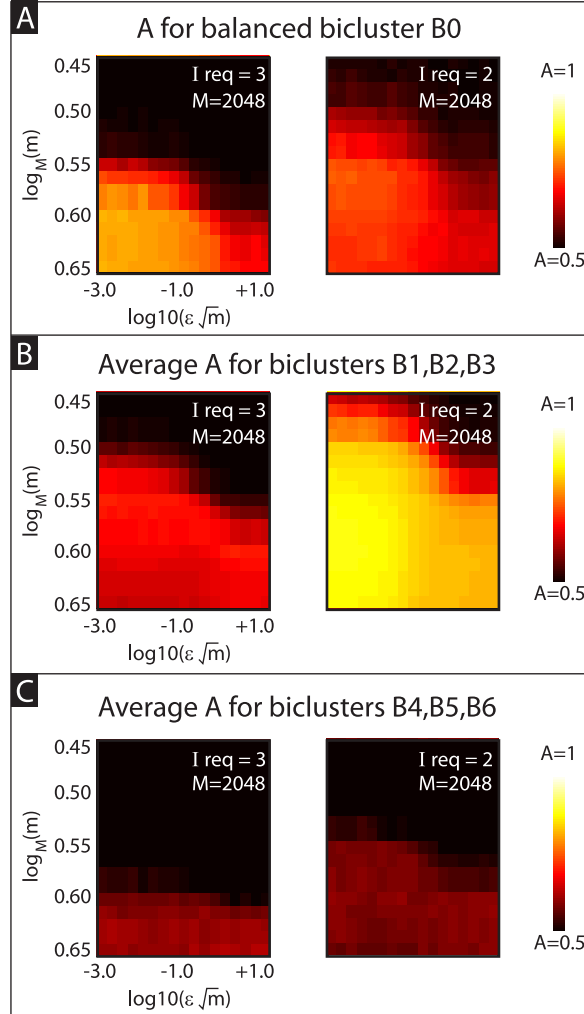


Figure 80: This figure illustrates the numerical experiments described in Fig 79. In panel-A we show the trial-averaged value of A_0 as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$. In panel-B we show the trial-averaged value of A_1, A_2, A_3 . In panel-C we show the trial-averaged value of A_4, A_5, A_6 . The left side of each panel displays results with $I_{\text{req}} = 3$, the right side with $I_{\text{req}} = 2$. Note that, when $I_{\text{req}} = 3$ the goal was to find B_0 , as this was the only case-specific bicluster that was balanced across all the continuous-covariates and covariate-categories. On the other hand, when $I_{\text{req}} = 2$ the goal was to find the most dominant of B_0, B_1, B_2 and B_3 , as each of these case-specific biclusters were balanced across all the continuous-covariates while extending across at least 2 covariate-categories. Note that our half-loop algorithm does a modest job of achieving these goal, even in the presence of the other biclusters B_4, B_5 and B_6 . Note also that this half-loop algorithm is not quite as good as our original loop-counting algorithm (compare with Fig 55).

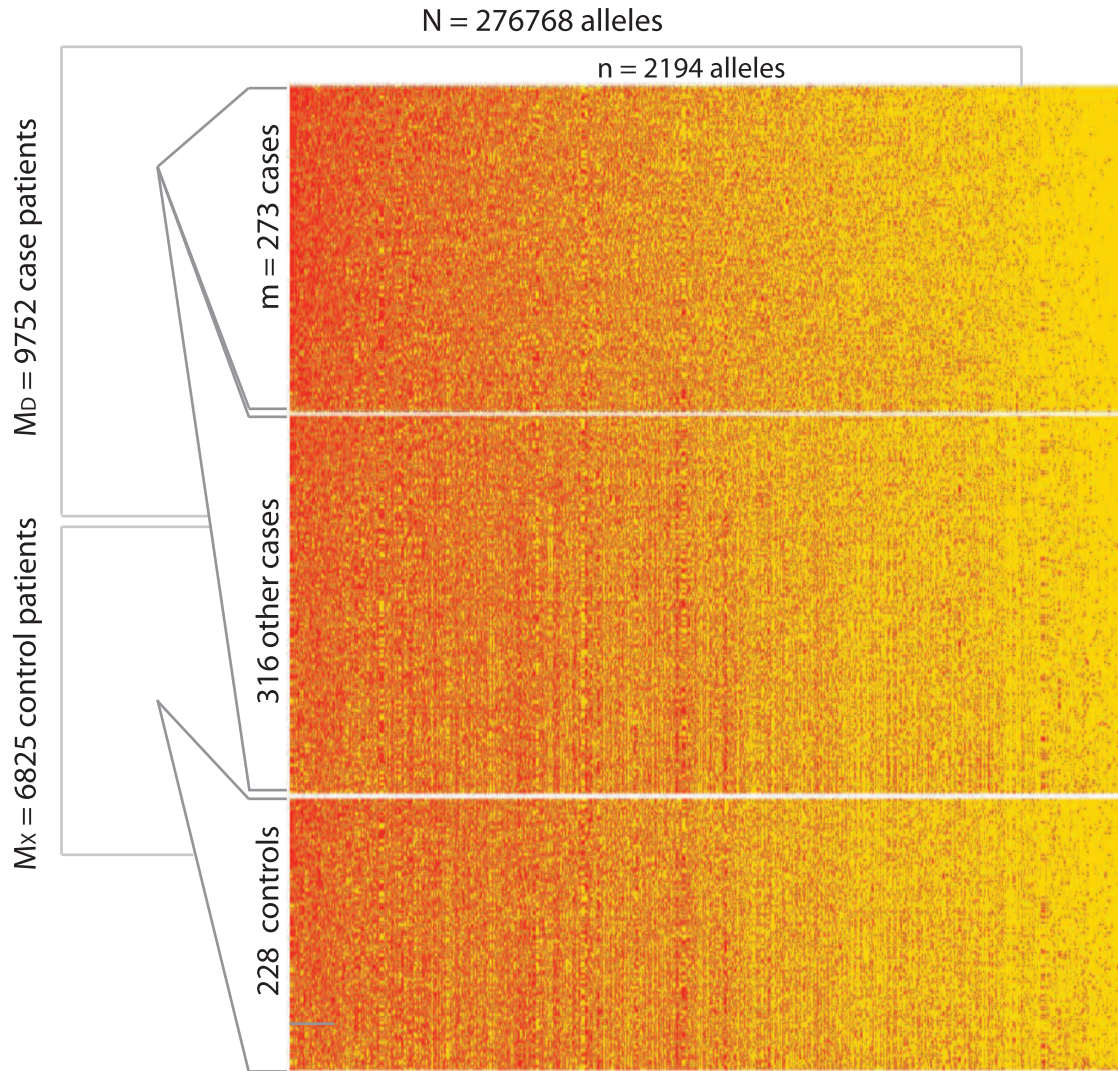


Figure 81: In this figure we illustrate the rank-0 bicluster found within the GWAS data-set discussed in Example-3 (see sections 1.4 and 15.2). This figure has the same format as Fig 11; the bicluster is shown on in the topmost portion of the inset, while a random selection of other case-patients and control-patients are shown below. While the distribution of genetic-data within this bicluster does not appear to be all that different from the other patients, the bicluster is indeed quite different from the remaining patients. See Fig 82 for more details.

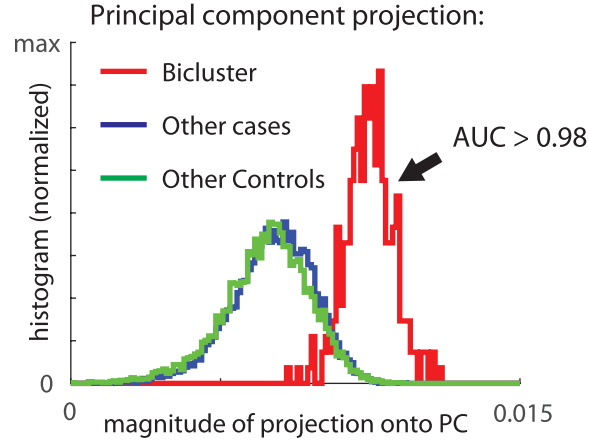


Figure 82: The bicluster shown in the top of Fig 81 was not obviously distinct from the remaining cases and controls. However, the structure within this bicluster can be revealed by first calculating the dominant right principal component of the bicluster, and then projecting each patient’s associated genetic information onto that principal component. In this figure we show histograms collecting the magnitude of this projection. While the absolute magnitude of the projection is not very large for any of the patients (which is why the bicluster was not visually distinct from the remaining patients), the actual values of the projection are quite distinct – the patients within the bicluster are indeed well-separated from the other patients. We can quantify this separation by calculating the Auc associated with the projected values, yielding in this case an Auc of $A > 0.98$.

imagine our data arranged into an array D of dimension $M \times N \times P$, where $D_{j,k,l}$ corresponds to the k^{th} measurement of the j^{th} -patient as they undergo therapy l .

Within this 3-dimensional array, it is often prudent to search for subsets of patients that exhibit some kind of simple structure across a subset of measurements as well as a subset of treatments. Such a ‘tricluster’ would correspond to a ‘sub-cube’ of the data, rather than simply a submatrix. The techniques we discussed earlier can readily be extended to search for these kinds of objects as well.

For exposition, let’s consider the ‘planted-tricluster’ problem. We’ll assume that the data-array D contains a hidden $m \times n \times p$ sub-array B involving patient-subset J_B , measurement-subset K_B , and treatment-subset L_B (i.e., J_B , K_B and L_B are unknown). The sub-array B will exhibit structure not exhibited by the rest of D , and our goal will be to locate the tricluster B (i.e., to find the row- column- and layer-subsets J_B , K_B and L_B).

We’ll assume that each entry of D that is not in B is drawn independently and randomly from, say, a median-0 distribution, but that the entries of B are drawn from a more structured distribution. For simplicity let’s assume B is formed from a collection of outer-products; one for each pair of array-dimensions:

$$B_{j,k,l} = u_j^1 v_k^1 + u_j^2 w_l^2 + v_k^3 w_l^3,$$

where $\vec{u}^1, \vec{u}^2 \in \mathbb{R}^m$, and $\vec{v}^1, \vec{v}^3 \in \mathbb{R}^n$, and $\vec{w}^2, \vec{w}^3 \in \mathbb{R}^p$. are each randomly chosen (but unknown) vectors that describe the low-rank structure within B .

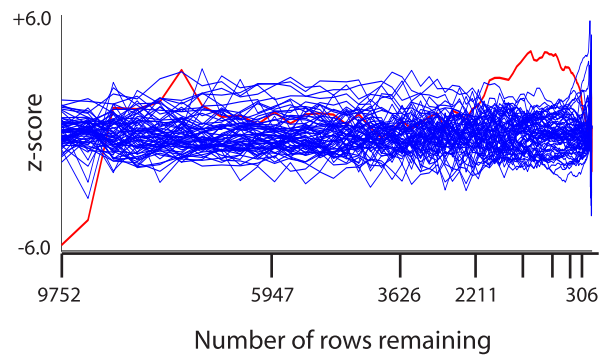
As before, we’ll binarize D , sending each entry to either +1 or –1, depending on its sign. Once we binarize D , we can calculate the following scores:

$$\begin{aligned} [Z_{\text{ROW}}]_j &= \sum_{\substack{j \text{ fixed, } j' \neq j; \\ k' \neq k; l}} \widetilde{D}_{jkl} \widetilde{D}_{j'kl} \widetilde{D}_{j'k'l} \widetilde{D}_{jk'l} + \sum_{\substack{j \text{ fixed, } j' \neq j; \\ k; l' \neq l}} \widetilde{D}_{jkl} \widetilde{D}_{j'kl} \widetilde{D}_{j'k'l'} \widetilde{D}_{jk'l'}, \\ [Z_{\text{COL}}]_k &= \sum_{\substack{k \text{ fixed, } k' \neq k; \\ j' \neq j; l}} \widetilde{D}_{jkl} \widetilde{D}_{jk'l} \widetilde{D}_{j'k'l} \widetilde{D}_{j'kl} + \sum_{\substack{k \text{ fixed, } k' \neq k; \\ j; l' \neq l}} \widetilde{D}_{jkl} \widetilde{D}_{jk'l} \widetilde{D}_{jk'l'} \widetilde{D}_{jk'l}, \\ [Z_{\text{LYR}}]_l &= \sum_{\substack{l \text{ fixed, } l' \neq l; \\ j' \neq j; k}} \widetilde{D}_{jkl} \widetilde{D}_{jk'l} \widetilde{D}_{j'k'l'} \widetilde{D}_{j'kl} + \sum_{\substack{l \text{ fixed, } l' \neq l; \\ j; k' \neq k}} \widetilde{D}_{jkl} \widetilde{D}_{jk'l'} \widetilde{D}_{jk'l} \widetilde{D}_{jk'l}, \end{aligned}$$

corresponding to the row-, column- and layer-scores, respectively. Once we’ve calculated these scores, we can remove the rows, columns and layers with low scores, and repeat the entire process. As before, this process will focus on the rows, columns and layers of B with high probability as long as $m \gtrsim \sqrt{M}$, $n \gtrsim \sqrt{N}$ and $p \gtrsim \sqrt{P}$.

The reason this process works is that – as before – the scores accumulate the ranks of the various loops within D . However, unlike the simpler situation discussed in section 2, D is a 3-dimensional array (and not merely a matrix).

z-score for row-trace for original-data (red) and H0x (blue)



Scatterplot of max vs average z-score

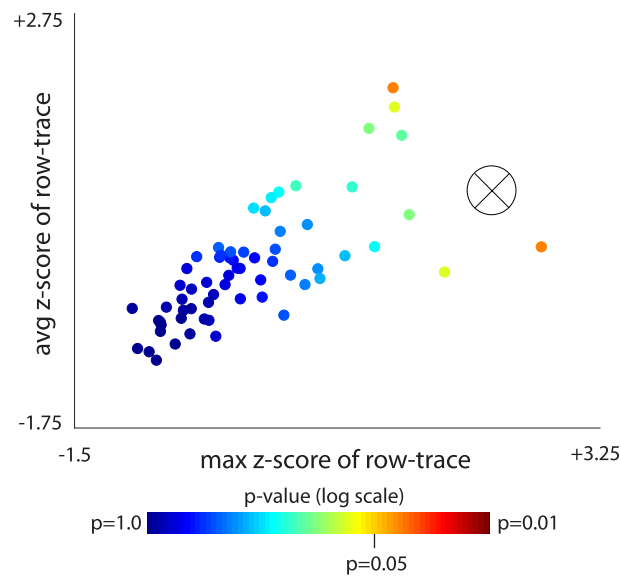
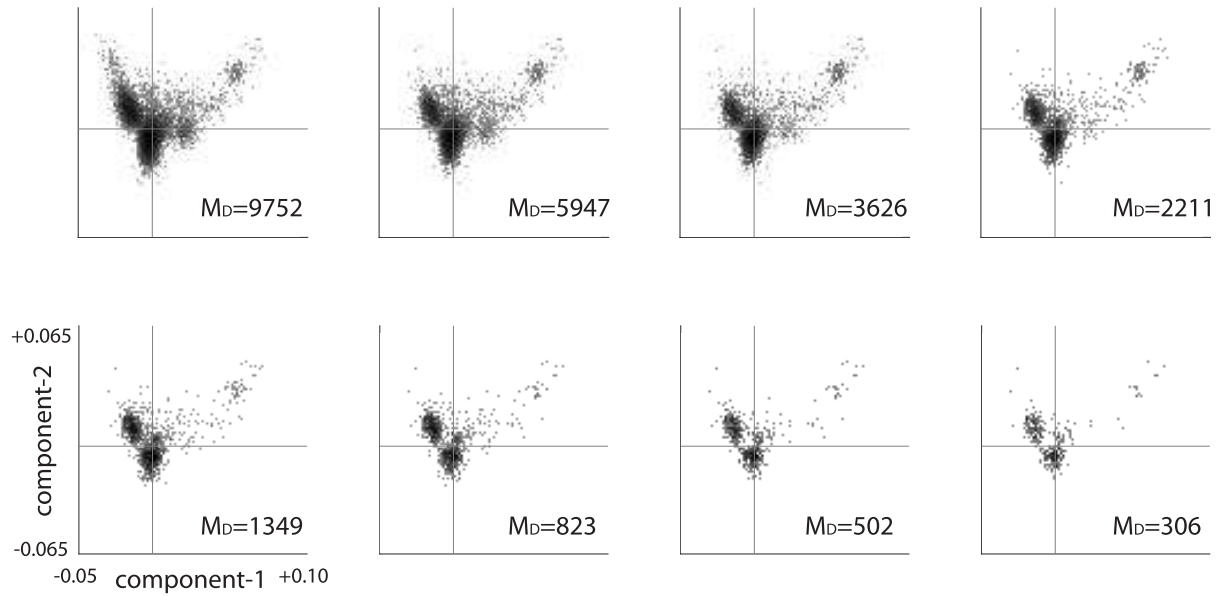


Figure 83: The bicluster shown in Fig 81 is significant with respect to our label-shuffled hypothesis H0x. This figure has the same format as Fig 63, and shows Z-scores for the bicluster shown in Fig 81. This bicluster is significant with respect to our label-shuffled hypothesis H0x, yielding a p-value of $\lesssim 0.03$.

