Spectrum: Fast density-aware spectral clustering for single and multi-omic data

Christopher R. John[1*], David Watson[2,3], Michael Barnes[1,3], Costantino Pitzalis[1], and Myles J. Lewis[1]

[1]Centre for Experimental Medicine and Rheumatology, William Harvey Research Institute, Bart's and The London School of Medicine and Dentistry, Queen Mary University of London, Charterhouse Square, London, EC1M 6BQ, UK.

[2]Oxford Internet Institute, University of Oxford, 1 St. Giles, Oxford, OX1 3JS, UK

[3]The Alan Turing Institute, 96 Euston Road, London, NW1 2DB, UK

*To whom correspondence should be addressed

Abstract

Clustering of single or multi-omic data is key to developing personalised medicine and identifying new cell types. We present Spectrum, a fast spectral clustering method for single and multi-omic expression data. Spectrum is flexible and performs well on single-cell RNA-seq data. The method uses a new density-aware kernel that adapts to data scale and density. It uses a tensor product graph data integration and diffusion technique to reveal underlying structures and reduce noise. We developed a powerful method of eigenvector analysis to determine the number of clusters. Benchmarking Spectrum on 21 datasets demonstrated improvements in runtime and performance relative to other state-of-the-art methods.

Contact: christopher.john@qmul.ac.uk

Introduction

Precision medicine is the concept that patients may be stratified into different classes to personalise therapy. A growing number of studies stratify patients using their genome wide expression data (e.g. mRNA, miRNA, protein, methylation), such as those by The Cancer Genome Atlas (TCGA)[1-7] and other large consortia[8]. Clustering algorithms are used to find patient classes and may be run on data from single or multiple platforms. Single-omic (not including single cell RNA-seq) cluster analysis is performed by algorithms such as: Monte Carlo consensus clustering (M3C)[9], CLEST[10], PINSPlus[11], and Similarity Network Fusion (SNF)[12]. However, clustering multi-omic data into an integrated solution is a major current challenge, state-of-the-art methods include: iClusterPlus[13], SNF, CIMLR[14], and PINSplus. The aims of multi-omic clustering are identifying a shared structure between platforms and reducing noise from individual platforms. These are two areas where there is scope for improvement on existing methods. Also important is improving runtimes and developing methods that are effective at both single and multi-omic cluster analysis.

Single-cell RNA-seq is a technique that can be used to detect new cell types by clustering of individual cell transcriptomes[15]. Analysing transcriptomes of individual cells has potential for furthering our understanding of biology and clinical applications. To date, single-cell RNA-seq clustering tools have been developed for clustering this data type only, however, this clustering problem is simply another type of single-omic data clustering. Differences with single-cell RNA-seq data include there are more samples and there are often many dense globular clusters. Tools applied in this domain include: single cell consensus clustering (SC3)[15], Seurat[16], MUDAN, and single-cell interpretation via multikernel learning (SIMLR)[17]. Maintaining fast runtimes is particularly important given the high samples sizes. SIMLR uses a sophisticated procedure to learn the optimal similarity matrix. However, it is quite time consuming and it is not clear if SIMLR provides clustering performance advantages relative to other methods. Given the importance of the above-mentioned clustering tasks, it is timely to develop an effective solution that can address any of these challenges.

Spectral clustering refers to a large class of algorithms that have grown rapidly in the machine learning field due to their ability to handle complex data[18-22]. They are characterised by clustering eigenvectors derived from a matrix representing the data's graph[18]. A few of these methods are currently applied in genomic data analysis[12,17]. However, there have been several advancements in spectral clustering that provide opportunities for new method development. A key development has been the density-aware kernel[22], which calculates sample density to strengthen connections in high density regions of the graph. A recent integrative method uses tensor product graph (TPG) integration and diffusion to leverage higher order information beyond that provided by each individual data source and reduce noise[19]. Another method retrieves eigenvectors of the data's graph selected according to their multimodality for Gaussian Mixture Modelling (GMM) with the Bayesian Information Criterion (BIC) to decide on the number of clusters (K)[20]. The Fast Approximate Spectral Clustering (FASP) method[23] enables rapid analysis of hundreds of thousands of samples on a desktop computer. Inspired by this work our aim was to assemble and advance it by developing Spectrum.

Spectrum includes both methodological advancements and implements pre-existing techniques. Highlights of Spectrum, that make it distinct from previous solutions[12,17], include: 1) A new self-tuning kernel that is density- and scale-aware; 2) A TPG data integration and diffusion procedure to combine data sources; 3) Implementation of the FASP method for massive datasets; 4) A new technique based on eigenvector distributional analysis to find the optimal K; 5) A nonlinear dimensionality reduction and visualisation procedure for multi-omic data. Spectrum is provided as a R software package (https://cran.r-project.org/web/packages/Spectrum/index.html).

