1 Appendix 2:

Our actual algorithm for finding low-rank biclusters is contained within `tutorial_lakcluster_1.m` and `lakcluster_tutorial_2.m`. These functions both implement the same algorithm using slightly different methods (both are based on Rangan 2012, but the latter uses the 'low-rank updates' we will describe later). Each of these files has its own documentation describing their inputs and outputs, as well as their own tutorials that can be run by calling each function with no arguments.

These functions have been written so that the majority of their inputs are not entered on the command line, but are instead bundled into a text input-file (e.g., '`GSE17536_n2x_frwd.in`'). For the most part the input-file lists the names of the various data-arrays to bicluster, the case- and ctrl-indices, the covariates, and so forth. The format for this input-file is described in detail within the Matlab files `tutorial_lakcluster_1.m` and `tutorial_lakcluster_2.m`, but the basic features of the algorithm can be accessed through the wrapper `tutorial_w1.m` without dealing with the input-files directly.

In broad strokes, the inputs to the low-rank biclustering-algorithm are:

- An $M_D \times N$ case-matrix D. The entry $D(m,n)$ is a real-number recording the gene-expression value for the n^{th}-gene of the m^{th}-case-patient.
- An $M_X \times N$ ctrl-matrix X. The entry $X(m,n)$ is a real-number recording the gene-expression value for the n^{th}-gene of the m^{th}-ctrl-patient.
- An optional $M_D \times 1$ case-covariate vector T. The entry $T(m)$ is the covariate-category of the m^{th}-case-patient.
- An optional $M_X \times 1$ ctrl-covariate vector S. The entry $S(m)$ is the covariate-category of the m^{th}-ctrl-patient.

1.1 Simple case: D only

In its simplest instantiation, X, T and S are all empty, and the algorithm is tasked with finding low-rank biclusters within the case-matrix D. In this case the algorithm reduces to the following very simple iteration:

Step 0 Binarize D, sending each entry to either $+1$ or -1 (i.e., '`D=2*(D>0)-1`');

Step 1 Calculate row-scores '`Z_row=diag(DD'DD');`' and column-scores '`Z_col=diag(D'DD'D);`'.

Step 2 Restrict attention to the row-indices for which Z_{ROW} is large, and column indices for which Z_{COL} is large – e.g., throw away the row/column for which Z_{ROW} or Z_{COL} is smallest.

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 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 again[1].

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, and refer to (Rangan 2012) for a more complete discussion.
- It's fast: Specifically, this algorithm requires only a handful of matrix-multiplications per iteration.Furthermore, the binarization step allow 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 `tutorial_lakcluster_2.m`), reducing the maximum total computation time across all iterations to $O(M_D N \min(M_D, N))$ – 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.

[1] We discuss how exactly we delineate a bicluster later on in section **??**.

The reason this algorithm is successful is because, when calculated across a large matrix, the row- and col-scores $Z_{\rm ROW}$ and $Z_{\rm COL}$ 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 1A) 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 1A, this means that the rows and cols of the embedded bicluster are perfectly correlated; any pair of rows or cols 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 1A, 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).

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 cyan 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 1B, 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 $[Z_{\rm ROW}]_j$. 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 $[Z_{\rm ROW}]_j$ 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 $[Z_{\rm ROW}]_j$ 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 $[Z_{\rm ROW}]_{\tilde{j}}$ 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 $[Z_{\rm ROW}]_{\tilde{j}}$. The loops that that straddle D will be roughly half rank-1 and half rank-2, contributing an average of 0 to $[Z_{\rm ROW}]_{\tilde{j}}$, 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 1C). 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 1C). 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 1C. Rather, after calculating all the row-scores we see something like Fig 1D. 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 the row of D corresponding to the lowest row-score. This eliminated row is 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 cols change, and so we recalculate the scores and repeat. As we eliminate rows and cols, D shrinks, but B is very likely to remain untouched. Consequently, as M_D and N shrink, the 'noise' in our score-distribution decreases (e.g., the noise associated with the row-scores is $\sim \sqrt{M_D N^2}$), whereas the 'signal' remains relatively constant (e.g., the signal associated with the row-scores is $\sim m_B n^2$). This results in the loop-scores of B becoming more and more distinguishable from the other loop-scores; the two distributions shown in Fig 1D become narrower and narrower, while preserving their means. Put another way, as the algorithm progresses the observed distribution of scores (shown in Fig 1D) gradually evolves into a bimodal distribution with two distinct peaks.