Scatterplots of patients in covariate-space as algorithm proceeds: continuous-covariate correction



Density of patients in covariate-space as algorithm proceeds: continuous-covariate correction

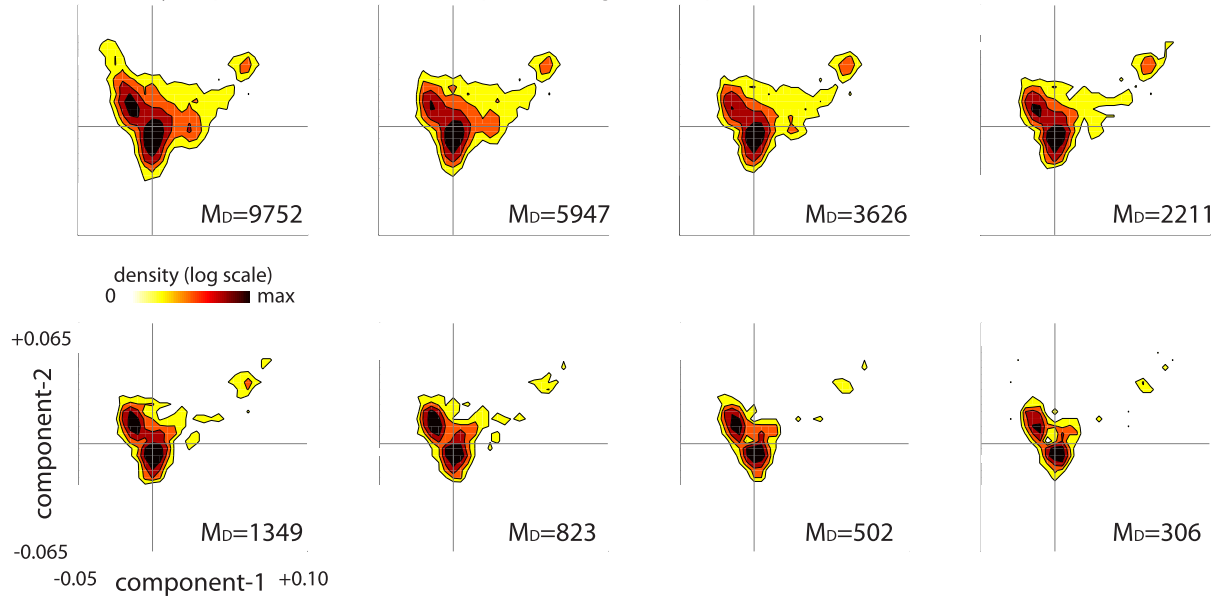


Figure 84: This figure has the same format as Fig 14, and shows the distribution of continuous-covariate components across the remaining case-patients as our algorithm proceeds. As before, our algorithm ensures that the covariate-distribution remains relatively well-balanced.

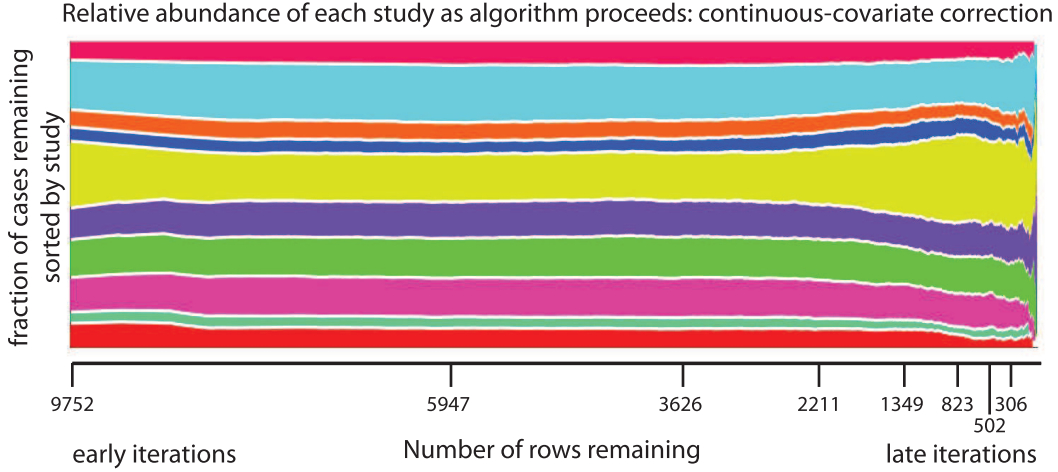


Figure 85: This figure has the same format as Fig 18, and shows the relative abundance of each study as our algorithm proceeds. As before, our algorithm is able to ensure that the remaining case-patients are drawn from a reasonable mixture of the studies.

Consequently, there are 3 different kinds of loops within D , each traversing a different pair of array-dimensions (see Fig 86). Any loop within D that does not lie entirely within B is just as likely to be rank-1 as it is to be rank-2. On the other hand, loops within D that are entirely contained within B are more likely to be rank-1 than rank-2. This probability is not 100%, but it is still significantly greater than 50%, with the exact value dependent on how the $\vec{u}^1, \vec{u}^2, \dots, \vec{u}^3$ are drawn. Moreover, this probability is still significantly greater than 50% even in the presence of a moderate amount of noise (e.g., if B were not exactly a sum of outer products).

A numerical experiment corroborating this intuition is shown in Fig 87. Techniques along these lines have been used to find triclusters in clinical data involving several patients, measurements and therapies. See [M44] for some preliminary results.

16 Various properties of multivariate gaussians

This section serves as somewhat of an appendix to section 2, collecting various observations about gaussian distributions.

16.1 one-dimensional gaussian:

We use $\rho_\sigma(x)$ to refer to the normal distribution $N(0, \sigma^2)$ on x :

$$\rho_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-x^2}{2\sigma^2}\right), \text{ with mean } 0 \text{ and variance } \sigma^2.$$

Note that the gaussian distribution ρ_σ satisfies:

$$\frac{d}{dx} \rho_\sigma(x) = -\frac{x}{\sigma^2} \rho_\sigma(x), \text{ implying}$$

$$\int_{-\infty}^{+\infty} x^{k+2} \rho_\sigma(x) dx = -\sigma^2 \int_{-\infty}^{+\infty} x^{k+1} \left(\frac{d}{dx} \rho_\sigma\right) dx = (k+1) \sigma^2 \int_{-\infty}^{+\infty} x^k \rho_\sigma(x) dx.$$

Consequently, the first few moments of ρ_σ are given by:

$$\int d\rho_\sigma = 1, \int x d\rho_\sigma = 0, \int x^2 d\rho_\sigma = \sigma^2, \int x^3 d\rho_\sigma = 0, \int x^4 d\rho_\sigma = 3\sigma^4,$$

implying that, if x is drawn from $\rho_\sigma(x)$, then the mean of x^2 will be σ^2 and the variance of x^2 will be $(3\sigma^4 - \sigma^4) = 2\sigma^4$.

One can also show that the 1-dimensional fourier-transform F of a 1-d gaussian is another 1-d gaussian:

$$F\{\rho_\sigma\}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-ikx} \rho_\sigma(x) dx = \frac{1}{\sigma} \rho_{1/\sigma}(k).$$

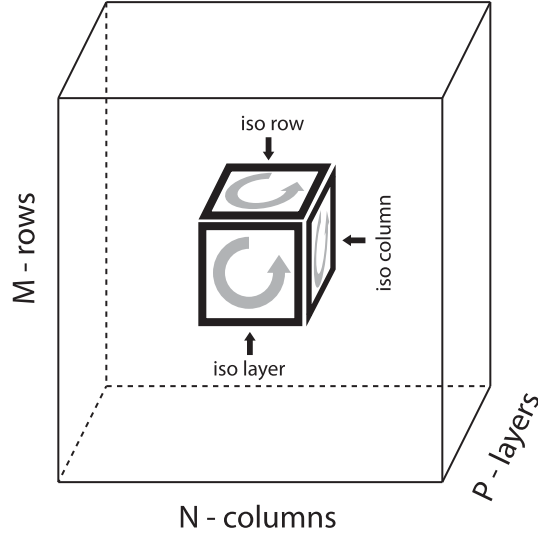


Figure 86: Illustration of the loops within a 3-dimensional array. We sketch the structure of a 3-dimensional data-array D , with J rows, K columns and P ‘layers’. Each entry $D_{j,k,l}$ will lie in the cube shown. The loops within D can be divided into 3-categories: (a) iso-layer loops that stretch across 2 rows and 2 columns, (b) iso-column loops that stretch across 2 rows and 2 layers, and (c) iso-row loops that stretch across 2 columns and 2 layers. The row-score $[Z_{\text{ROW}}]_j$ aggregates all the iso-column and iso-layer loops associated with row- j . The column-score $[Z_{\text{COL}}]_k$ aggregates all the iso-row and iso-layer loops associated with column- k . The layer-score $[Z_{\text{LYR}}]_l$ aggregates all the iso-row and iso-column loops associated with layer- l .

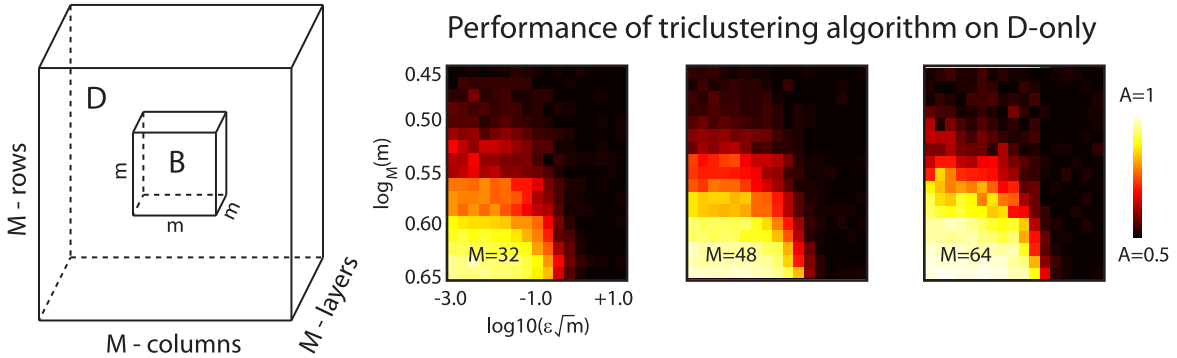


Figure 87: This numerical experiment involves a ‘planted-tricluster’ problem, where we implant an $m \times m \times m$ tricluster B within an $M \times M \times M$ array D . The setup of this experiment is shown to the far left. Each entry of D is chosen independently and randomly from a distribution with median 0. The tricluster B is constructed by first forming a sum of three outer products: $B_{j,k,l} = u_j^1 v_k^1 + u_j^2 v_k^2 + v_k^3 w_l^3$, and then flipping the sign of a randomly-chosen fraction f of the entries of B . We choose the sign-flipping probability f as described in section 5.1 (i.e., f depends on $\epsilon\sqrt{m}$). Our loop-counting algorithm produces a list of row-, column- and layer-indices of D in the order in which they are eliminated; those indices retained the longest are expected to be members of B . For each numerical experiment we calculate the auc A_R (i.e., area under the receiver operator characteristic curve) associated with the row-indices of B with respect to the output list from our algorithm. We calculate the auc A_C and A_L for the columns and layers similarly. Finally, we use $A = (A_R + A_C + A_L)/3$ as a metric of success; values of A near 1 mean that the rows, columns and layers of B were retained the longest by our algorithm. Values of A near 0.5 mean that our algorithm failed to detect B . On the right-hand side we show the trial-averaged value of A as a function of $\epsilon\sqrt{m}$ and $\log_M(m)$ for a few values of M . Note that our loop-counting algorithm is generally successful when $\epsilon\sqrt{m}$ is sufficiently small and m is sufficiently large.

16.2 convolution of two one-dimensional gaussians:

It is well known that the convolution of two one-dimensional gaussians is yet another gaussian:

$$[\rho_\alpha \star \rho_\beta](x) = \int_{-\infty}^{\infty} \rho_\alpha(x-y) \rho_\beta(y) dy = \rho_\gamma(x), \text{ where } \alpha^2 + \beta^2 = \gamma^2.$$

This can be understood by noting the fourier-transform above and appealing to the convolution theorem (i.e., $F\{\rho_\alpha \star \rho_\beta\} = F\{\rho_\alpha\} F\{\rho_\beta\}$). This can also be understood by considering a random vector x^N with N independently chosen entries, each either -1 or $+1$. Obviously, any particular entry of x^N has mean 0 and variance 1. Therefore the sum $S_N = \sum_{n=1}^{n=N} x_n^N$ is a random variable with mean 0 and variance N ; when N is large S_N will be drawn from $\rho_{\sqrt{N}}$. Now, considering a situation where $\alpha^2 + \beta^2 = \gamma^2$, we immediately see that S_{γ^2} must be drawn from ρ_γ , whereas S_{α^2} and S_{β^2} are drawn from ρ_α and ρ_β respectively. Given that there is a one-to-one mapping between vectors x^{γ^2} and vectors $x^{\alpha^2} \oplus x^{\beta^2}$, the random variables S_{γ^2} and $S_{\alpha^2} + S_{\beta^2}$ must have the same distribution. Consequently, ρ_γ must be equal to $\rho_\alpha \star \rho_\beta$.

16.3 two-dimensional gaussian:

We also use $\rho_{a,b,\theta}(\vec{x})$ to refer to the mean 0 anisotropic multivariate gaussian distribution in 2-dimensions with variances a^2 and b^2 , and first principal-component oriented at angle θ . That is to say:

$$\rho_{a,b,\theta}(\vec{x}) = \frac{1}{2\pi ab} \exp\left(-\frac{1}{2}\vec{x}^\top R_\theta \begin{bmatrix} \frac{1}{a^2} & \\ & \frac{1}{b^2} \end{bmatrix} R_\theta^\top \vec{x}\right),$$

$$\text{where } R_\theta = \begin{bmatrix} \cos \theta & -\sin \theta \\ +\sin \theta & \cos \theta \end{bmatrix}.$$

In terms of visualization, the contours of $\rho_{a,b,\theta}$ are ellipses with one principal axis pointing in the θ -direction, and the other principal axis perpendicular to the θ -direction; these elliptical contours will have principal radii proportional to a and b , respectively.

Using our notation, we see that $\rho_{a,b,0}(\vec{x})$ is separable: $\rho_{a,b,0}(\vec{x}) = \rho_a(x_1) \rho_b(x_2)$. Thus, we can think of $\rho_{a,b,\theta}(\vec{x})$ as:

$$\rho_{a,b,\theta}(\vec{x}) = \rho_{a,b,0}(R_{-\theta} \cdot x) = \rho_a(x_1 \cos \theta + x_2 \sin \theta) \cdot \rho_b(-x_1 \sin \theta + x_2 \cos \theta).$$

These observations can be easily used to show that the 2-dimensional fourier-transform F of a 2-d gaussian is yet another 2-d gaussian:

$$\begin{aligned} F[\rho_{a,b,\theta}(\vec{x})](\vec{k}) &= \frac{1}{2\pi} \int \rho_{a,b,\theta}(\vec{x}) \exp(-i\vec{k}^\top \cdot \vec{x}) dx_1 dx_2 = F[\rho_{a,b,0}(\vec{x})](R_{-\theta} \cdot \vec{k}) \\ &= \left[\frac{1}{a}\rho_{1/a} \otimes \frac{1}{b}\rho_{1/b}\right](R_{-\theta} \cdot \vec{k}) = \frac{1}{ab}\rho_{1/a,1/b,0}(R_{-\theta} \cdot \vec{k}) \\ &= \frac{1}{ab}\rho_{1/a,1/b,\theta}(\vec{k}). \end{aligned}$$

Note that $\rho_{1/a,1/b,\theta}$ has the same orientation as $\rho_{a,b,\theta}$, but with inverted principal-values.