Results

Spectrum provides fast effective clustering of single and multi-omic data

First, we tested Spectrum's ability to identify the ground truth K on individual simulated Gaussian datasets (Fig. S1). In each case, Spectrum correctly identified the optimal K. The method can also detect more complex non-Gaussian structures (Fig. S2). To demonstrate performance of Spectrum on real data from a single platform, we ran the algorithm on seven TCGA RNA-seq datasets[1-7]. We used log-rank tests to evaluate the significance of survival time differences between identified clusters. Comparison of Spectrum p values with those from CLEST, M3C, PINSplus, and SNF found that Spectrum performed substantially better overall in finding clusters significantly related to patient survival (Fig. 1a). For comparing different methods, we took both a rank and p value based approach to assess performance, individual p values and rankings for each method on each dataset are included in Supplementary Table 1. To give an initial indication of the relative computational resources required for a single platform analysis, algorithm runtime was investigated on a kidney cancer RNA-seq dataset[6] with 240 samples and 5000 features. This analysis was performed on a single core of an Intel Core i7-6560U CPU @ 2.20GHz laptop computer with 16GB of DDR3 RAM. Spectrum was the fastest method (1.13 seconds), closely followed by SNF (2.67 seconds), PINSplus was still fast (8.53 seconds), M3C (123.91 seconds) and CLEST (283.34 seconds) were both considerably slower.

Next, we conducted a multi-omic data simulation that generated Gaussian clusters with added noise (Fig. S3a) to test the behaviour of Spectrum's data integration system. Individual platform clustering using Spectrum did not detect the optimal K (Fig. S3b,c). However, using the TPG data integration and diffusion method, Spectrum identified the optimal K for the combined dataset (Fig. S3d). We proceeded to test Spectrum's ability to detect clusters with significant differences in survival time on seven multi-omic TCGA datasets relative to other methods (Table 1). The analysis included mRNA, miRNA, and protein data. Similar to our observations on a single platform, Spectrum performed very well (Fig. 1b). Spectrum performed particularly well on the larger datasets with greater potential for clinical significance, namely, the breast (p = 1.47E-07) and brain cancer (p = 3.76E-16) datasets. Next, to gain an initial insight into relative multi-omic runtimes we tested the algorithms on the kidney TCGA dataset[6]. Spectrum performed the fastest (2.5 seconds), followed by SNF (4.06 seconds), PINSplus (27.22 seconds), CIMLR (59.56 seconds), and iClusterPlus (305.35 seconds). A more detailed analysis of runtime was performed for all single and multi-omic algorithms using simulated data. This worked by increasing the number of samples from 100 to 1000 in steps of 100, with each dataset containing 5000 features (Fig. 1c-f). These analyses demonstrated the greater speed of Spectrum relative to other methods. Spectrum's strong performance in finding clinically related clusters comes with a bonus of faster runtimes.

We next demonstrated the advantage of Spectrum's adaptive density-aware kernel by comparison with the classic Zelnik-Manor kernel[21], a non-density-aware kernel that adapts to local data scale only. First, Spectrum using either of the two kernels was run on a non-Gaussian synthetic dataset consisting of two worm structures. The clustering demonstrated that the density-aware kernel improved the classification (Fig. S4a,b). Next differences on TCGA multi-omic data were examined. Analysis of the brain cancer multi-omic dataset[3] found the density-aware kernel detected two additional clusters in comparison with the Zelnik-Manor kernel (Fig. 2a). Spectrum's method for visualising multi-omic clusters was demonstrated. It runs Uniform Manifold Approximation and Projection (UMAP)[28] on the integrated similarity matrix (Fig. 2a), this is a new method inspired by the recent approach applied in single-cell RNA-seq analysis by SIMLR[17]. Plots of the reduced data from UMAP demonstrated the density-aware kernel results in more compact clusters than the Zelnik-

Manor kernel. This was expected, as the adaptive kernel strengthens local connections in the graph where the samples are in regions of high density.

The eigengap method Spectrum uses to decide on K by default was demonstrated (Fig. 2b). For the density-aware kernel, the greatest gap is between eigenvectors five and six, implying an optimal K of five, while for the Zelnik-Manor kernel, the greatest gap is found between eigenvectors three and four, implying an optimal K of three. The survival p values produced by the different methods were shown on survival curves (Fig. 2c). Spectrum obtained a greater level of significance using the density-aware kernel (p = 3.76E-16) than using the Zelnik-Manor kernel (p = 1.68E-11). We expanded this comparison to include all seven TCGA multi-omic datasets to find that the density-aware kernel has a noticeable advantage over the Zelnik-Manor non density-aware kernel (Table S2). These findings demonstrate the improvement gains by using a kernel that considers local density.