1.2 Calculating the scores:

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

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

Note that the row-dependent component of this score is simply $[DD^\intercal DD^\intercal]_{j,j}$, which is sufficient for the purposes of our simple algorithm. The col-scores can be computed similarly, reducing to

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

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

1.3 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 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):

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

Consequently, the row-scores for E can be calculated using those from D along with an update involving only matrix-vector multiplications such as $c^\intercal E$, $(c^\intercal E)\,E^\intercal$, $Er^\intercal$, 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 function `tutorial_lakcluster_2.m`, which also uses similar observations to update the column-scores.

1.4 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 likelihood '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 still be on the same order as $m_B n^2$. Similarly, the col-scores of B will be drawn from a distribution with a mean 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 (Rangan 2012). To start with we imagine that each row 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 $\varepsilon < 1$. 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 $l/\left(l + (m_B - l)\,\varepsilon^2\right) \sim l/\left(l + m_B \varepsilon^2\right)$, 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 ellipses, 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., eccentricity 0).

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 critical 'threshold' will be crossed when $\varepsilon \gtrsim 1/\sqrt{m_B}$, at which point 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 distribition ρ is no longer well captured by its first l principal components.

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 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). On the other hand, v_1 and v_2 will correspond to a rank-2 loop if they happened to lie in adjacent quadrants (thus falling onto adjacent corners of the unit-square). 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

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

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 $N\left(0, \sigma^2\right)$ 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 1.6 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 $P^{2\leftarrow m} = [u^x, u^y]^\intercal$. 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 $[w^x, w^y]^\intercal$ 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 $O\left(1/\sqrt{m}\right)$), 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:

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

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

In the expression above each of the $\hat{w}_j$ represents a 2-element vector $\left[w_j^x, w_j^y\right]^\intercal$, and each of the x_j are drawn from the fixed version of ρ described above; that is to say: $x_1, \ldots, x_l$ are each drawn independently from $\rho_1(x)$, and $x_{l+1}, \ldots, x_m$ 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 $\rho_{\sqrt{1/m},\sqrt{1,m},0}$, implying that the orientation θ_j of the vector $\hat{w}_j$ is uniformly distributed in $[0, 2\pi]$, and the magnitude r_j is drawn from the distribution $r\sqrt{2\pi m}\rho_{\sqrt{1/m}}(r)$, which has mean $\sqrt{\pi/2m}$ and variance $\sim 0.4/m$ (see section 1.6). 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:

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

where each $r_1, \ldots, r_l$ is drawn independently from the gaussian $r\sqrt{2\pi m}\rho_{\sqrt{1/m}}(r)$, and each $\theta_1, \ldots, \theta_l$ is drawn independently from the uniform distribution on $[0, 2\pi]$. The multiple convolution $\left\{\rho_{r_1,0,\theta_1} \star \cdots \star \rho_{r_l,0,\theta_l}\right\}$ 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 $\left\{\rho_{r_1,0,\theta_1} \star \cdots \star \rho_{r_l,0,\theta_l}\right\}$ 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:

$$\begin{aligned} g_{1,\varepsilon,m} &\sim \int_{r_1=0}^{r_1=\infty}\int_0^{2\pi} \frac{1}{2\pi}\left[1-2p+2p^2\right] r_1 \exp\left(\frac{-r_1^2}{2}\right) d\theta_1 dr_1 \\ \text{with } p &= \frac{1}{\pi}\arctan\left(\frac{\frac{\varepsilon\sqrt{m}}{r_1}\sqrt{1+\left[\frac{\varepsilon\sqrt{m}}{r_1}\right]^2}}{-\frac{1}{2}\sin 2\omega}\right). \end{aligned}$$

For the case $l > 1$ we can write out formulae for $g_{l,\varepsilon,m}$ via induction using Eq. 1 in section 1.6. 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 2. 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 $\varepsilon \lesssim 1/\sqrt{m}$ (or even perhaps $\varepsilon \lesssim 10/\sqrt{m}$). In practice, we expect $m := m_B$ to be on the order of 100, (with M_D on the order of $10,000$), implying that our algorithm will succeed so long as $\varepsilon \lesssim 0.1$ or so.

1.5 Why binarize? Advantages and disadvantages

The first step in our algorithm above is to binarize the data: that is, to send each entry of D to -1 or $+1$, depending on its sign.

1.6 Various properties of multivariate gaussians

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

1.6.1 one-dimensional gaussian:

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

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

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 $\left(3\sigma^4 - \sigma^4\right) = 2\sigma^4$.