16.4 isotropic two-dimensional gaussian:

Note that $\rho_{\sigma,\sigma,\theta}$ is an isotropic gaussian distribution in 2-dimensions with variance σ , and does not depend on θ ; the contours of $\rho_{\sigma,\sigma,\theta}$ are circles. A vector $[x_1, x_2]$ drawn from $\rho_{\sigma,\sigma,0}$ can be constructed via $\rho_{\sigma,\sigma,0}(\vec{x}) = \rho_\sigma(x_1) \rho_\sigma(x_2)$. Because $\rho_{\sigma,\sigma,0}$ is isotropic, the vector $[x_1, x_2]$ will have an orientation $\omega = \arctan(x_2/x_1)$ drawn uniformly from $[0, 2\pi]$, and a magnitude $r = \sqrt{x_1^2 + x_2^2}$ drawn from the Rayleigh distribution

$$\frac{r}{\sigma^2} \exp\left(\frac{-r^2}{2\sigma^2}\right) = \frac{\sqrt{2\pi}r}{\sigma} \rho_\sigma(r) \text{ on } r \in [0, \infty), \text{ with mean } \sigma\sqrt{\pi/2} \text{ and variance } \sigma^2(2 - \pi/2).$$

16.5 restriction of two-dimensional gaussian to quadrants:

In general, given a gaussian $\rho_{a,b,\theta}$, the fraction p of this gaussian restricted to the first and third quadrants is

$$p(a, b, \theta) = \frac{1}{\pi} \operatorname{arccot} \left(\left(\frac{b}{a} - \frac{a}{b} \right) \frac{1}{2} \sin 2\theta \right).$$

This can be determined by first observing that p is the integral of $\rho_{\alpha,b,\theta}$ over the domain Ω with boundaries given by the x and y axes, subtending an angle of π :

$$\Omega = \{(r, \theta) \mid \theta \in [0, \pi/2] \text{ or } [\pi, 3\pi/2]\}.$$

With this definition of Ω , we can write:

$$p(a, b, \theta) = \int_{\Omega} \rho_{a,b,\theta}(\vec{x}) d\vec{x}.$$

Given this formulation, we can first rotate space (and Ω) by $-\theta$:

$$p(a, b, \theta) = \int_{R_{\theta}^T \Omega} \rho_{a,b,\theta}(R_{\theta} \vec{x}) d\vec{x} = \int_{R_{\theta}^T \Omega} \rho_{a,b,0}(\vec{x}) d\vec{x},$$

and then dilate space (and $R_{\theta}^T \Omega$) by $\Sigma = \text{diag}(1/a, 1/b)$:

$$p(a, b, \theta) = \int_{\Sigma R_{\theta}^T \Omega} \rho_{a,b,0}(\Sigma^{-1} \vec{x}) d\Sigma^{-1} \vec{x} = \int_{\Sigma R_{\theta}^T \Omega} \rho_{1,1,0}(\vec{x}) d\vec{x}.$$

Within this final formulation the integrand is a uniform isotropic gaussian $\rho_{1,1,0}$, and so $p(a, b, \theta)$ is simply $1/2\pi$ times the angle subtended by the transformed domain $\Omega' = \Sigma R_{\theta}^T \Omega$. This angle in turn can be determined by applying ΣR_{θ}^T to the boundaries of Ω (i.e., to the x and y axes), yielding the formula above.

16.6 approximation of orthonormal \hat{u}, \hat{v} :

If we randomly draw the numbers u_1, \dots, u_m and v_1, \dots, v_m independently from $\rho_{\sqrt{1/m}}$, then each u_j^2 and v_j^2 will be drawn from a distribution with mean $(1/m)$ and variance $(2/m^2)$. Thus, when m is large, the random variables $|u|^2$ and $|v|^2$ will be drawn from a gaussian-distribution with mean 1 and variance $2/m$. Each term $u_j v_j$, on the other hand, will be drawn from a gaussian distribution with mean 0 and variance $1/m^2$, implying that the random variable $u \cdot v$ is drawn from $\rho_{1/m}$. Thus, the unit vector $\hat{u} = u/|u|$ is likely within $O(1/\sqrt{m})$ of u . Because \hat{u} and v are independent, the unit-vector $\tilde{v} = v - \hat{u} \hat{u}^T v$ is likely within $O(1/\sqrt{m})$ of v , and will be orthogonal to \hat{u} . Similarly, the unit-vector $\hat{v} = \tilde{v}/|\tilde{v}|$ will also be orthogonal to \hat{u} and within $O(1/\sqrt{m})$ of v . Given this construction, the unit vectors \hat{u} and \hat{v} will be orthonormal and uniformly distributed (conditioned on the fact that $\hat{u} \cdot \hat{v} = 0$). In other words, the original vectors u and v were close to orthonormal to begin with.

16.7 convolution of two two-dimensional gaussians:

Given random variables \vec{x} and \vec{y} drawn from $\rho_{a,b,\theta}$ and $\rho_{c,d,\phi}$, the sum $\vec{z} = \vec{x} + \vec{y}$ will be drawn from $\rho_{a,b,\theta} \star \rho_{c,d,\phi}$ (i.e., the distribution obtained by convolving $\rho_{a,b,\theta}$ and $\rho_{c,d,\theta}$) This distribution will also be a gaussian, of the form $\rho_{f,g,\omega}$, for some f, g, ω . This latter distribution can be understood via the fourier-transform. That is to say:

$$F\{\rho_{f,g,\omega}\}(\vec{k}) = F\{\rho_{a,b,\theta}\}(\vec{k}) \cdot F\{\rho_{c,d,\phi}\}(\vec{k}), \text{ or}$$

$$\frac{1}{fg} \rho_{1/f, 1/g, \omega}(\vec{k}) = \frac{1}{ab} \rho_{1/a, 1/b, \theta}(\vec{k}) \cdot \frac{1}{cd} \rho_{1/c, 1/d, \phi}(\vec{k}).$$

This distribution $\rho_{f,g,\omega}$ can be determined by observing:

$$\frac{1}{fg} \rho_{1/f, 1/g, \omega}(\vec{k}) = \frac{1}{2\pi} \exp\left(\frac{1}{2} \vec{k}^T R_{\omega} \begin{bmatrix} f^2 & \\ & g^2 \end{bmatrix} R_{\omega}^T \vec{k}\right), \text{ and}$$

$$\frac{1}{ab} \rho_{1/a, 1/b, \theta}(\vec{k}) \cdot \frac{1}{cd} \rho_{1/c, 1/d, \phi}(\vec{k}) = \frac{1}{4\pi^2} \exp\left(\frac{1}{2} \vec{k}^T R_{\theta} \begin{bmatrix} a^2 & \\ & b^2 \end{bmatrix} R_{\theta}^T \vec{k} + \frac{1}{2} \vec{k}^T R_{\phi} \begin{bmatrix} c^2 & \\ & d^2 \end{bmatrix} R_{\phi}^T \vec{k}\right).$$

Thus, the distribution $\rho_{f,g,\omega}$ can be determined by finding the f, g, ω that satisfy the following equations (drawn from the exponents in the previous equalities):

$$R_{\theta} \begin{bmatrix} a^2 & \\ & b^2 \end{bmatrix} R_{\theta}^T + R_{\phi} \begin{bmatrix} c^2 & \\ & d^2 \end{bmatrix} R_{\phi}^T = R_{\omega} \begin{bmatrix} f^2 & \\ & g^2 \end{bmatrix} R_{\omega}^T.$$

These equalities can be rearranged into the following:

$$\begin{aligned} (a^2 - b^2) \cos 2\theta + (c^2 - d^2) \cos 2\phi &= (f^2 - g^2) \cos 2\omega \\ (a^2 - b^2) \sin 2\theta + (c^2 - d^2) \sin 2\phi &= (f^2 - g^2) \sin 2\omega \\ [a^2 + b^2] + [c^2 + d^2] &= [f^2 + g^2]. \end{aligned} \tag{2}$$

Thus, given 2θ , $[a^2 - b^2]$, 2ϕ and $[c^2 - d^2]$, we can use the first two equalities to find 2ω and $[f^2 - g^2]$ (see, e.g., Fig 88). Once this is accomplished we can use the third equality to find $[f^2 + g^2]$.

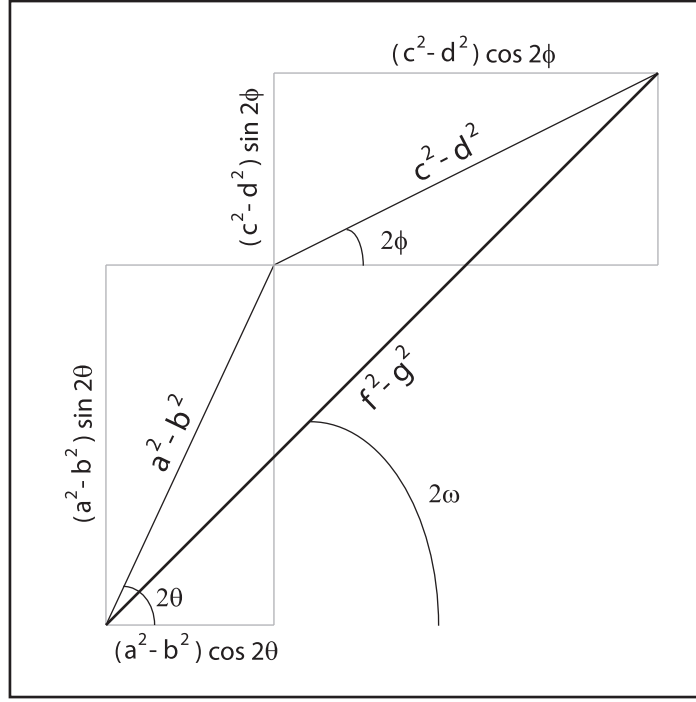


Figure 88: In this figure we show the relationships between $(f^2 - g^2)$, ω , $(a^2 - b^2)$, θ , $(c^2 - d^2)$, ϕ that are expressed in Eq. 2.

17 Asymptotic formula for $g_{1,\varepsilon,m}$:

When $\varepsilon\sqrt{m}$ is large (say, greater than 10), then Eq. 1 can be approximated via:

$$p\left(\omega, \frac{\varepsilon\sqrt{m}}{r}\right) \approx \frac{1}{2\pi} \left\{ \pi + \sin(2\omega) \frac{r^2}{\varepsilon^2 m} \right\}, \text{ and } g_{1,\varepsilon,m} \approx \frac{1}{2} + \frac{2}{\pi^2 \varepsilon^4 m^2}.$$

When $\varepsilon\sqrt{m}$ is small (say, less than 0.01), then Eq. 1 can be approximated via:

$$p\left(\omega, \frac{\varepsilon\sqrt{m}}{r}\right) \approx \frac{1}{\pi} \frac{\varepsilon\sqrt{m}}{r} \cdot \frac{1}{-\frac{1}{2} \sin 2\omega + \frac{2}{\pi} \frac{\varepsilon\sqrt{m}}{r}} = \frac{1}{2} \left[1 - \frac{\pi}{4} \frac{r}{\varepsilon\sqrt{m}} \sin 2\omega \right]^{-1} \text{ when } \sin 2\omega < 0, \text{ and}$$

$$p\left(\omega, \frac{\varepsilon\sqrt{m}}{r}\right) \approx 1 - \frac{1}{\pi} \frac{\varepsilon\sqrt{m}}{r} \cdot \frac{1}{+\frac{1}{2} \sin 2\omega + \frac{2}{\pi} \frac{\varepsilon\sqrt{m}}{r}} = 1 - \frac{1}{2} \left[1 + \frac{\pi}{4} \frac{r}{\varepsilon\sqrt{m}} \sin 2\omega \right]^{-1} \text{ when } \sin 2\omega > 0.$$

The integral associated with g is then given by:

$$\iint \frac{1}{2\pi} \{1 - 2p + 2p^2\} r e^{-r^2/2} d\theta dr = \int_0^\infty \left\{ r - \frac{4}{\pi} \varepsilon\sqrt{m} \left\{ \frac{2}{\pi} \log \left(\frac{\pi}{2} \frac{r}{\varepsilon\sqrt{m}} \right) + \frac{1}{\pi} \right\} \right\} e^{-r^2/2} d\theta dr,$$

implying that:

$$g_{1,\varepsilon,m} \approx 1 - \frac{4}{\pi} \varepsilon\sqrt{m} \left[\frac{1}{\sqrt{2\pi}} + \frac{2}{\sqrt{2\pi}} \log \left(\frac{\pi}{2\varepsilon\sqrt{m}} \right) - \frac{2}{\pi} K \right],$$

where $K = - \int_0^\infty \log(r) \exp(-r^2/2) dr \approx 0.7961$.

18 Low-noise limit for $B^\top B$:

Let's consider a noiseless $m \times n$ rank- l bicluster B (i.e., with $\varepsilon = 0$). We can analytically solve for the expected value of the magnitude of the covariances $[B^\top B]_{kk'}$ in the limit $m, n \rightarrow \infty$.

Recall from section 5 that each column $B_{:,k}$ is obtained as follows: First we draw a sample vector $A_{:,k}$ of size $m \times 1$ from an eccentric gaussian-distribution on \mathbb{R}^m with l principal-values equal to 1, and the remaining principal-values equal to 0. Then we define $B_{:,k}$ to be the binarization $B_{:,k} = \text{sign}(A_{:,k})$.

Because $\varepsilon = 0$, we construct the matrix A as follows:

$$A_{jk} = u_{j1}v_{k1} + \cdots + u_{jl}v_{kl} = \sum_{q=1}^l U_{jq}V_{qk}^\top,$$

where $U \in \mathbb{R}^{m \times l}$ and $V \in \mathbb{R}^{n \times l}$, and $A = UV^\top$ is a singular-value-decomposition. Defining

$$\vec{u}_j = U_j \text{ and } \vec{v}_k = V_k:$$

to be the j^{th} -row of U and the k^{th} -row of V , respectively, we can write A as:

$$A_{jk} = \vec{u}_j \cdot \vec{v}_k.$$

This allows us to write B as:

$$B_{jk} = \text{sign}(\vec{u}_j \cdot \vec{v}_k),$$

and $[B^\top B]$ as:

$$[B^\top B]_{kk'} = \sum_{j=1}^m \text{sign}(\vec{u}_j \cdot \vec{v}_k) \text{sign}(\vec{u}_j \cdot \vec{v}_{k'}).$$

For any particular $[B^\top B]_{kk'}$ the row-vectors \vec{v}_k and $\vec{v}_{k'}$ are fixed. Any particular summand (corresponding to a particular j) will be $+1$ if that row-vector \vec{u}_j makes an acute angle with both \vec{v}_k and $\vec{v}_{k'}$. Similarly, a summand will be $+1$ if \vec{u}_j makes an obtuse angle with both \vec{v}_k and $\vec{v}_{k'}$. On the other hand, a summand will be -1 if and only if \vec{u}_j makes an acute angle with one of the $\vec{v}_k, \vec{v}_{k'}$, and an obtuse angle with the other. If m is large, we can assume that each of the \vec{u}_j are independent and randomly oriented. Thus, if we let $\theta \in [0, \pi]$ be the angle between \vec{v}_k and $\vec{v}_{k'}$, then any particular \vec{u}_j has probability θ/π of contributing a summand of -1 to $[B^\top B]_{kk'}$. Conversely, any particular \vec{u}_j has probability $1 - \theta/\pi$ of contributing a summand of $+1$. Thus, given a fixed set of \vec{v}_k and $\vec{v}_{k'}$, the expected value of $[B^\top B]_{kk'}$ will be $m(1 - 2\theta/\pi)$.

To determine the expected value of $[B^\top B]_{kk'}$ for an arbitrary \vec{v}_k and $\vec{v}_{k'}$, we simply integrate $m(1 - 2\theta/\pi)$ over the distribution $\mu(\theta)$ of θ . This distribution θ depends on the ambient dimension l :

$l = 1$: The \vec{v}_k and $\vec{v}_{k'}$ lie on a line, and are either parallel or antiparallel: $\mu(\theta) = \delta(\theta - 0) + \delta(\theta - \pi)$.

$l = 2$: The \vec{v}_k and $\vec{v}_{k'}$ lie on a circle: $\mu(\theta) = 1/\pi$

$l = 3$: The \vec{v}_k and $\vec{v}_{k'}$ lie on a sphere: $\mu(\theta) = \sin(\theta)/2$

$l = 4$: The \vec{v}_k and $\vec{v}_{k'}$ lie on a sphere in $l = 4$ dimensions: $\mu(\theta) = \sin^2(\theta)/(\pi/2)$

$l > 4$: The \vec{v}_k and $\vec{v}_{k'}$ lie on a sphere in l dimensions: $\mu(\theta) = \sin^{l-2}(\theta)/Z_l$.

In the cases where $l \geq 2$, the normalization factor Z_l is equal to:

$$Z_l = \int_0^\pi \sin^{l-2}(\theta) d\theta = 2Y_{l-2}, \text{ where}$$

$$Y_q = \begin{cases} \frac{(q-1)(q-3)(q-5)\cdots(1)\pi}{(q-0)(q-2)(q-4)\cdots(2)2} & \text{if } l \text{ is even} \\ \frac{(q-1)(q-3)(q-5)\cdots(2)1}{(q-0)(q-2)(q-4)\cdots(1)} & \text{if } l \text{ is odd} \end{cases}$$

Note that the first few Y_q are given by:

$$Y_0 = \pi/2, Y_1 = 1, \text{ and } Y_q = \frac{q-1}{q}Y_{q-2} \text{ for } q \geq 2.$$

Using this construction we can calculate the expected value of magnitude of $[B^\top B]_{kk'}$, assuming that $k \neq k'$:

$$E(|[B^\top B]_{kk'}|) = m \int_0^\pi |1 - 2\theta/\pi| \mu(\theta) d\theta = 2m \int_0^{\pi/2} (1 - 2\theta/\pi) \mu(\theta) d\theta.$$

Noting that:

$$\int_0^{\pi/2} \sin^q(\theta) d\theta = Y_q, \text{ and } \int_0^{\pi/2} \theta \cdot \sin^q(\theta) d\theta = X_q, \text{ with}$$

$$X_0 = \frac{\pi^2}{8}, X_1 = 1, \text{ and } X_q = \frac{1}{q^2} + \frac{q-1}{q}X_{q-2} \text{ for } q \geq 2,$$

we see that:

$$E(|[B^\top B]_{kk'}|) = m \left[1 - \frac{2}{\pi} \frac{X_{l-2}}{Y_{l-2}} \right].$$

Evaluating this expression for the first few l gives:

$$l = 1 : E(|[B^\top B]_{kk'}|) = m \cdot 1$$

$$l = 2 : E(|[B^\top B]_{kk'}|) = m \cdot [1 - 1/2] = m/2$$

$$l = 3 : E(|[B^\top B]_{kk'}|) = m \cdot \left[1 - \frac{2}{\pi} \right] \approx m \cdot 0.3634$$

$$l = 4 : E(|[B^\top B]_{kk'}|) = m \cdot \left[1 - \frac{2}{\pi} (1/4 + \pi^2/16) / (\pi/4) \right] \approx m \cdot 0.2974$$

While the following is not an exact formula, when $l \geq 2$ this expected value is quite close to:

$$E(|[B^\top B]_{kk'}|) \approx \frac{m}{2\sqrt{l-1}}.$$

19 Information-theoretic phase-transitions for rank-1 planted-bicluster problem

In this section we discuss in more detail the information-theoretic size- and noise-transitions for the rank-1 planted-bicluster problem. This problem involves sifting through an $M \times M$ binary data-matrix D , and distinguishing between the following two hypotheses.