Spectrum performs well at identifying cell types in single-cell RNA-seq data

We examined Spectrum's performance on simulated datasets that resemble single-cell RNA-seq, as they were made to consist of many Gaussian blobs that can overlap (Fig. S5). Spectrum identified the correct K for both the K=10 simulated dataset and the K=20 dataset (Fig. S5a-d). We examined Spectrum's performance relative to other methods on seven real single-cell RNA-seq datasets[24-30] by comparing the assigned clusters with the provided cell type labels using Normalised Mutual Information (NMI) (Table 2). Spectrum achieved the highest summed NMI (NMI=5.89), this was closely followed by Seurat (NMI=5.74), MUDAN (NMI=5.71), SC3 (NMI=5.49), and SIMLR (NMI=5.01). Spectrum's summed NMI was favourably weighted by its performance on the Pollen dataset[27] (NMI = 0.95). Using a rank-based score to eliminate this advantage, Spectrum still came joint first with SC3 (Table 2). T-distributed stochastic neighbour embedding (t-SNE) plots showing the Spectrum clustering assignments were produced (Fig. 3a-f, Fig. S6a,b). Overall, Spectrum, Seurat, SC3, and MUDAN performed similarly in these comparisons, however, SIMLR did not perform as well (Table 2).

In the described comparative analysis (Table 2), since the Baron[24] and Muraro[28] datasets had higher numbers of samples, to reduce runtime Spectrum was run using the FASP method (with 900 centroids). Even with the FASP data compression for these two datasets, Spectrum yielded the highest NMI relative to the other methods. Comparing Spectrum runtime on the Baron dataset (N=8569) yielded 1.95 hours without FASP versus 14.23 seconds with FASP, analyses were performed on a single core of an Intel Core i7-6560U CPU @ 2.20GHz laptop computer with 16GB of DDR3 RAM. On the Muraro dataset (N=2126), without FASP took 1.97 minutes and with took 11.83 seconds. Since the complexity of spectral clustering is cubic, $O(N^3)$ and the complexity of k means is linear, $O(KNT)$, where T is the number of k means iterations, using k means as a precursor to compress the data (FASP) is computationally advantageous on larger datasets.

To gain an initial insight into relative runtimes of all methods (without using Spectrum's FASP implementation) methods were run on the Camp dataset[23] (777 samples). This analysis found MUDAN performed the fastest (0.23 seconds), followed by Seurat (2.45 seconds), Spectrum (12.64 seconds), SC3 (183.66 seconds), and SIMLR (264.31 seconds). A runtime analysis was performed for all algorithms on simulated datasets with 500 to 4000 samples in steps of 500 with 1000 features (Fig. S7). Spectrum was in the middle in terms of speed, usually performing faster than the SC3 algorithm. However, SC3 adjusted its own parameters to work faster at higher numbers of samples making it of comparable speed to Spectrum. Spectrum was slower than MUDAN and Seurat, but much faster than SIMLR. Overall, these data demonstrate Spectrum is well suited to clustering small to large single cell RNA-seq datasets, with FASP required for the later.

A fast new heuristic for finding K when performing spectral clustering

Since the eigengap method does not automatically recognise both Gaussian and non-Gaussian structures (Fig. S1,2), we developed a complementary method which can. The method involves examining the multimodality of the eigenvectors of the data's graph Laplacian, so we call it 'the multimodality gap'. To demonstrate this method, five Gaussian blobs were generated (Fig. 4a) and the multimodality of the data's graph's eigenvectors were also displayed (Fig. 4b). The dip-test statistic[31] (Z) which measures multimodality demonstrated a large gap between eigenvectors five and six. Therefore, using this method it was correctly concluded that K=5. Each individual eigenvector was plotted out (Fig. 4c) to visualise the changing distribution of the eigenvectors. As observed in the analysis of the set of Z values, there was a transition from a multimodal distribution at eigenvector five to a unimodal distribution at eigenvector six, confirming K=5. To further validate the method, several simulations were run and the method successfully clustered both complex non-Gaussian (Fig. S8a,b,c,d) and Gaussian clusters (Fig. S9a,b,c,d). However, since the simple method of looking for the greatest gap in the set of Z values can get stuck in local minima (Fig. S9d), the method was further enhanced by adding an algorithm to search for the last substantial gap.