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

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

1.6.2 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:

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

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

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:

$$\begin{aligned} F\left[\rho_{a,b,\theta}(\vec{x})\right]\left(\vec{k}\right) &= \frac{1}{2\pi}\int \rho_{a,b,\theta}(\vec{x})\exp\iota\left(\vec{k}^\intercal\cdot\vec{x}\right)dx_1dx_2 = F\left[\rho_{a,b,0}(\vec{x})\right]\left(R_{-\theta}\cdot\vec{k}\right) \\ &= \left[\frac{1}{a}\rho_{1/a}\otimes\frac{1}{b}\rho_{1/b}\right]\left(R_{-\theta}\cdot\vec{k}\right) = \frac{1}{ab}\rho_{1/a,1/b,0}\left(R_{-\theta}\cdot\vec{k}\right) \\ &= \frac{1}{ab}\rho_{1/a,1/b,\theta}\left(\vec{k}\right). \end{aligned}$$

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

1.6.3 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.

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

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

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

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

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

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 $\Omega' = \Sigma R_\theta^\intercal\Omega$. This angle in turn can be determined by applying $\Sigma R_\theta^\intercal$ to the boundaries of Ω (i.e., to the x and y axes), yielding the formula above.

1.6.4 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 distribution

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

1.6.5 approximation of orthonormal $\hat{u}, \hat{v}$:

If we randomly draw the numbers $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$ 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 $\left(2/m^2\right)$. 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 $\tilde{v} = v - \hat{u}\hat{u}^\intercal v$ 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 $\hat{u} \cdot \hat{v} = 0$). This means that the original vectors u and v were close to orthonormal to begin with.

1.6.6 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:

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

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

This latter distribution can be determined by observing:

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

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

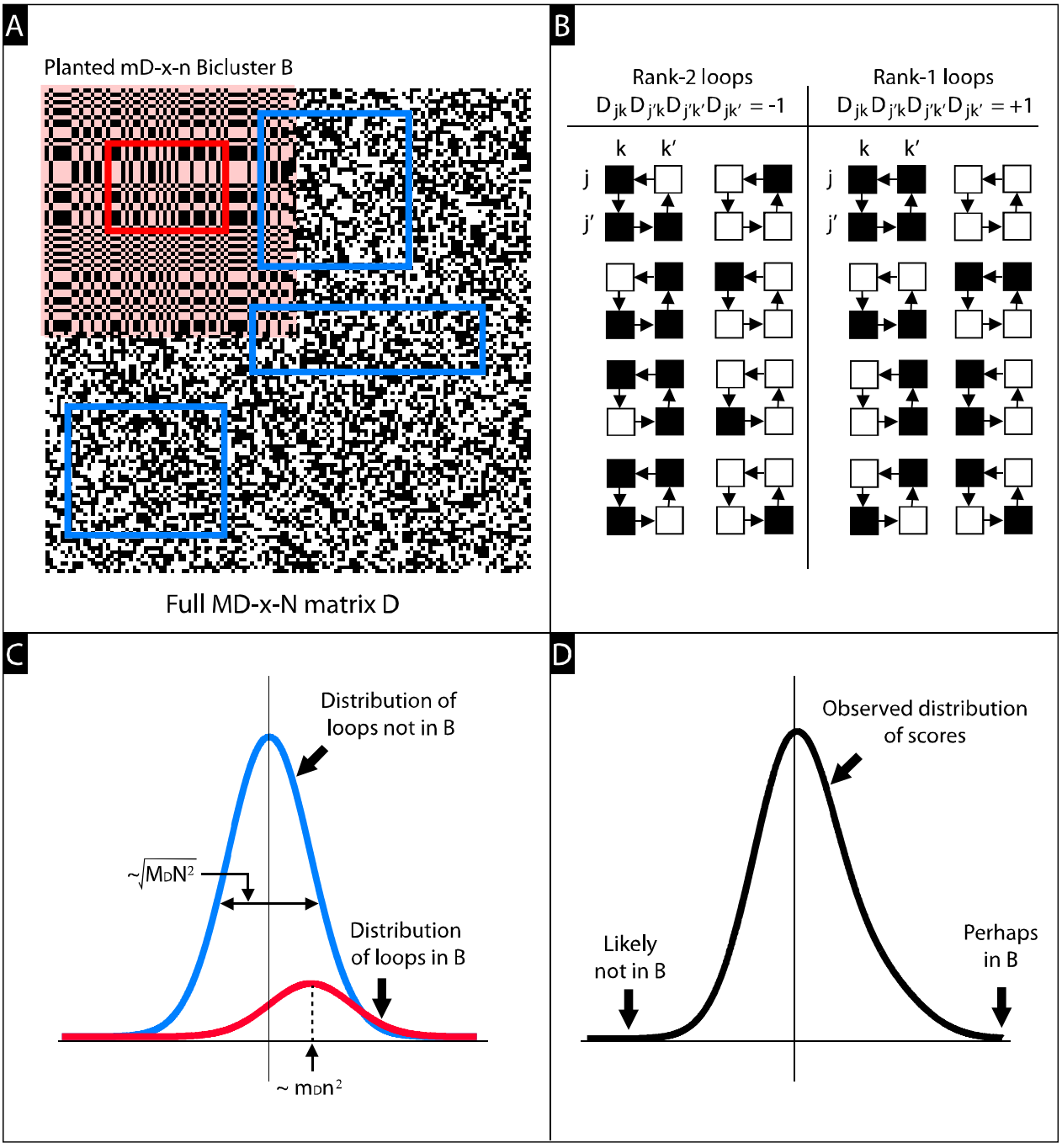
Thus, the distribution $\rho_{f,g,\omega}$ can be determined by finding the f, g, ω that satisfy the following equations (drawn from the exponents in the previous equalities):