H0: The null hypothesis. In this hypothesis the data-matrix D has no planted bicluster within it; each entry of D is chosen randomly and independently from $\{-1, +1\}$.

H1: The planted-bicluster hypothesis: In this hypothesis the data-matrix D is altered by planting within it a smaller $m \times m$ submatrix which corresponds to the binarization of a randomly-oriented rank-1 error- ε bicluster. The entries of D outside of this $m \times m$ submatrix will be chosen randomly and independently as before, but the entries of D within this $m \times m$ submatrix will depend on one another.

One step towards understanding the planted-bicluster problem is to determine when H1 is distinguishable from H0; i.e., how large m needs to be (and how small ε must be) before the planted-bicluster impacts the distribution of the elements of D . To address this question we assume that D is drawn from H0, and ask what kinds of structures might naturally exist within D , even without any deliberate attempt to plant any specific biclusters. As we'll see below, even under H0 we still expect to find within D a vast number of biclusters. These biclusters include (a) small noiseless biclusters of size $m \lesssim 2 \log_2 M$ and (b) much larger and noisier biclusters of size $m \sim \sqrt{M}$ and $\varepsilon \gtrsim [\pi \log M]^{-1/2}$, as well as many structured elements in between these two extremes. Nevertheless, under H0 we are exponentially unlikely to find biclusters that are both large and low-noise.

Size phase-transition: Let's assume the noise $\varepsilon = 0$ for the moment, and consider the limit $M \gg m \gg 1$. Let's take a particular $m \times m$ submatrix B within D . The requirement that B be rank-1 places constraints on the signs of $(m-1)^2$ elements of B . That is, B will be rank-1 with probability $p_0 = 2^{-\binom{m-1}{2}}$, or approximately $-\log p_0 \approx \log 2 \cdot m^2$. The total number of such $m \times m$ submatrices within D is $Z = \binom{M}{m}^2$, which can be approximated as $Z \approx \exp(2m \log(M/m) + 2m + 2m^2/M)$, or approximately $\log Z \approx 2m \log M$. The product $P_0 = 1 - (1 - p_0)^Z \approx Z p_0$ provides an upper bound on the probability that D contains at least one $m \times m$ rank-1 submatrix (this is technically an overestimate because the ranks of each of the Z submatrices are not independent of one another). Obviously, this upper bound P_0 tends to 0 when $-\log p_0 \gg \log Z$, which occurs when m is larger than $[2/\log 2] \log M = 2 \log_2 M$. This means that, for $M \gg m \gg 1$, a random binary-matrix D is exponentially unlikely to contain any $m \times m$ rank-1 matrix, as long as $m \gtrsim 2 \log_2 M$. One consequence of these observations is that, if we were to embed within D a submatrix B of size m in the range $2 \log_2(M) < m < \sqrt{M}$, say $m \sim \sqrt[3]{M}$, then B would be very significant but still undetectable to our algorithm.

Noise phase-transition: Now let's consider the second phase-transition, this time assuming $M \gg m \gg \log M$. As before, we'll assume that D is an $M \times M$ binary-matrix with each entry chosen independently from $\{-1, +1\}$, and we'll consider $m \times m$ submatrices B within D . This time we are interested in the probability $p_{\varepsilon\sqrt{m}}$ that a particular B is rank-1 with noise at most $\varepsilon\sqrt{m}$. As we pointed out above, when $\varepsilon\sqrt{m} = 0$ we impose constraints on the signs of $\approx m^2$ elements of B ; any particular B will be exactly rank-1 with probability $p_0 \approx 2^{-m^2}$. If instead we require B to be rank-1 with noise $\varepsilon\sqrt{m}$, we actually only impose constraints on $\approx (1 - 2f_{\varepsilon\sqrt{m}}) m^2$ randomly chosen elements of B

(where $f_{\varepsilon\sqrt{m}}$ is the ‘flip-probability’ discussed in section 5.1). Assuming that $\varepsilon\sqrt{m} \gtrsim 10$, we can approximate $f_{\varepsilon\sqrt{m}}$ with $f_{\varepsilon\sqrt{m}} \approx 1/2 - [\sqrt{2\pi} \cdot \varepsilon\sqrt{m}]^{-1}$. An asymptotic expansion in terms of $\varepsilon\sqrt{m}$ reveals that, within the large- $\varepsilon\sqrt{m}$ regime:

$$p_{\varepsilon\sqrt{m}} \approx \sum_{f m^2=0}^{f m^2=f_{\varepsilon\sqrt{m}} \cdot m^2} \binom{m^2}{f m^2} \frac{1}{2^{m^2}} \approx \frac{1}{2} \frac{\varepsilon\sqrt{m}}{m} \cdot \exp\left(-\frac{m^2}{\pi \varepsilon^2 m}\right).$$

Thus, we see that:

$$-\log p_{\varepsilon\sqrt{m}} \approx \frac{m^2}{\pi (\varepsilon\sqrt{m})^2} + \log m + \log 2 - \log(\varepsilon\sqrt{m}).$$

Comparing this with $\log Z = 2m \log M - 2m(\log m - 1)$, we see that $P_{\varepsilon\sqrt{m}} = 1 - (1 - p_{\varepsilon\sqrt{m}})^Z \approx Z p_{\varepsilon\sqrt{m}}$ vanishes when

$$\frac{m^2}{\pi (\varepsilon\sqrt{m})^2} + \log m + \log 2 - \log(\varepsilon\sqrt{m}) \gg 2m \log M - 2m(\log m - 1).$$

If we assume that $m \sim \sqrt{M}$, then $P_{\varepsilon\sqrt{m}}$ will vanish when

$$\frac{1 + 1/M \cdot \log 2 - 1/M \cdot \log(\varepsilon\sqrt{m})}{\log M + 2} \gg \pi \varepsilon^2.$$

Consequently, when $\varepsilon\sqrt{m} \gtrsim 10$ and $m \sim \sqrt{M}$, we can expect that $P_{\varepsilon\sqrt{m}}$ will be essentially 0 whenever $\pi \varepsilon^2 \ll 1/\log M$, or when $\varepsilon\sqrt{m} \ll M^{1/4}/\sqrt{\pi \log M}$. This implies that a large random $M \times M$ matrix D is exponentially unlikely to contain any $\sqrt{M} \times \sqrt{M}$ rank-1 biclusters B which have a first-principal-component accounting for $\gtrsim \pi \log M/\sqrt{M}$ of their total variance. Put another way, if we were to embed within D an ε -error rank-1 $\sqrt{M} \times \sqrt{M}$ bicluster B which had, say, $\varepsilon \sim 1/\sqrt[3]{M}$, then B would be very statistically significant but still undetectable to our algorithm.

20 Does binarization destroy angular information?

This section describes the details of Fig 28. The scenario we investigate involves loops drawn from an $m \times m$ ε -error rank-1 bicluster B . If we select 2 columns of B at random, any 1×2 row-vector from these two columns will be drawn from the 2-dimensional distribution $\tilde{\rho}$ described in section 5. The distribution $\tilde{\rho} \sim \rho_{r_1, 0, \theta_1} \star \rho_{\varepsilon, \varepsilon, 0}$, where r_1 is drawn from $r\sqrt{2\pi m\rho}/\sqrt{1/m}(r)$ and θ_1 is uniformly distributed on $[0, 2\pi]$. Given r_1 and $\theta_1 = 0$, a 1×2 vector randomly drawn from $\tilde{\rho}$ gives rise to an angle ω drawn from:

$$\rho_B^{\text{original}}(\omega) \sim \frac{1}{2\pi} \frac{ab}{b^2 \cos^2(\omega - \theta_1) + a^2 \sin^2(\omega - \theta_1)}, \text{ with } a = \sqrt{r_1^2 + \varepsilon^2}, \text{ and } b = \sqrt{0^2 + \varepsilon^2} = \varepsilon.$$

By contrast, the angle ω of a 1×2 vector drawn from outside B is uniformly distributed (i.e., $\rho_D^{\text{original}}(\omega) = 1/2\pi$). These two distributions can be used to calculate the relative-entropy $\mathcal{D}_{KL}^{\text{original}}(\varepsilon\sqrt{m})$ associated with the original bicluster. After binarization, the distributions $\rho_B^{\text{binarized}}(\omega)$ and $\rho_D^{\text{binarized}}(\omega)$ are given by:

$$\begin{aligned} \rho_B^{\text{binarized}}(\omega) &= \begin{cases} \frac{1}{\pi} \operatorname{arccot}\left(\left|\frac{b}{a}\right| - \left|\frac{a}{b}\right|\right) \frac{1}{2} \sin 2\theta_1 & \text{for } \omega = \pi/4 \text{ or } 5\pi/4 \\ \frac{1}{\pi} \operatorname{arccot}\left(\left|\frac{a}{b}\right| - \left|\frac{b}{a}\right|\right) \frac{1}{2} \sin 2\theta_1 & \text{for } \omega = 3\pi/4 \text{ or } 7\pi/4 \end{cases} \\ \rho_D^{\text{binarized}}(\omega) &= \begin{cases} 1/2 & \text{for } \omega = \pi/4 \text{ or } 5\pi/4 \\ 1/2 & \text{for } \omega = 3\pi/4 \text{ or } 7\pi/4 \end{cases} \end{aligned}$$

These two distributions can be used to calculate the relative-entropy $\mathcal{D}_{KL}^{\text{binarized}}(\varepsilon\sqrt{m})$ associated with the binarized version of the bicluster. Shown in Fig 28A are several randomly drawn distributions of this type (each drawn with a different r_1 and θ_1). As illustrated in Fig 28B, there is little difference between $\mathcal{D}_{KL}^{\text{original}}$ and $\mathcal{D}_{KL}^{\text{binarized}}$ when $\varepsilon\sqrt{m} \gtrsim 0.01$.

21 Analysis of loop-scores

In this section we analyze our loop-scores and show that they are, in a certain sense, close to optimal within the context of our methodology. In this section we’ll use the notation $l[J; K]$ to refer to a $c \times c$ -submatrix involving rows $J = \{j_1, \dots, j_c\}$ and columns $K = \{k_1, \dots, k_c\}$. To ease the discussion, we’ll refer to these $c \times c$ submatrices as ‘nets’. Our analysis begins by noting that our original loop-score was just one of a large class of ‘ c -scores’ which are constructed using the following general template:

Step (i) Given a row j , consider the set S_j of all $c \times c$ -nets $l[j_1, \dots, j_c; k_1, \dots, k_c]$ with $j \in \{j_1, \dots, j_c\}$.

Step (ii) Measure some scalar quantity $\mu(l)$ for each $l \in S_j$;

Step (iii) Aggregate the sum $[Z_{\text{ROW}}]_j = \sum_{l \in S_j} \mu(l)$.

Our original loop-scores fall within the above template using $c = 2$ and $\mu(l)$ related to the rank of l (i.e., $\mu(l) = 3 - 2 \cdot \text{rank}(l)$). It is natural to ask: Could we gain an advantage if we allowed for the more general steps (i) and (ii) above? Specifically, instead of merely looking at loops $l[j_1, j_2; k_1, k_2]$, we might consider larger $c \times c$ nets $l[j_1, \dots, j_c; k_1, \dots, k_c]$. Moreover, instead of simply measuring the rank $\mu(l)$, we could use some more informative measurement instead (e.g., any other scalar-valued function $\mu : \{-1, +1\}^{c \times c} \rightarrow \mathbb{R}$). It turns out that neither of these modifications can substantially improve the c -score; no matter which fixed c we choose, or how we construct μ , we won't be able to overcome the size-threshold of \sqrt{M} .

To see why this might be the case, let's consider a strictly easier problem: let's assume that the signs of u and v are known to be all positive; i.e., B is simply a rank-1 $m \times m$ matrix with each entry equal to $+1$. We'll show below that no algorithm built using the template above can overcome the size-threshold of \sqrt{M} for this strictly easier problem. This then suggests that the same size-threshold applies to our original planted-bicluster problem.

In step (i) of the template, each net $l[J; K]$ is defined with the restriction that $j \in J$. Any net l may overlap B ; there may be $x = |J \cap J_B|$ rows in common and $y = |K \cap K_B|$ columns in common. If $x = 0$ and $y = 0$, then $l[J; K]$ is drawn from outside B . While the joint-distribution across all such nets is highly structured, each individual net drawn from outside B looks 'fully-random'. That is to say, if a single net is drawn from outside B , the joint-distribution from which its own entries are drawn will be indistinguishable from that of a net whose entries are each drawn independently from $\{-1, +1\}$. Any individual net with $x = 0$ or $y = 0$ will also look fully-random. Only when both $x \geq 1$ and $y \geq 1$ will the structure of B impact the distribution from which the entries of $l[J; K]$ are drawn. When $x = 1$ and $y = 1$ the entries of $l[J; K]$ no longer look fully random; there will be some randomly-located entry that is guaranteed to be equal to $+1$, and the remaining entries of l will be drawn independently from $\{-1, +1\}$. As x and y increase, the structure of B impacts the entries of $l[J; K]$ more and more severely, restricting a larger and larger fraction of $l[J; K]$ to be positive until, when $x = m$ and $y = m$, the entire net $l[J; K]$ is forced to be $+1$.

It turns out that, in the limit as M and m go to infinity (with $m/M \rightarrow 0$), only the singly-overlapping nets with $x = 1$ and $y = 1$ contribute meaningfully to the sum in step (iii). That is to say, we can safely ignore all nets with multiple overlapping entries (i.e., with $x + y > 2$). There are two reasons which, together, ensure that this is the case.

1. The first reason is that, as m and M get larger, there will be $\sim m^{x+y} M^{2c-x-y}$ different nets with row- and column-overlaps of x and y , respectively. If the given row j lies inside B , then there will be $\sim m^1 M^{2c-2}$ nets with $x = 1, y = 1$, and only $\sim m^2 M^{2c-3}$ multiply-overlapping nets with $x + y > 2$. Similarly, if j lies outside B , then there will be $\sim m^2 M^{2c-3}$ nets with $x = 1, y = 1$, and only $\sim m^3 M^{2c-4}$ multiply-overlapping nets. In both cases the fraction of nets which are impacted by the presence of B is dominated by those nets with the smallest possible overlap – in this case $x = 1$ and $y = 1$. The fraction of multiply-overlapping nets will always be $\sim m/M$ smaller than the fraction of singly-overlapping nets.
2. The second reason is that, no matter the choice of c , the distributions of possible configurations allowed within the multiply-overlapping nets is always comparable to the distribution of possible configurations allowed within the singly-overlapping nets. In other words, the relative-entropy between these two configuration-distributions remains bounded by a constant depending only on c , regardless of the choice of x or y for the multiply-overlapping nets. Put another way, in order for us to safely ignore multiply-overlapping nets, it is imperative that these nets cannot adopt a configuration forbidden to singly-overlapping nets. This is the case here: there are only $2^{c \cdot c}$ possible configurations allowed for any net, and all of these configurations can be found (with probability bounded away from 0) amongst either the singly- or multiply-overlapping nets. We remark that a similar statement would not hold if, say, the data were not binarized and the planted bicluster were exactly rank-1 with unknown u and v . In such a scenario any multiply overlapping nets with $c = 2, x = 2$ and $y = 2$ would all have a second singular value of exactly 0. If we were to compare this configuration-distribution with that of the $c = 2, x + y < 4$ nets, we would find the relative-entropy unbounded: the multiply-overlapping nets contain infinitely more information than the singly-overlapping nets, and cannot be ignored (this is the same scenario we brought up earlier when discussing the information-loss associated with binarization).

Given the justification above, let's restrict our attention to $l[J; K]$ for which $x = 1$ and $y = 1$. These were the nets which contained a single randomly-located entry that was fixed to be $+1$ (with the other entries randomly chosen to be either $+1$ or -1). Because only a single entry within each net has a fixed sign, we don't need to worry about the joint-distribution of signs across entries. Therefore, the only relevant quantity for each of these nets is the number $\chi(l)$ of positive entries within l (i.e., $\chi(l)$ ranges from 0 to c^2). Thus, any μ that we construct must be a function of $\chi(l)$ (i.e., $\mu = \mu(\chi(l))$). Consequently, given step (iii), our final c -score for row j must be of the form $[Z_{\text{ROW}}]_j = \mu(0) L_0^j + \dots + \mu(c^2) L_{c^2}^j$, where

L_χ^j is the total number of nets intersecting row j that contain exactly χ positive entries. In other words, the final c -score for row j takes the form $[Z_{\text{ROW}}]_j = \vec{\mu} \cdot \vec{L}^j$, where $\vec{\mu} \in \mathbb{R}^{1+c^2}$ is the vector formed from the various $\mu(\chi)$, and $\vec{L}^j \in \mathbb{N}^{1+c^2}$ is the vector formed from the L_χ^j . The difference between the c -score for some row j and the c -score for another row j' depends only on the difference between \vec{L}^j and $\vec{L}^{j'}$.

To understand the structure of \vec{L}^j more clearly, let's restrict our attention to a fixed column-subset $K = \{k_1, \dots, k_c\}$. Given that column-subset K is fixed, we can consider various row-subsets $\bar{J} = \{j_1, \dots, j_{c-1}\}$, each chosen such that $j \notin \bar{J}$ and $\bar{J} \cap J_B = \emptyset$. For each choice of \bar{J} , the $(c-1) \times c$ sub-net $l[\bar{J}; K]$ will have some number of positive entries $\chi(l[\bar{J}; K])$. This collection of values for χ (taken across the various \bar{J}) can be encoded as a vector $\vec{P}^K \in \mathbb{N}^{1+(c-1)c}$ (i.e., \vec{P}_χ^K is the number of row-subsets \bar{J} such that $\chi(l[\bar{J}; K]) = \chi$). Now the $(c-1) \times c$ sub-net $l[\bar{J}; K]$ can be expanded by adding the row $l[j; K]$ to form $l[j \cup \bar{J}; K]$. Obviously, $\chi(l[j \cup \bar{J}; K])$ is equal to $\chi(l[j; K]) + \chi(l[\bar{J}; K])$, implying that the collection of values for $\chi(l[j \cup \bar{J}; K])$ (taken across the various \bar{J}) can be determined simply by embedding the vector \vec{P}^K in \mathbb{N}^{1+c^2} and shifting it forward by $\chi(l[j; K])$ indices. That is to say, the collection of values for $\chi(l[j \cup \bar{J}; K])$ is simply $\vec{P}^K \star \vec{e}_{\chi(l[j; K])}$, where \star denotes a vector-convolution and \vec{e}_χ is a vector with entry χ equal to 1 and the other entries equal to 0.

With this notation the vector \vec{L}^j can be written as:

$$\vec{L}^j = \sum_K \vec{P}^K \star \vec{e}_{\chi(l[j; K])},$$

where the vector \vec{P}^K is calculated by summing over all row-subsets \bar{J} that exclude j . At this point we are ready to compare the row j , which we'll assume lies in J_B , with another row j' taken from outside B . We can immediately see that:

$$\vec{L}^j - \vec{L}^{j'} = \sum_K \vec{P}^K \star [\vec{e}_{\chi(l[j; K])} - \vec{e}_{\chi(l[j'; K])}],$$

where the vector \vec{P}^K is calculated by summing over all row-subsets \bar{J} that exclude both j and j' (i.e., nets which include both j and j' will contribute to the same χ -index within \vec{L}^j and $\vec{L}^{j'}$). Recalling our discussion above, we need only consider nets $l[\bar{J}; K]$ that have an overlap $x = |J \cap J_B|$, and $y = |K \cap K_B|$ of at most 1. This implies that we can calculate the vector \vec{P}^K by summing over only those row sub-sets \bar{J} that exclude J_B as well as j' . With this assumption each $l[\bar{J}; K]$ is disjoint from B , and each $\chi(l[\bar{J}; K])$ will be drawn from the same binomial distribution

$$P(\chi(l[\bar{J}; K]) = \chi) = \binom{(c-1)c}{\chi} 2^{-(c-1)c}.$$

Thus the expected value of $\sum_K \vec{P}^K$ is merely the binomial distribution above multiplied by the number of possible \bar{J} :

$$E\left(\sum_K \vec{P}_\chi^K\right) = \binom{M-m-1}{c-2} \binom{(c-1)c}{\chi} 2^{-(c-1)c}.$$

Based on the linearity of expectation and the independence of \vec{P}^K and $\chi(l[j; K])$ and $\chi(l[j'; K])$, we have that:

$$E(\vec{L}^j - \vec{L}^{j'}) = E\left(\sum_K \vec{P}_\chi^K\right) \star [\vec{E}_j^K - \vec{E}_{j'}^K], \text{ where}$$

$$[\vec{E}_j^K]_\chi = \left[E \left\{ \sum_K \vec{e}_{\chi(l[j; K])} \right\} \right]_\chi \sim \binom{M}{c} \left\{ \frac{M-m}{M} \binom{c}{\chi} 2^{-c} + \frac{m}{M} \binom{c-1}{\chi-1} 2^{-c+1} \right\}, \text{ and}$$

$$[\vec{E}_{j'}^K]_\chi = \left[E \left\{ \sum_K \vec{e}_{\chi(l[j'; K])} \right\} \right]_\chi \sim \binom{M}{c} \binom{c}{\chi} 2^{-c}.$$

Thus, the ability for Z_{ROW} to discriminate between j and j' is limited by the difference between the distributions \vec{E}_j^K and $\vec{E}_{j'}^K$. A calculation of their relative-entropy shows that these two distributions \vec{E}_j^K and $\vec{E}_{j'}^K$ are maximally differentiable when $c = 1$. When $c = 1$ the distribution $E(\sum_K \vec{P}_\chi^K)$ is trivial and the choice of $\vec{\mu}$ is irrelevant (so long as $\mu(0) \neq \mu(1)$). In this case the best c -score for each row is simply the number of positive entries in that row. This is equivalent to simply using the 'degree' of each row (treating the matrix D as the adjacency matrix of a bigraph). A similar statement holds for the columns: The best c -score for each column is simply the number of positive entries in that column. This simple score – i.e., counting degree – is only informative when $m \gtrsim \sqrt{M}$.

Putting all this together, we see that – for the easier problem when B is a matrix of all ‘+1’s – there is no advantage to be gained by considering larger $c \times c$ -submatrices, or by considering a more general measurement $\mu(l)$. The choice of $c = 1$ is asymptotically optimal and, regardless of the choice of μ , the scores will only be indicative of J_B and K_B when $m \gtrsim \sqrt{M}$ (and won’t be indicative of B when $m = o(\sqrt{M})$). In other words; algorithms built using the template above will always be subject to the same size-threshold of $m \gtrsim \sqrt{M}$. Therefore, we conclude that our original problem also suffers from the same size-threshold limitation; regardless of our choice of c or μ , the template above won’t be able to construct scores which indicate the rows and columns of B when $m = o(\sqrt{M})$.

To summarize: we have argued that the size-transition exhibited by our loop-scores will also be exhibited by a rather large class of similar scores. To be more specific, we conclude that – for fixed $\varepsilon\sqrt{m}$ – all scores constructed using the template above will be informative only when $m \gtrsim \sqrt{M}$, and will not be informative when $m = o(\sqrt{M})$. This then implies that – when $\varepsilon\sqrt{m} = 0$ – our loop-scores are asymptotically optimal (within the constraints of the template above).

22 Comparison with a simple spectral-biclustering method

Our loop-counting algorithm is closely related to ‘spectral-biclustering’ [M22,M40]. Indeed, if we ignore the correction for collapsed-loops, we see that our loop-score $[Z_{\text{ROW}}]_j$ is the diagonal entry $[DD^\top DD^\top]_{jj}$, which is proportional to the rayleigh-quotient $\vec{x}^\top [D^\top D] \vec{x}$, with $\vec{x}^\top = D_{j,:}$ representing the j^{th} -row of D . Similarly, $[Z_{\text{COL}}]_k$ is the rayleigh-quotient $\vec{y}^\top [DD^\top] \vec{y}$, with $\vec{y} = D_{:,k}$ representing the k^{th} -column of D . Our algorithm attempts to focus on those rows and columns which correspond to large rayleigh-quotients; i.e., to those rows and columns which are most parallel to the dominant eigenvectors of $D^\top D$ and DD^\top , respectively. If we decompose $D = U\Sigma V^\top$ using the singular-value-decomposition, we see that the dominant eigenvectors of $D^\top D$ and DD^\top are equal to the dominant right- and left-singular-vectors $\vec{v} = V_{:,1}$ and $\vec{u} = U_{:,1}$ of the data-matrix D . Therefore, one can imagine that the rows and columns which our algorithm focuses on will be those that are most highly correlated with \vec{v} and \vec{u} , respectively. These rows and columns should thus correspond to high-magnitude components within the vectors \vec{u} and \vec{v} , respectively.

Given these observations, it is natural to consider the following simple spectral-biclustering method:

Step 0 Binarize D , sending each entry to either +1 or -1 (i.e., $D = \text{sign}(D)$ or $D = 2(D > 0) - 1$).

Step 1 Calculate the singular-value-decomposition $D = U\Sigma V^\top$. Set \vec{u} and \vec{v} to be the first columns of U and V .

Step 2 Set $[Z_{\text{ROW}}]_j = \vec{u}_j^2$ and $[Z_{\text{COL}}]_k = \vec{v}_k^2$ to be the entrywise-squares of the singular-vectors \vec{u} and \vec{v} .

Step 3 Use Z_{ROW} and Z_{COL} to produce a ranked list of rows and columns of D .

Another way to think about the connection between our loop-counting algorithm and this spectral-biclustering method is to first consider the bigraph corresponding to the binarized version of D . Within this context, we compute the row-loop-score by considering each $(1 + 1) \times 2 = 4$ -step path $(j, k) \rightarrow (j', k) \rightarrow (j', k') \rightarrow (j, k')$ through the bigraph associated with D (see Fig 89). Let us denote such a path by $p[J; K]$, indexed by ordered row-subsets $J = \{j, j'\}$ and column-subsets $K = \{k, k'\}$ (here j is fixed, but j', k and k' are arbitrary). Note that this path begins and ends at the fixed row j (i.e., this path is a loop) . For each $p[J; K]$ we calculate the product of the matrix entries along that path:

$$\pi(p[\{j, j'\}; \{k, k'\}]) = \prod_{s=0}^1 D_{j^{[s]}k^{[s]}} D_{j^{[s+1]}k^{[s+1]}}$$

where $j^{[s]}$ corresponds to the j -index with s ‘primes’ atop it, adopting the convention that $j^{[2]} = j'' = j$. As expected from our loop-scores, these simple $(1 + 1) \times 2$ -step paths produce a product of the form:

$$\pi(p[\{j, j'\}; \{k, k'\}]) = \prod_{s=0}^1 D_{j^{[s]}k^{[s]}} D_{j^{[s+1]}k^{[s+1]}} = D_{jk} D_{j'k} D_{j'k'} D_{jk'}.$$

Note that the path $p[J; K]$ steps on each row and each column an even number of times. Consequently, if $p[J; K]$ were fully contained within a bicluster that was exactly rank-1, then π would always equal +1. More generally, within the context of the planted-bicluster problem, the product $\pi(p[J; K])$ is more likely to be +1 if the path $p[J; K]$ is fully contained within a planted bicluster. On the other hand, we expect that $\pi(p)$ will be equally likely to be +1 or -1 when the path p is not fully contained within a planted bicluster. Once we have the product $\pi(p)$ for each path $p[J; K]$, we aggregate the results to form the score for row- j , which we’ll denote by $[Z_{\text{ROW}}^1]_j := [Z_{\text{ROW}}]_j$:

$$[Z_{\text{ROW}}^1]_j = \sum_{J, K, j \text{ fixed}} \pi(p[J; K]), \text{ which is equal to } [(DD^\top)^{1+1}]_{jj}.$$

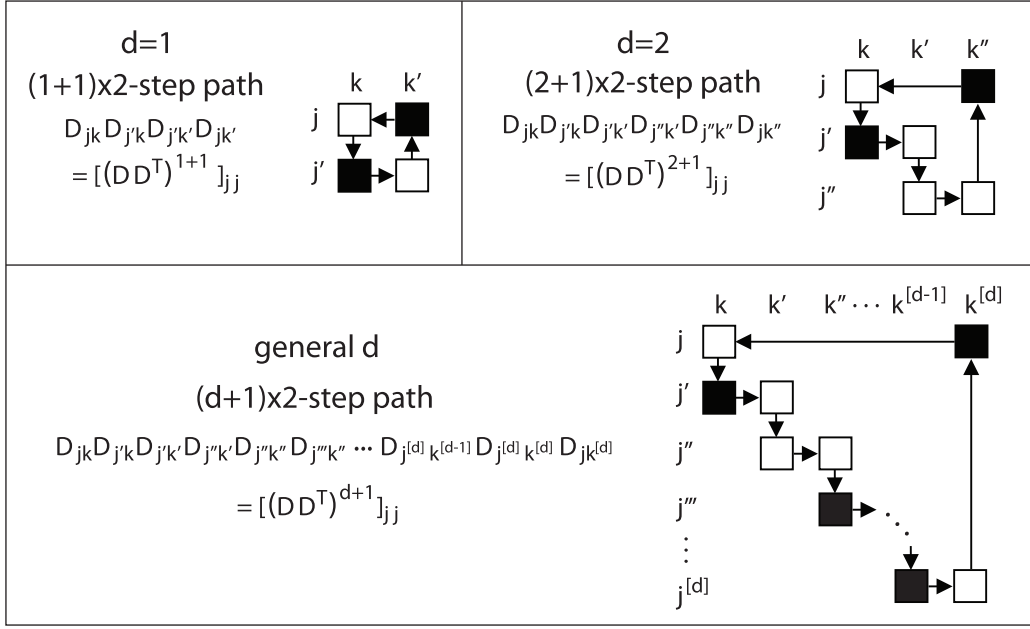


Figure 89: Illustration of paths through a bigraph. Our original row-scores $[Z_{\text{ROW}}^1]_j$ are computed by considering all $(1+1) \times 2 = 4$ -step paths through the bigraph of D which both start and end at row- j (see top-left). If such a 4-step path lies entirely within a rank-1 bicluster it will satisfy the following property: the product of the matrix-entries along the path will be equal to $+1$. The accumulation of such a product over all possible paths is given by $[Z_{\text{ROW}}^1]_j = [(DD^T)^{1+1}]_{jj}$. We may also consider other paths through the bigraph which start and end at row- j . On the top-right we show one such family of paths: namely $(2+1) \times 2 = 6$ -step paths. One can easily see that these 6-step paths also have the same property as the 4-step paths above: namely, the product of the matrix-entries along the path will be equal to $+1$ if the path lies entirely within a rank-1 bicluster. The accumulation of these products is given by $[(DD^T)^{2+1}]_{jj}$. On the bottom we generalize this notion to $(d+1) \times 2$ -step paths. Again, if such a path is contained within a rank-1 bicluster, the product of its entries will be $+1$. The accumulation of these products is given by $[(DD^T)^{d+1}]_{jj}$. As d grows, the accumulation $[(DD^T)^{d+1}]_{jj}$ converges (after appropriate rescaling) to the j^{th} -component of the dominant left-singular-vector of D . A similar statement can be made for paths which begin and end at column- k : as d increases the accumulation $[(D^T D)^{d+1}]_{kk}$ converges to the k^{th} -component of the dominant right-singular-vector of D .

We can generalize this approach by considering longer $(d+1) \times 2$ -step paths, rather than simple $(1+1) \times 2$ -step paths. Some examples are illustrated in Fig 89. These paths involve larger row-subsets $J = \{j, j', \dots, j^{[d]}\}$ and $K = \{k, k', \dots, k^{[d]}\}$. The product $\pi(p)$ is given by:

$$\pi(p[J; K]) = \prod_{s=0}^d D_{j^{[s]}k^{[s]}} D_{j^{[s+1]}k^{[s]}}$$

with the convention that $j^{[d+1]} = j$. As before, within the context of the planted-bicluster problem, $\pi(p)$ is more likely to be $+1$ whenever p is fully contained within a planted bicluster; if p is not contained within a planted bicluster then $\pi(p)$ is more likely to be drawn from the uniform distribution on $\{-1, +1\}$ (we'll quantify this more precisely below). The accumulation of such products, which we'll denote by $[Z_{\text{ROW}}^d]_j$ is now given by:

$$[Z_{\text{ROW}}^d]_j = \sum_{J, K, j \text{ fixed}} \pi(p[J; K]) = [(DD^T)^{d+1}]_{jj}$$

As d increases, the d -score $[Z_{\text{ROW}}^d]$ converges in direction to the entrywise-square of the dominant left-singular-vector: i.e., \vec{u}^2 . Similarly, the d -score $[Z_{\text{COL}}^d]$ converges in direction to the entrywise-square of the dominant right-singular vector: i.e., \vec{v}^2 . These scores are identical to the scores used in Step-2 of the simple spectral algorithm.

In the next two sections we argue that the size-threshold and noise-threshold associated with this simple spectral method can't be much better than the detection-thresholds associated with our loop-counting algorithm. Later, in section 22.3, we refine this argument to examine the behavior of the spectral method near the detection-threshold, explaining some of the results shown in Figs 29,30 and 31.