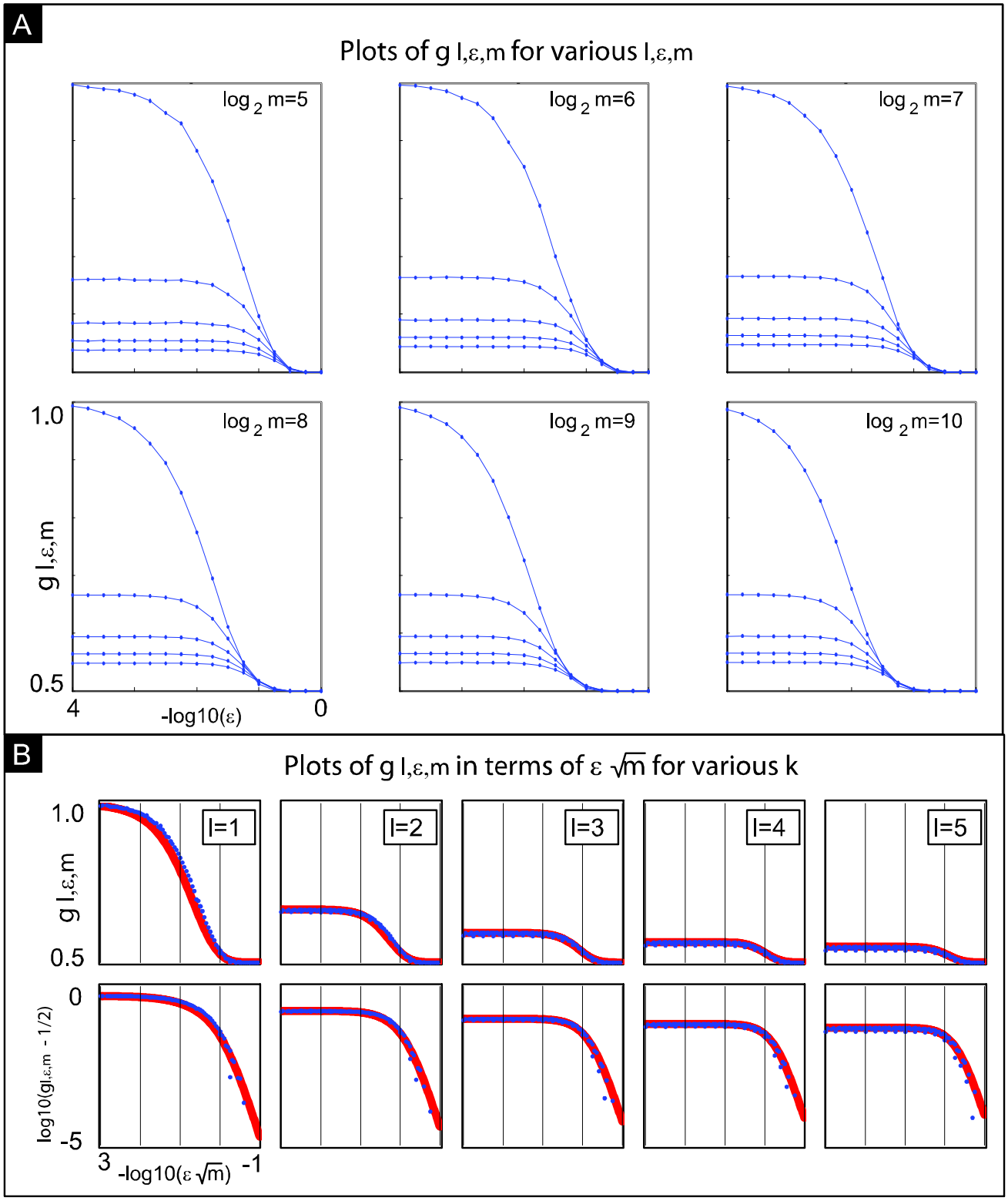
$$R_\theta \begin{bmatrix} a^2 & \\ & b^2 \end{bmatrix} R_\theta^\intercal + R_\phi \begin{bmatrix} c^2 & \\ & d^2 \end{bmatrix} R_\phi^\intercal = R_\omega \begin{bmatrix} f^2 & \\ & g^2 \end{bmatrix} R_\omega^\intercal.$$