22.1 Size-threshold:

At this point we can use a simple counting argument to estimate how effective these scores $[Z_{\text{ROW}}^d]$ might be when it comes to detecting a planted bicluster. This counting argument is the first stage of the commonly-used ‘moment-method’ for analyzing the distribution of eigenvalues in a random matrix [15]. While we strongly encourage the interested reader to pursue this reference, we provide the basics of this argument here.

For now let’s focus on the noiseless limit, and say that the $m \times m$ planted bicluster B is perfectly rank-1, and involves the row-subset J_B and column-subset K_B . Let’s refer to the matrix-entries of D that are not in B as B^c . We’ll also assume that d is fixed and that M and m are very large, with $M \gg m \gg d$; we’ll pay attention to the change in behavior that occurs when $m \sim \sqrt{M}$. When we refer to quantity such as $O(M^{2d+1})$ we’ll ignore terms that do not grow like a power of m or M (e.g., we’ll ignore terms of order d , e^d , and so forth). To ease the notation, let’s define $\delta = (d - 1)/2$; we’ll later use δ to describe paths outside of B that contribute a $\pi(p)$ of $+1$.

Any $(d + 1) \times 2$ -step path $p[J; K]$ will step on up to $2(d + 1)$ different matrix-entries within D . For a given path p , some of its matrix-entries may be traversed only once, but others may be stepped on multiple times (i.e., they may be traversed with multiplicity greater than 1). Paths p fall into one of several categories, depending on the unique matrix-entries in p and their multiplicities. The essential idea behind our argument will be that, in order for a path p to contribute positively to the row-score, its matrix-entries from B^c must be multiplicity-2, but its matrix-entries from B need only be multiplicity-1.

Category-D0: This category includes paths p that start at some $\hat{j} \notin J_B$ and remain outside of B , while also having some matrix-entries with odd multiplicity. These paths will have a product $\pi(p[J; K])$ that is drawn from the uniform distribution on $\{-1, +1\}$. Given a fixed $\hat{j} \notin J_B$, there are $O(M^d \cdot M^{d+1})$ such paths. We arrive at this estimate by noting that there are $\sim M$ choices for each of the d rows $j^{[1]}, \dots, j^{[d]}$, and $\sim M$ choices for each of the $(d + 1)$ columns $k^{[0]}, k^{[1]}, \dots, k^{[d]}$. These Category-D0 paths each contribute a term of type $\pi(p) = \pm 1$ to the score $[Z_{\text{ROW}}^d]_j$. When averaging over all possible configurations of D , the total contribution of these Category-D0 paths will be 0. If these paths were all independent from one another, their total contribution would have a variance close to $M^d \cdot M^{d+1}$. Because the category-D0 paths are correlated, the variance of their total contribution is even larger, but is still is proportional to $M^d \cdot M^{d+1}$ multiplied by a prefactor of the form $[\text{constant}]^d$. The exact value of this prefactor is not as relevant as its magnitude, which grows with d but does not depend on M for fixed d . Thus, with respect to m and M , the variance of the total contribution of the Category-D0 paths is $O(M^d \cdot M^{d+1})$.

Category-D: This category includes paths p that start at some $\hat{j} \notin J_B$ and remain outside of B , while only having matrix-entries with even multiplicity. These paths will have a product $\pi(p) = +1$. For a given $\hat{j} \notin J_B$ there will be $O(M^\delta \cdot M^{\delta+1})$ paths available. We arrive at this estimate by noting that each matrix-entry must be traversed an even number of times (e.g., not once, but 0, 2, 4, \dots times): there can be at most δ unique rows amongst $j^{[1]}, \dots, j^{[d]}$, and at most $\delta + 1$ unique columns amongst $k^{[0]}, k^{[1]}, \dots, k^{[d]}$. A path p with the maximum number of unique rows and columns thus has $d + 1 = 2\delta + 2$ unique matrix-entries, starting out at $(\hat{j}, k^{[0]})$, including $(j^{[s+1]}, k^{[s]})$ and $(j^{[s+1]}, k^{[s+1]})$ for $s \in 0, \dots, \delta$, and potentially terminating at $(\hat{j}, k^{[\delta]})$. Such a path has multiplicity-2 for each of its matrix-entries. When averaging over all possible configurations of D , the total contribution of these Category-D paths will be $O(M^\delta \cdot M^{\delta+1})$; because $\delta \approx d/2$, this is bounded above in magnitude by the standard-deviation associated with the Category-D0 paths. Note that in this estimate we need not worry about other Category-D paths that involve fewer than δ unique rows and $\delta + 1$ unique columns, because the quantity of these other paths is an order of magnitude lower than the multiplicity-2-paths just discussed.

Category-B: This category includes paths p that start at some $j \in J_B$ and remain inside B . These paths will necessarily step on each row and each column an even number of times (by construction), and will have a product $\pi(p) = +1$. Given a fixed $j \in J_B$, there will be $O(m^d \cdot m^{d+1})$ such paths. We arrive at this estimate by noting that there are $\sim m$ choices for each of the d rows $j^{[1]}, \dots, j^{[d]}$, and $\sim m$ choices for each of the $(d + 1)$ columns $k^{[0]}, k^{[1]}, \dots, k^{[d]}$. The total contribution of these Category-B paths will be $O(m^d \cdot m^{d+1})$.

Category-C0: This category includes paths p that include some matrix-entries in B , as well as some matrix entries outside of B (i.e., in B^c). Additionally, the paths p must each have at least one of the following properties: either (i) some of the matrix-entries outside of B have an odd multiplicity, and/or (ii) the collection of matrix-entries inside of B does not include each row in J_B an even number of times, and/or (iii) the collection of matrix-entries inside of B does not include each column of K_B an even number of times. These paths produce a product $\pi(p)$ which is drawn uniformly from $\{-1, +1\}$. There are many such paths, but their total number is bounded by $O(m \cdot M^{2d})$ if the path p starts at some given $j \in J_B$ and by $O(m^2 \cdot M^{2d-1})$ if the path starts at some given $\hat{j} \notin J_B$, both of which are bounded by the number of Category-D0 paths. The reason there are so few of these paths is that at least one of the matrix-entries $(j^{[s]}, k^{[s]})$ or $(j^{[s+1]}, k^{[s]})$ must have a row drawn from J_B and a column drawn from K_B . Consequently, after averaging over all possible configurations of D , the total contribution of these paths to

the row-score will be 0, and the variance of this contribution will be bounded by the variance contributed by the Category-D0 paths.

Category-C: This category includes paths p that include some matrix-entries in B , as well as some matrix entries outside of B (i.e., in B^{\complement}). In addition, paths in Category-C must have all of the following properties: both (i) all of the matrix-entries in B^{\complement} must have an even multiplicity, and (ii) the collection of matrix-entries inside of B must include each row of J_B and each column of K_B an even number of times. These paths will produce a product $\pi(p) = +1$. If a Category-C path p has $y < d$ unique matrix-entries from B^{\complement} , then the maximum number of unique matrix-entries it can have in total is $y + x$, where $x = 2d + 2 - 2y \geq 4$ denotes the number of matrix-entries coming from B . This is because each of the y matrix-entries will have multiplicity at least 2. Assuming the path starts out inside B , the very first step of the path corresponds to one unique row-index from J_B and one unique column-index from K_B . Because the path itself can be thought of as a cycle, the final step of the path does not involve any unique row- or column-indices. Moreover, each step along the path corresponds to at most one additional unique row-index from J_B or $J_{B^{\complement}}$, or one additional unique column-index from K_B or $K_{B^{\complement}}$. Because $2y$ steps along the path must be associated with the multiplicity-2 matrix-entries from B^{\complement} , the entire path can include at most x unique indices from J_B and K_B , and y unique indices from $J_{B^{\complement}}$ and $K_{B^{\complement}}$. Combining these observations we see that, for fixed x, y such that $y < d$ and $x + 2y = 2d + 2 - 2y$, if we require that the path p begins at a given row-index $j \in J_B$, then there will be at most $O(m^{x-1}M^y)$ such paths available. On the other hand, if we require that the path p begins at a given row-index $\hat{j} \notin J_B$, then there will be only $O(m^x M^{y-1})$ such paths available. A Category-C path p with exactly $y = d$ unique matrix-entries from B^{\complement} will only have $x = 1$ unique matrix-entry from B (since p cannot satisfy property (ii) if $x = 2$). Such a path has at most 2 unique matrix-entries from J_B and K_B , and $y - 1$ unique matrix-entries from $J_{B^{\complement}}$ and $K_{B^{\complement}}$. Combining these observations we see that, for fixed $x = 1$ and $y = d$, if we insist that the path p begins at a given row-index $j \in J_B$, then there will be at most $O(m^1 M^{d-1})$ such paths available. On the other hand, if we insist that the path p begins at a given row-index $\hat{j} \notin J_B$, then there will be only $O(m^2 M^{d-2})$ such paths available. Note that, similar to the Category-D case, we can safely ignore Category-C paths that involve fewer than the maximum number of unique matrix-entries, as their total quantity will be an order of magnitude lower than the paths just discussed.

Given the categories above, we can fix d and analyze the behavior of the typical row-score $[Z_{\text{ROW}}^d]_j$ when $j \in J_B$, and compare it against the typical row-score $[Z_{\text{ROW}}^d]_{\hat{j}}$ when $\hat{j} \notin J_B$. We have two cases:

If $m = o(\sqrt{M})$: The most numerous Category-C paths will correspond to $y = d$ and $x = 1$, implying an average Category-C contribution of $O(m^1 M^{d-1})$ when $j \in J_B$, and $O(m^2 M^{d-2})$ when $\hat{j} \notin J_B$. The Category-C and Category-B contributions combined will be negligible compared to the standard-deviation of the contribution from the Category-D0 and Category-C0 paths. The row-score won't be significantly different for J_B than for $J_{B^{\complement}}$.

If $m \gtrsim \sqrt{M}$: The most numerous Category-C paths will correspond to $y = 1$ and $x = 2d$, implying an average Category-C contribution of $O(m^{2d-1}M)$ when $j \in J_B$, and $O(m^{2d})$ when $\hat{j} \notin J_B$. The sum of the average Category-C and Category-B contributions will be an order of magnitude higher for $j \in J_B$ than for $\hat{j} \notin J_B$. Moreover, these contributions dwarf the standard-deviation of the contribution from Category-D0 and Category-C0. The row-score will be significantly greater for J_B than for $J_{B^{\complement}}$.

In summary, if we fix d and take the limit as M and m go to infinity with $M \gg m \gg d$, then the typical row-score $[Z_{\text{ROW}}^d]_j$ be greater than the typical row-score $[Z_{\text{ROW}}^d]_{\hat{j}}$ only when m is on the same order as (or larger than) \sqrt{M} . This is the same size-threshold that pertains to our loop-score (i.e., $d = 1$), and suggests that this size-threshold applies for all d , as well as for the simple spectral method above.

22.2 Noise-threshold:

If we assume that B is not perfectly rank-1, but instead only approximately rank-1 with error ε , then the rough calculation above changes slightly. The Category-D0 and Category-C0 paths still contribute a variance on the order of $O(M^{2d+1})$ to the row-score $[Z_{\text{ROW}}^d]_{\hat{j}}$ when $\hat{j} \notin J_B$. On the other hand, the Category-B and Category-C paths now contribute a weaker signal to the row-scores $[Z_{\text{ROW}}^d]_j$ when $j \in J_B$. Looking at the Category-B paths for now, we see that there are still $O(m^{2d+1})$ paths within B that start and end at row- j . However, the noise within B will flip the sign of some of the entries of B , and not all these paths will contribute a product $\pi(p) = +1$. These paths now fall into two sub-categories:

Category-B+: This category includes paths p in B that step on any perturbed entries (i.e., those with sign flipped by the noise) an even number of times in total. These paths will produce a $\pi(p) = +1$.

Category-B-: This category includes paths p in B that step on any perturbed entries an odd number of times in total. These paths will produce a $\pi(p) = -1$.

The fraction of Category-B paths that are in Category-B+ is some number $G_{1,\varepsilon,m}^d$ in between $1/2$ and 1 (note that $G_{1,\varepsilon,m}^1$ is the same as the $g_{1,\varepsilon,m}$ we calculated in section 5). The average contribution of all the Category-B paths is now $O((2G_{1,\varepsilon,m}^d - 1) \cdot m^{2d+1})$, and we expect the average contribution of all the Category-C paths to suffer from a similar reduction. Given this reduction, we now expect the score $[Z_{\text{ROW}}^d]_j$ to be distinguishable from $[Z_{\text{ROW}}^d]_j$ only when $G_{1,\varepsilon,m}^d$ is significantly greater than $1/2$.

Recall that B is generated by first drawing the columns of a matrix A from a randomly-oriented gaussian-distribution ρ and then binarizing the result. The distribution ρ has a dominant principal-value equal to 1 , and subsequent singular-values equal to ε (i.e., A is rank-1 with error- ε). The distribution of singular-values of ρ implies that only a fraction ϕ of the total variance of ρ is associated with its first principal component. This fraction ϕ is $\sim 1/(1 + \varepsilon^2 m)$, and is only significantly greater than 0 when $g_{1,\varepsilon,m} > 1/2$ (i.e., when $\varepsilon\sqrt{m} \lesssim 1$). When $\varepsilon\sqrt{m} \gg 1$ and $g_{1,\varepsilon,m} \sim 1/2$, then the fraction ϕ will be close to 0 , and we expect $G_{1,\varepsilon,m}^d$ to be close to $1/2$ as well. These observations suggest that $G_{1,\varepsilon,m}^d$ cannot be much higher than $g_{1,\varepsilon,m}$, which would imply that – at best – the row-scores Z_{ROW}^d obey the same noise-transition as our original loop-score (i.e., $\varepsilon\sqrt{m} \lesssim 1$). Consequently, we expect the simple spectral method to obey the same noise-transition at $\varepsilon\sqrt{m} \lesssim 1$.

22.3 Behavior near the detection-threshold

Comparisons between our loop-score method and the simple spectral method can be seen in Fig 29, as well as Figs 30 and 31. Note that our loop-score algorithm outperforms the spectral method in many circumstances, especially when the planted-bicluster has a rank $l > 1$. It turns out that this phenomenon is related to the behavior of these two methods *near* the detection-threshold. More specifically, all our analysis so far has focused on distinguishing the $m = o(\sqrt{M})$ situation from the $m \gtrsim \sqrt{M}$ situation. To understand why loop-counting outperforms the simple spectral method we need to focus on the situation where m is on the *same* order as \sqrt{M} . In other words, we should assume that m/\sqrt{M} is some constant close to 1 .

For illustration, let's consider a planted-bicluster problem where the matrix D is of size $M = 768$ and the embedded bicluster B is of rank $l = 2$ and size $m = 39$ with $\varepsilon = 0$. These parameters are close to the $M = 1024$, $\log_M(m) = 0.55$ position in Fig 30 (i.e., a parameter regime where loop-counting outperforms the spectral method). For this choice of parameters m is quite close to \sqrt{M} : i.e., $m/\sqrt{M} \approx 1.4$. Hence, the phenomenon we see for these parameters will be indicative of what happens near the detection-threshold. In this example we'll focus on the row-scores, although our argument applies equally well to the column-scores.

Given this planted-bicluster problem, we construct multiple trials; for each trial we calculate the d -scores $[Z_{\text{ROW}}^d]$ for various d . In this situation we have no controls; there is no convenience associated with correcting for collapsed-loops, and so we'll use the formulae for the d -scores in section 22; these scores will always be positive. Going forward, it will be more convenient to refer to these scores by their square-roots, which we'll denote by $[z_{\text{ROW}}^d]$. In general, each individual $[z_{\text{ROW}}^d]_j$ will be drawn from one of two distinct distributions, depending on whether or not j is in J_B or J_{B^c} . We'll denote these two distributions by ζ_B^d and $\zeta_{B^c}^d$, respectively. These two distributions will depend on d , in addition to the parameters M , m , l and ε . Ultimately, we'll be interested how these two distributions change as a function of d , for m close to \sqrt{M} .

For the particular parameters in this example, we illustrate the distributions of the square-roots of the d -scores in Fig 90A. Each of these subplots shows the distributions ζ_B^d and $\zeta_{B^c}^d$ for a particular d (obtained by trial-averaging). The distribution ζ_B^d is shown in red, while the distribution $\zeta_{B^c}^d$ is shown in blue.

Note that, when $d = 1$, we recover our original loop-scores. Our analysis from section 5 applies almost perfectly; the loop-scores $[Z_{\text{ROW}}^1]$ are distributed roughly like gaussians with means much greater than 0 (recall that these scores include the contribution of collapsed-loops), and hence so too are their square-roots $[z_{\text{ROW}}^1]$. Consequently, the distributions ζ_B^1 and $\zeta_{B^c}^1$ are also roughly gaussian. Moreover, when $d = 1$ we can use our expression for $g_{2,\varepsilon,m}$ to calculate the shift in the means of these two gaussian distributions.

As d increases, the two square-root-score-distributions ζ_B^d and $\zeta_{B^c}^d$ change shape; for this choice of parameters they become more and more lopsided. When d is very large the scores $[Z_{\text{ROW}}^\infty]$ converge to the scores of the simple spectral method: namely, the entrywise-squares of the dominant left-singular-vector of D (i.e., \vec{u}^2). Consequently, the square-roots of the scores (i.e., $[z_{\text{ROW}}^\infty]_j$) converge to the absolute-values $|\vec{u}_j|$. For this choice of parameters, both ζ_B^∞ and $\zeta_{B^c}^\infty$ are rather lopsided (with the latter equivalent to half a gaussian-distribution). It is the lopsidedness of ζ_B^∞ that compromises the performance of the spectral method, and we'll discuss this more below.

One way to quantify the difference between the square-root-score-distributions is to measure the trial-averaged AUC obtained by comparing ζ_B^d to $\zeta_{B^c}^d$ (shown in each panel). Note that the AUC is somewhat high for the original loop-scores

(i.e., $d = 1$), increases as a function of d up until a point, and then decreases as d gets larger and larger. The behavior of the trial-averaged Auc as a function of d is shown in Fig 90B (thick black line).

If we were to calculate our scores only once and use a linear threshold to distinguish rows inside B from rows outside B , then the Auc we just described would dictate the success of our methodology. However, our algorithm is iterative; we typically remove a small fraction of the rows and columns of D with each iteration. As mentioned in Fig 32, this fraction is usually $\lesssim 5\%$. Our hope is that – by removing the rows and columns associated with the lowest scores – we preferentially remove rows and columns from outside B , leaving the rows and columns of B mostly untouched. This strategy will be successful only if the lowest scores are more likely to come from outside B than from inside B . Thus, with regards to our iterative framework, a more revealing measure of the quality of the various d -score-distributions is the Auc obtained by comparing the bottom 5^{th} -percentile of row-scores in B with the bottom 5^{th} -percentile of row-scores outside B ⁸. This trial-averaged metric – referred to as ‘A05’ – is shown in Fig 90C (thick black line).

Note that both the Auc and the A05 are markedly higher when d is small than when d is very large. This fact is a direct result of the lopsided nature of ζ_B^d for large d , and is the reason why our loop-counting algorithm outperforms the simple spectral method in this example. We now turn to a discussion of this lopsidedness.

Recall that the square-root-scores $[z_{\text{ROW}}^\infty]$ associated with the simple spectral method are drawn from ζ_B^∞ and $\zeta_{B^c}^\infty$, which describe the distribution of the coefficients of the vector $|\vec{u}|$. The distribution of these coefficients can in turn be determined analytically by using the techniques of [M32].

Before we engage in this analysis, we make the following observations:

1. The singular-vector \vec{u} is equal to the dominant eigenvector of the covariance-matrix DD^\top/M .
2. The covariance-matrix DD^\top/M can be approximated by the sum of (a) a random covariance matrix W and (b) a symmetric matrix C with the following block structure: C is mostly 0, except for an $m \times m$ block corresponding to J_B with off-diagonal entries roughly proportional to $\pm m/(2M\sqrt{\max(1/2, l-1)})$. See section 18 for a derivation. In this particular example, because $l = 2$, the nonzero off-diagonal entries of C are close to $\pm m/(2M)$.
3. The block matrix C is rank-1 with a dominant eigenvalue roughly equal to $\sim m^2/(2M\sqrt{\max(1/2, l-1)})$.
4. The dominant eigenvector $\vec{\psi}$ of C shares the block-structure of C : The m entries of ψ corresponding to J_B are each equal to $\pm 1/\sqrt{m}$, while the other $M - m$ entries of ψ are 0.

At this point we can adopt the approach of [M32], which observes that the dominant eigenvector \vec{u} of DD^\top/M obeys an implicit equation involving the entries of $\vec{\psi}$ and \vec{u} itself, as well as the empirical spectral distribution of W (which is, in this case, given by a Marchenko-Pastur law with aspect-ratio 1.0). Following this train of logic, one can show that – depending on m – the vector \vec{u} may or may not point in the direction of $\vec{\psi}$. More specifically, if we define ω to be the angle between \vec{u} and $\vec{\psi}$, one can show that $\cos(\omega)^2$ can be approximated by:

$$\cos(\omega)^2 \approx \max \left\{ 0, 1 - \frac{M}{m^2} 2\sqrt{\max(1/2, l-1)} \right\}, \quad (3)$$

with this approximation holding with high probability in the limit as $m, M \rightarrow \infty$ (with m/\sqrt{M} fixed). Note also that, aside from the restriction that $\vec{u} \cdot \vec{\psi} = \cos(\omega)$, the vector \vec{u} is drawn from a uniform distribution on the sphere in \mathbb{R}^M .

We illustrate this phenomenon with a numerical experiment in Fig 90D. For this experiment we fix $M = 768$, $l = 2$ and $\varepsilon = 0$. We then vary m ; for each m we construct multiple trials and measure ω . The trial-averaged $\cos(\omega)^2$ is plotted in black, while the analytical approximation to $\cos(\omega)^2$ from Eq. 3 is shown in grey.

The value of ω can be used to determine the distribution of \vec{u} on \mathbb{R}^M : \vec{u} is drawn uniformly from the set of vectors that are of unit norm and have $\vec{\psi}$ -component equal to $\cos(\omega)$ (i.e., \vec{u} is drawn from a slightly smaller sphere of dimension $M - 1$ centered at $\cos(\omega) \cdot \vec{\psi}$ and normal to $\vec{\psi}$). This distribution on \vec{u} in turn determines the two distributions ζ_B^∞ and $\zeta_{B^c}^\infty$.

In general, the distribution $\zeta_{B^c}^\infty$ will be a half-gaussian; these square-root-scores are associated with the component of \vec{u} that is randomly oriented. On the other hand, the distribution ζ_B^∞ may or may not look like $\zeta_{B^c}^\infty$. If m is too low, then $\cos(\omega)$ will be essentially 0, and the distribution ζ_B^∞ will look essentially like $\zeta_{B^c}^\infty$. But, if m is large then $\cos(\omega)$ will be large, and the distribution ζ_B^∞ will be very different from $\zeta_{B^c}^\infty$.

Coming back to our example, we have $m = 39$ and $m/\sqrt{M} \approx 1.4$; the trial-averaged $\cos(\omega)$ is approximately 0.19. If we use this value of ω to determine ζ_B^∞ and $\zeta_{B^c}^\infty$ as described above (i.e., drawing \vec{u} from the appropriate sphere), we obtain the grey lines shown in the background of the spectral square-root-scores in Fig 90A. Note that this approximation is very close to the trial-averaged distributions shown in blue and red.

Taking a step back from our example we can immediately see that, if we were to increase m , then $\cos(\omega)$ would increase as well, and \vec{u} would align more closely with $\vec{\psi}$. As \vec{u} and $\vec{\psi}$ become more aligned, the distribution ζ_B^∞ moves to the right,

⁸for this example, this A05 is obtained by comparing the bottom 2 scores in B with the bottom 36 scores outside B

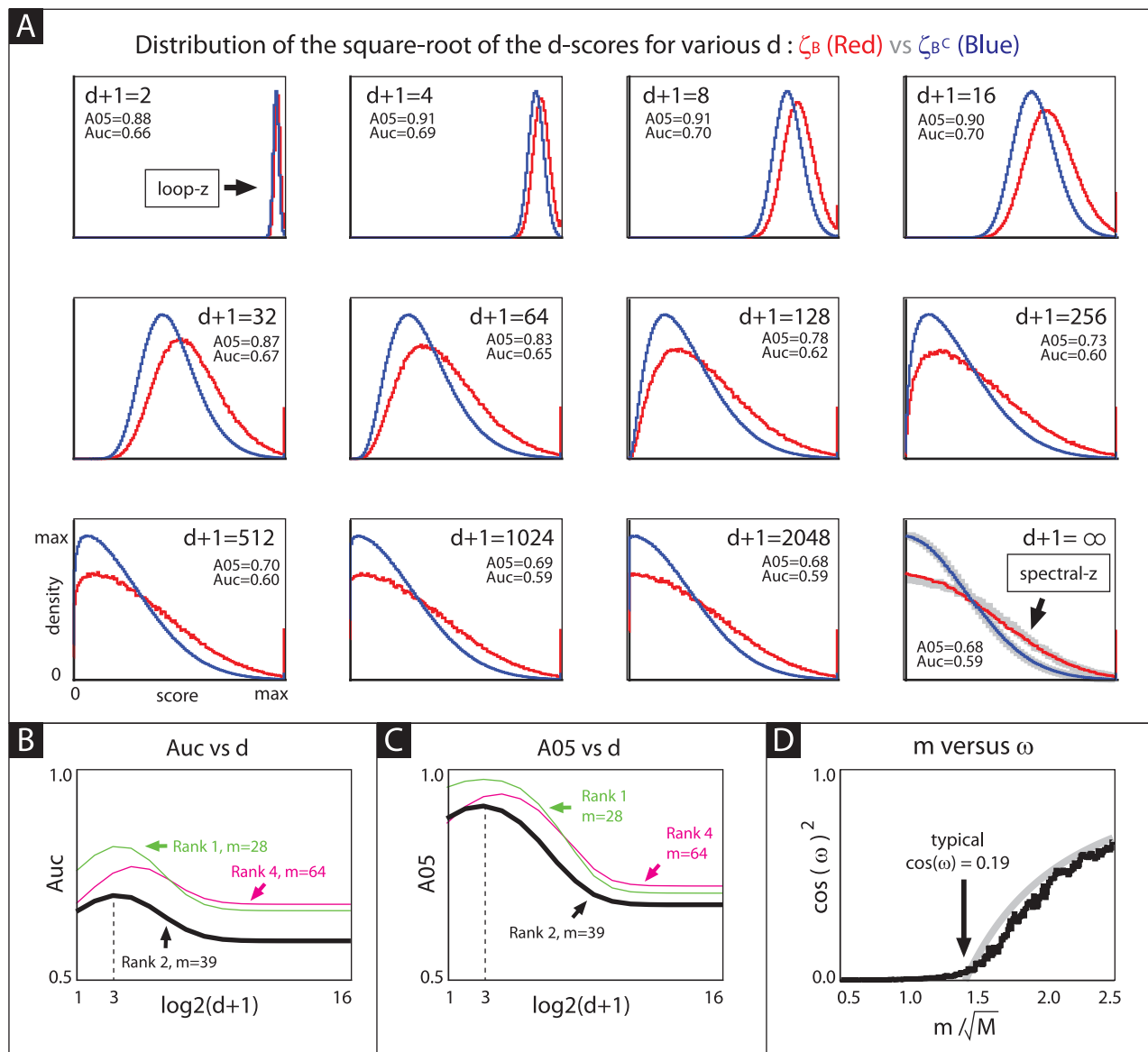


Figure 90: Illustration of square-roots of the d -scores for the planted-bicluster problem with $M = 768$, $l = 2$, $m = 39$, and $\varepsilon = 0$. Panel-A shows the trial-averaged distribution of the square-roots of the d -scores both within (red) and outside (blue) the planted bicluster B . Panel-B shows the trial-averaged Auc obtained by comparing all scores in B to all scores outside B (thick black line). Panel-C shows the trial-averaged Auc obtained by comparing the bottom 5th percentile of scores in B to the bottom 5th-percentile of scores outside B (thick black line). Panel-D shows the relationship between m/\sqrt{M} and the typical angle ω made by the dominant left singular-vector of the data-matrix D and the ‘true’ singular-vector associated with B .

Distribution of square-root of spectral-scores for various ω

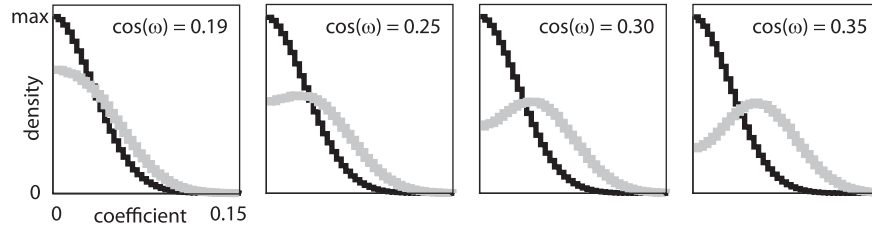


Figure 91: The spectral square-root-score-distributions associated with various ω , given $M = 768$, $l = 2$ and $\varepsilon = 0$. The distribution $\zeta_{B^c}^\infty$ is shown in black, and the distribution ζ_B^∞ is shown in grey. See text for full description.

as shown in Fig 91. When $\cos(\omega)$ is sufficiently large, then ζ_B^∞ is very well distinguished from $\zeta_{B^c}^\infty$, and the simple spectral method performs well.

At this point we can see that the value of $\cos(\omega)$, and hence the performance of the spectral method, is a function of the bicluster size m , as well as the bicluster rank l and error ε . For many choices of parameters, such as those shown in the example above, $\cos(\omega)$ will be small and the loop-scores will be more useful than the spectral-scores (at least within the context of our methodology). This phenomenon is particularly pronounced when the rank l of the planted-bicluster is greater than 1; in this situation $\cos(\omega)$ increases only gradually as a function of m/\sqrt{M} . On the other hand, when $l = 1$ then $\cos(\omega)$ increases rather quickly as a function of m/\sqrt{M} , and this phenomenon is not as pronounced. Indeed, for some choices of parameters (e.g., if $l = 1$ and m and ε are sufficiently large) then it is possible for the spectral-scores to be more useful than our loop-scores (see, e.g., label ‘E’ near the detection boundary in Fig 29C).

Overall, we believe that the loop-scores are a better choice than the spectral-scores. Not only are the loop-scores typically more useful with regards to the planted-bicluster problem (as discussed above), but the loop-scores are also less sensitive to certain kinds of spectral noise.

We close by highlighting two more planted-bicluster problems identical to the example discussed above, but with different values for m and l . The first involves $m = 28$, $l = 1$, and the second involves $m = 64$, $l = 4$. The behavior of the distributions ζ_B^∞ and $\zeta_{B^c}^\infty$ for these problems is qualitatively similar to the behavior shown in Fig 90A, but the separation between the two distributions depends on the details. The trial-averaged Auc and A05 for these two problems is shown in Figs 90B,C using magenta and green lines in the background. Note that the general trend observed in the $m = 39$, $l = 2$ case still holds: the spectral scores are less useful than the loop-scores. Also note that the peak values for the Auc and the A05 occur at different values of d , depending on the problem parameters.

22.4 A remark regarding message-passing algorithms

As readily observed in the examples above, the difference between the two distributions ζ_B^d and $\zeta_{B^c}^d$ depends on d . We’ve quantified this difference using the metrics Auc and A05 (which are easy to measure in a numerical experiment), but one could imagine a more comprehensive measurement of the information \mathcal{D}_{KL} (i.e., relative-entropy) between these two distributions. Given what we’ve seen so far, it seems likely that the ‘best’ d -score for any particular planted-bicluster problem is neither the loop-score (i.e., $d = 1$) nor the spectral-score (i.e., $d = \infty$), but rather somewhere in the middle.

Taking this reasoning a step further, it is tempting to try and construct some nonlinear function of the ‘best’ d -score which allows for the ‘optimal’ classification of J_B versus J_{B^c} . Going even further, one might imagine a score that is itself constructed nonlinearly; instead of building the d -score $[Z_{\text{ROW}}^d]$ by applying power-iteration to the covariance-matrix DD^T , one might try and build an even better score by applying a nonlinearity in between each stage of the power-iteration.

The message-passing algorithms of [M33], [M19], and [M41] proceed along these lines. Generally speaking, these algorithms choose an appropriate nonlinearity to apply between stages of a ‘message-passing procedure’ similar to power-iteration. By choosing this nonlinearity carefully, these methods can significantly reduce their detection-thresholds for a variety of problems very similar to our planted-bicluster problem (such as, e.g., the planted-clique problem). Given the success of the message-passing algorithms of [M33], [M19], and [M41], it seems certain that there exists a message-passing algorithm that outperforms our loop-counting algorithm when applied to the planted-bicluster problem; we fully intend to pursue this line of research in the future.

We remark that, when developing a message-passing algorithm, it is critical to ensure that the algorithm performs robustly even outside the framework of the planted-bicluster problem. That is to say, it is important to ensure that the message-passing procedure (and the choice of nonlinearities) is not too specific to the assumptions of the planted-bicluster problem, and that the algorithm generalizes to the kinds of scenarios seen in real data sets.

For example, we’ve already seen that the ‘best’ d -score for the planted-bicluster problem depends strongly on the problem parameters. More specifically, the ‘best’ d depends not only on the size and spectrum of the planted bicluster

B , but also on the structure of the rest of D . For example, if the spectrum of D were to decay, or if D were to contain a heterogeneous collection of other biclusters, then the distributions ζ_B^d and $\zeta_{B^c}^d$ will reflect this structure. Simply put: the optimal choice of d for one instance of the planted-bicluster problem may no longer be optimal once the parameters change, or when things get a little messier. When analyzing real data-sets we often have no knowledge of the size or spectrum of any hidden structures, and it is usually difficult to characterize the noise accurately. A good message-passing algorithm should perform well even in the absence of this information (perhaps adaptively tailoring its message-passing procedure as it analyzes the data).