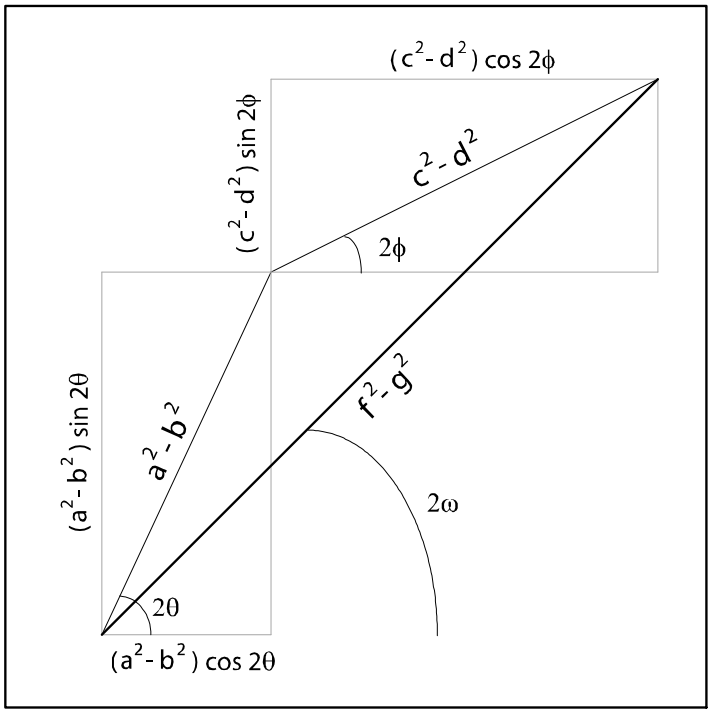
These equalities can be rearranged into the following:

$$\begin{aligned} \left(a^2 - b^2\right)\cos 2\theta + \left(c^2 - d^2\right)\cos 2\phi &= \left(f^2 - g^2\right)\cos 2\omega \\ \left(a^2 - b^2\right)\sin 2\theta + \left(c^2 - d^2\right)\sin 2\phi &= \left(f^2 - g^2\right)\sin 2\omega \\ \left[a^2 + b^2\right] + \left[c^2 + d^2\right] &= \left[f^2 + g^2\right]. \end{aligned} \tag{1}$$

Thus, given θ, $\left[a^2 - b^2\right]$, ϕ and $\left[c^2 - d^2\right]$, we can use the first two equalities to find ω and $\left[f^2 - g^2\right]$ (see, e.g., Fig 3). Once this is accomplished we can use the third equality to find $\left[f^2 + g^2\right]$.


A
Planted mD-x-n Bicluster B
Full MD-x-N matrix D
B
Rank-2 loops
$D_{jk}D_{j'k}D_{j'k'}D_{jk'} = -1$
Rank-1 loops
$D_{jk}D_{j'k}D_{j'k'}D_{jk'} = +1$
k
k′
j
j′
C
Distribution of loops not in B
$\sim\sqrt{M_D N^2}$
Distribution of loops in B
$\sim m_D n^2$
D
Observed distribution of scores
Likely not in B
Perhaps in B


A
Plots of g l,ε,m for various l,ε,m
log 2 m=5
log 2 m=6
log 2 m=7
log 2 m=8
log 2 m=9
log 2 m=10
1.0
0.5
g l,ε,m
4
-log10(ε)
0
B
Plots of g l,ε,m in terms of ε √m for various k
l=1
l=2
l=3
l=4
l=5
1.0
0.5
g l,ε,m
0
-5
log10(gl,ε,m - 1/2)
3
-log10(ε √m)
-1


$(c^2 - d^2)\cos 2\phi$
$(c^2 - d^2)\sin 2\phi$
$c^2 - d^2$
2ϕ
$(a^2 - b^2)\sin 2\theta$
$a^2 - b^2$
$r^2 - g^2$
2ω
2θ
$(a^2 - b^2)\cos 2\theta$