22.5 Accounting for controls:

Much like our loop-counting methods, the simple spectral method described above can also be modified to account for several aspects of experimental design. In this subsection we'll elaborate on these techniques, discussing the control-correction in detail. In fact, as we'll see below, there are several ways of correcting the spectral-method which are quite similar to the corrections we've used for our biclustering methods (e.g., in sections 8 and 15.2).

Recall that, within Step-1 of the simple spectral method defined in section 22, we used \vec{u} and \vec{v} to refer to the first columns of U and V (respectively), where $D = U\Sigma V^\top$ is a singular-value-decomposition. Note that \vec{u} is the dominant eigenvector of DD^\top , and \vec{v} is the dominant eigenvector of $D^\top D$. We can view $\vec{u} \in \mathbb{R}^{M_D}$ as the direction of greatest variance for the columns of D , and $\vec{v} \in \mathbb{R}^N$ as the direction of greatest variance for the rows of D . We'll use this perspective below to generalize the simple spectral method.

To begin with, let's assume that – in addition to the $M_D \times N$ case-matrix D – we also have an $M_X \times N$ control-matrix X (as in section 8). Our goal will now be to find a direction \vec{v} in \mathbb{R}^N that best serves to separate these two categories. Given this goal, it is reasonable to consider different measures of separation 'S'.

One of the simplest such measures tries to find a \vec{v} such that the projections ' $D\vec{v}$ ' of the case-patients onto \vec{v} are, collectively, as large (in magnitude) as possible, whereas the projections ' $X\vec{v}$ ' of the control-patients onto \vec{v} are as small as possible. This measure is given by:

$$S_1^{\text{col}}(\vec{v}; D, X) := \frac{1}{2} \frac{1}{NM_D} \sum_{j \in D} (D_{j:} \cdot \vec{v})^2 - \frac{1}{2} \frac{1}{NM_X} \sum_{j' \in X} (X_{j':} \cdot \vec{v})^2,$$

which aggregates the squares of each projected variable, adding $|D\vec{v}|^2$ and subtracting $|X\vec{v}|^2$. In this expression the vectors $D_{j:}$ and $X_{j:}$ are the j^{th} rows of the D and X matrices, respectively. If we constrain the norm of \vec{v} and maximize $S_1^{\text{col}}(\vec{v})$ we find that the optimal \vec{v} is given by the dominant eigenvector of:

$$M_1^{\text{col}} := \frac{1}{NM_D} D^\top D - \frac{1}{NM_X} X^\top X. \quad (4)$$

Using this vector as \vec{v} in Step-1 of the simple spectral method (instead of merely using the dominant eigenvector of $D^\top D$) allows us to correct for certain features of the control submatrix. Specifically, such a \vec{v} would highlight columns k that participated in biclusters within D , but do not participate in any biclusters within X .

While such a control-correction is similar in many ways to our control-correction in section 8, it is not equivalent! To explain the difference, consider two cases of the planted-bicluster problem involving D and X of equal size (i.e., $M_D = M_X$). In case-1, we'll implant a $2m \times n$ bicluster $B_{[D;X]}$, with m rows in D and m rows in X . In case-2 we'll implant one $m \times n$ bicluster B_D in D , and another $m \times n$ bicluster B_X in X , with B_D and B_X occupying the same columns. Note that both case-1 and case-2 involve two $m \times n$ biclusters; one in D and one in X , both occupying the same columns. In case-1, both biclusters have the same low-rank structure (i.e., both are part of $B_{[D;X]}$). On the other hand, in case-2, the two biclusters have different (arbitrary) low-rank structure.

The control-corrected loop-counting algorithm described in section 8 will (correctly) ignore the case-specific component of $B_{[D;X]}$ in case-1, because $B_{[D;X]}$ straddles both D and X ; it will also (correctly) find the bicluster B_D in case-2, because this bicluster is case-specific and possesses a low-rank structure which is distinct from the corresponding columns of X . On the other hand, the control-corrected spectral method derived from S_1^{col} and Eq. 4 will ignore both the case-specific component of $B_{[D;X]}$ in case-1, as well as the case-specific bicluster B_D in case-2. This is because the term $-X^\top X$ in M_1^{col} will penalize any column k in \vec{v} whenever there is *any* low-rank structure within X involving k , regardless of whether or not that low-rank structure is the *same* in D as it is in X .

To rectify this deficiency, we could instead try to find a \vec{v} such that the projection $D\vec{v}$ correlates with each column of D only when the projection $X\vec{v}$ is uncorrelated with the corresponding column of X (or vice-versa). This measure is given by:

$$S_2^{\text{col}}(\vec{v}; D, X) := \frac{1}{2} \frac{1}{N} \sum_k \left(\frac{1}{M_D} D_{k:}^\top D \cdot \vec{v} - \frac{1}{M_X} X_{k:}^\top X \cdot \vec{v} \right)^2.$$

If, as before, we constrain the norm of \vec{v} and maximize $S_2^{\text{col}}(\vec{v})$, we find that the optimal \vec{v} is given by the dominant eigenvector of:

$$M_2^{\text{col}} := \frac{1}{N} \left[\frac{1}{M_D M_D} D^\top D D^\top D - \frac{1}{M_D M_X} D^\top D X^\top X - \frac{1}{M_X M_D} X^\top X D^\top D + \frac{1}{M_X M_X} X^\top X X^\top X \right]. \quad (5)$$

Each term of M_2^{col} can be interpreted as affecting \vec{v} in a different way. The first term $D^\top D D^\top D$ rewards columns that participate in biclusters within D . The two middle-terms $D^\top D X^\top X$ and $X^\top X D^\top D$ penalize columns that participate in nonspecific biclusters that exhibit the same low-rank structure across both D and X . The fourth term $X^\top X X^\top X$ rewards columns that participate in biclusters within X .

In the above discussion we focused on finding \vec{v} ; Now we turn to finding \vec{u} . Taking inspiration from S_1^{col} and S_2^{col} , we can devise a strategy for finding $\vec{u} \in \mathbb{R}^{M_D}$: we can try and ensure that the projection $D^\top \vec{u}$ correlates strongly (either positively or negatively) with each row of D , while not correlating as strongly with the rows of X . This measure is given by:

$$S_2^{\text{row}}(\vec{u}; D, X) := \frac{1}{2} \frac{1}{N^2 M_D} \sum_{j \in D} (D_{j \cdot} D^\top \cdot \vec{u})^2 - \frac{1}{2} \frac{1}{N^2 M_X} \sum_{j' \in X} (X_{j' \cdot} D^\top \cdot \vec{u})^2.$$

As before, we constrain the norm of \vec{u} and maximize $S_2^{\text{row}}(\vec{u})$; the optimal \vec{u} is given by the dominant eigenvector of:

$$M_2^{\text{row}} := \frac{1}{N^2 M_D} D D^\top D D^\top - \frac{1}{N^2 M_X} D X^\top X D^\top. \quad (6)$$

Each term of M_2^{row} affects \vec{u} in a different way; the first term $D D^\top D D^\top$ rewards rows that participate in biclusters within D , whereas the second term $D X^\top X D^\top$ penalizes rows that participate in nonspecific biclusters that exhibit the same low-rank structure across both D and X .

Note that the control-correction we use in our loop-counting algorithm (see section 8) is quite closely related to maximizing S_2^{row} and S_2^{col} . Indeed, our control-corrected row-score essentially involves calculating M_2^{row} , and our control-corrected column-score essentially involves calculating the first term of M_2^{col} , as well as one of the two middle terms. We don't use the fourth term of M_2^{col} , as our goal in section 8 is to find case-specific biclusters, not to avoid control-specific biclusters (unless, of course, they are part of non-specific biclusters, which will be penalized by either of the middle-terms).

Other measures of case-control separation: Note that S_1^{col} , S_2^{col} and S_2^{row} are only a few of the many possible measures of 'separation'. For example, we could try and ensure that the average value of $D\vec{v}$ was different than the average value of $X\vec{v}$, defining a separation:

$$S_3^{\text{col}} := \frac{1}{2} \left(\frac{1}{M_D} \sum_{j \in D} D_{j \cdot} \vec{v} - \frac{1}{M_X} \sum_{j' \in X} X_{j' \cdot} \vec{v} \right)^2,$$

which has an optimal \vec{v} given by the dominant eigenvector of:

$$M_3^{\text{col}} := \left(\frac{1}{M_D} D^\top \mathbf{1}_D - \frac{1}{M_X} X^\top \mathbf{1}_X \right) \left(\frac{1}{M_D} \mathbf{1}_D^\top D - \frac{1}{M_X} \mathbf{1}_X^\top X \right),$$

where $\mathbf{1}_D$ is the $M_D \times 1$ vector of all ones, and $\mathbf{1}_X$ is the $M_X \times 1$ vector of all ones.

We could also try and ensure a large magnitude difference between the projected case-rows $D_{j \cdot} \vec{v}$ and control-rows $X_{j' \cdot} \vec{v}$, defining:

$$S_4^{\text{col}} := \frac{1}{2} \frac{1}{M_D M_X} \sum_{j \in D, j' \in X} (D_{j \cdot} \vec{v} - X_{j' \cdot} \vec{v})^2,$$

which has an optimal \vec{v} given by the dominant eigenvector of:

$$M_4^{\text{col}} := \frac{1}{M_D} D^\top \left(D - \frac{1}{M_X} \mathbf{1}_D \mathbf{1}_X^\top X \right) + \frac{1}{M_X} X^\top \left(X - \frac{1}{M_D} \mathbf{1}_X \mathbf{1}_D^\top D \right).$$

Each of these different strategies has its advantages and disadvantages; focusing on certain kinds of structures and ignoring others. While no particular strategy is 'best' overall, there will be certain applications where some strategies outperform others. Note that, within the context of biclustering, the control-correction we use in our 'half-loop' algorithm (see section 15.2) is somewhat related to maximizing S_4^{col} for \vec{v} and its analog S_4^{row} for \vec{u} .

Continuous case-control status: Finally, we discuss how to extend the methodology above to deal with continuous case-control status. Such a situation may arise when the categorization of patients into 'cases' and 'controls' isn't all that

clear-cut. For example, we might imagine that our data includes N measurements taken across M total patients, each of which has a different blood-pressure $\kappa_j \in \mathbb{R}$ (i.e., an associated continuous clinical variable).

If we are interested in finding a direction $\vec{v} \in \mathbb{R}^N$ which separates high-blood-pressure patients from low-blood-pressure patients, we could pick some threshold $\bar{\kappa}$ and divide the patients into case- and control-groups based on whether their blood-pressure κ_j was greater or less than $\bar{\kappa}$. This simple thresholding might be sufficient in some cases, but relies on an (arbitrary) threshold $\bar{\kappa}$.

Alternatively, we can treat κ_j as a fully continuous variable, and try to find a $\vec{v} \in \mathbb{R}^N$ that maximizes the following separation:

$$S_5^{\text{col}} := \frac{1}{2} \sum_{j,j'} (D_{j,:}\vec{v} - D_{j',:}\vec{v})^2 (\kappa_j - \kappa_{j'})^2. \quad (7)$$

Note that if $\bar{\kappa}$ is indeed binary (e.g., each κ_j takes on values of 0 or 1), then this S_5^{col} is essentially the same thing as the S_4^{col} above. If we go about our usual business (constraining the norm of \vec{v} and optimizing S_5^{col}), then we find that \vec{v} is an eigenvector of:

$$M_5^{\text{col}} := MD^TK^2D - (1^T\bar{\kappa})2D^TKD + \left(1^T\bar{\kappa}^2\right)D^TD - D^TK^211^TD + 2D^TK11^TKD - D^T11^TK^2D, \quad (8)$$

where K is the diagonal matrix with $K_{jj} = \kappa_j$, and 1 is the $M \times 1$ vector of all ones.

It is also possible that each patient has a multi-dimensional continuous-category $\vec{\kappa}_j \in \mathbb{R}^t$ (e.g., a vector of t -values including blood-pressure, weight, etc). We can immediately extend the methodology above using a slightly modified separation:

$$S_6^{\text{col}} := \frac{1}{2} \sum_{j,j',t} (D_{j,:}\vec{v} - D_{j',:}\vec{v})^2 (\kappa_{j,t} - \kappa_{j',t})^2, \quad (9)$$

where κ_{jt} is the t -component of $\vec{\kappa}_j$ for the j^{th} -data-point. Optimizing this objective involves finding the dominant eigenvector of :

$$M_6^{\text{col}} = \sum_t MD^TK_t^2D - 2(1^T\bar{\kappa}_t)D^TK_tD + \left(1^T\bar{\kappa}_t^2\right)D^TD - D^TK_t^211^TD + 2D^TK_t11^TK_tD - D^T11^TK_t^2D, \quad (10)$$

where $\kappa_t \in \mathbb{R}^M$ is the vector formed from the κ_{jt} for various j , and K_t is the diagonal matrix such that $[K_t]_{jj} = \kappa_{jt}$.

As we've seen earlier, there are control-corrected spectral methods that are quite similar to our control-corrected biclustering algorithm (compare, e.g., Eqs. 5,6 with the loop-score of section 8). By analogy, it seems likely that Eqs. 8,10 can be used to extend our biclustering methodology to account for continuous case-control status. We intend to pursue this line of research in the future.

References

- [1] Cunha S.I., Bocci M., Lovrot J., Eleftheriou N., Roswall P., Cordero E., Lindstrom L., Bartoschek M., Haller B.K., Pearsall R.S., Mulivor A.W., Kumar R., Larsson C., Bergh J. and Pietras K. Endothelial ALK1 Is a Therapeutic Target to Block Metastatic Dissemination of Breast Cancer. *Cancer Res.* 75(12):2445-56 (2015).
- [2] Smith J.J., Dean N.G., Wu F., Merchant N.B., Zhang B., Jiang A., Lu P., Johnson J.C., Schmidt C., Bailer C.E., Eschrich S., Kis C., Levy S., Washington M.K., Heslin M.J., Coffey R.J., Yeatman T.J., Shyr Y. and Beauchamp R.D. Experimentally Derived Metastasis Gene Expression Profile Predicts Recurrence and Death in Patients with Colon Cancer. *Gastroenterology.* 138:958-968 (2009).
- [3] Eschrich S., Yang I., Bloom G., Kwong K.Y., Boulware D., Cantor A., Coppola D., Kruhoffer M., Aaltonen L., Orntoft T.F., Quackenbush J. and Yeatman T.J. Molecular staging for survival prediction of colorectal cancer patients. *Journal of Clinical Oncology*, 23:15. (2005)
- [4] Mulcahy H.E. and O'Donoghue D.P. Duration of colorectal cancer symptoms and survival: the effect of confounding clinical and pathological variables. *Eur J Cancer.* 33(9):146177. (1997)
- [5] <https://www.ncbi.nlm.nih.gov/books/NBK22218>
- [6] Abstracts of the XXIIIrd World Congress of Psychiatric Genetics (WCPG): Final symposia and plenary abstracts. *European Neuropsychopharmacology.* (2015).
- [7] Sklar P., Ripke S., Scott L.J., et al. Large-scale genome-wide association analysis of bipolar disorder identifies a new susceptibility locus near ODZ4. *Nature Genetics.* 43(10):977-983. doi:10.1038/ng.943. (2011)

- [8] Stahl E., Forstner A., McQuillin A., Ripke S., the Bipolar Disorders Working Group of the PGC, Ophoff R., Scott L., Cichon S., Andreassen O., Sklar P., Kelsoe J. and Breen G., Genomewide association study identifies 30 loci associated with bipolar disorder. *bioRxiv*. doi = 10.1101/173062, <http://www.biorxiv.org/content/early/2017/08/08/173062.1.full.pdf>, (2017)
some affinity propagation method used to cluster columns and rows ; In this case biclusters are ‘rank-0’ ;
- [9] Kaiser S., Santamaria R., Khamiakova T., Sill M., Theron R., Quintales L., Leisch F. and De Troyer E. BiCluster Algorithms (biclust) version 1.2.0. Date 2015-05-08 *Repository CRAN Repository/R-Forge/Project biclust Repository/R-Forge/Revision 225* 2015-05-12
- [10] Prelic A., Bleuler S., Zimmermann P., Wil A., Buhlmann P., Gruissem W., Hennig L., Thiele L. and Zitzler E. A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data *Bioinformatics*. *Oxford Univ Press*, 22, 1122-1129 (2006)
- [11] Cheng Y., and Church G.M. Biclustering of Expression. *Data Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, 1, 93-103 (2000)
- [12] Murali T., and Kasif S. Extracting Conserved Gene Expression Motifs from Gene Expression Data. *Pacific Symposium on Biocomputing*, 77-88 (2003)
- [13] Li G., Ma Q., Tang H., Paterson A.H. and Xu Y. QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Res.* Aug;37(15):e101. doi: 10.1093/nar/gkp491. Epub 2009 Jun 9. (2009)
- [14] Wang Z., Li G., Robinson R.W. and Huang X. UniBic: Sequential row-based biclustering algorithm for analysis of gene expression data. *Nature Scientific Reports* 6, Article number: 23466. doi:10.1038/srep23466 (2016)
- [15] Wigner E.P. Characteristic vectors of bordered matrices with infinite dimensions. *The Annals of Mathematics*, 62(3):548-564, (1955)