We found the multimodality gap requires kernel tuning to perform well on certain datasets. This was evident in non-Gaussian data simulations, as with kernel tuning there is a perfect clustering result for the spirals test data (Fig. S10a), while without kernel tuning the method fails to cluster correctly (Fig. S10b). Kernel tuning is performed by simply changing the P parameter of the self-tuning kernel and for each kernel finding the maximum multimodality gap between any pair of eigenvectors. The kernel that yields the greatest gap is the optimal kernel, where the most negative D value corresponds to that kernel with the maximum gap (Fig. S10c).

We examined the performance of the multimodality gap across the seven TCGA multi-omic datasets to demonstrate its applicability as a complementary method to the eigengap. This analysis found the multimodality gap can provide different p values compared with the eigengap (Table S3). Optimal methods will vary according to the data. For example, the multimodality gap (p = 0.0019) has a lower p value than the eigengap (p = 0.91) on the kidney cancer data[6]. Including a second method to automatically decide K gives the user power to find the best approach for their data. This method also presents a novel solution to an open problem in spectral clustering.

Discussion

Spectrum provides fast, flexible spectral clustering for complex genome wide expression data. Previous tools have been developed for a single purpose, for example multi-omic clustering[12-14], single-omic clustering[9,10], or single cell RNA-seq clustering[15,17]. However, Spectrum has been demonstrated to work well on all these tasks and therefore is distinct to other methods. Our experiments demonstrated that Spectrum offers state-of-the-art performance in patient stratification, often identifying clusters with greater significant differences in survival time than other methods. Examining our density-aware kernel in comparison with the Zelnik-Manor kernel[21] demonstrated Spectrum's emphasis on strengthening local connections in the graph in regions of high density, partially accounts for its performance advantage. Spectrum is flexible and adapts to the data by using the k nearest neighbour distances instead of global parameters[18,22] when performing kernel calculations.

Spectrum was the fastest method in single-platform and multi-omic TCGA analysis, this is partially due to code optimisation of the density-aware kernel calculation. Even though the kernel is relatively sophisticated, it runs quickly. However, single cell RNA-seq clustering tools are typically faster than single-omic clustering tools used on TCGA data. As we demonstrated, Spectrum performs well at single cell RNA-seq clustering, but has a drawback of increased runtime relative to the community detection methods Seurat and MUDAN. This is compensated for by implementation of a fast data compression method[23] for massive datasets. Even without data compression, Spectrum performed similarly in speed to SC3 and much faster than SIMLR.

The new multimodality gap heuristic for finding K increases the flexibility of the method and increases the appeal of Spectrum to researchers in other disciplines. This is because it can recognise both complex shapes and Gaussian clusters. There are few good solutions to this problem, none of which are implemented in a publicly available R program. The Zelnik-Manor self-tuning algorithm[21] involves a gradient descent technique that is complex to code and time consuming. In contrast, the multimodality gap is more straightforward, effective, and can be used to tune the kernel. Incorporating this method into Spectrum gives the user power to find the best approach for their data. Overall, Spectrum is well suited to clustering complex omic' data and other data types. With its strong performance both in runtime and clustering results it should appeal to a broad range of data analysts.

Methods

Spectrum. Spectrum can be used to cluster continuous single or multi-omic data, the approach can also be used generally for single and multi-view clustering. Let us now describe the method. Let E denote an expression matrix $E \in R^{N \times M}$, where N is the number of samples and M is the number of features. If we are dealing with multiple datasets, then we have a set of T matrices $L = \{E_1, E_2, E_3 \ldots, E_T\}$, where $E_i \in R^{N \times M_i}$ where M_i is the number of features per sample from the i^{th}

data type. The first step of the algorithm is to calculate the similarity matrix or matrices using the adaptive density-aware kernel.

Adaptive density-aware kernel. We first calculate the similarity matrix $A_i \in R^{N_i \times N_i}$ for the i^{th} expression matrix in L, where N_i equals the number of samples from the i^{th} data type. For the i^{th} data type, given a set of N points, $S = \{s_1, s_2, s_3 \dots, s_{N_i}\}$, the adaptive density-aware kernel is:

$$A_{ij} = exp\left(\frac{-d^2(s_i s_j)}{\sigma^i \sigma^j (CNN(s_i s_j)+1)}\right) \quad (1)$$

Where $d(s_i s_j)$ denotes the Euclidean distance between points s_i and s_j, $CNN(s_i s_j)$ denotes the common nearest neighbours shared between the i^{th} and j^{th} samples in the C^{th} nearest neighbours neighbourhood of each sample. σ^i is the distance between the i^{th} sample and its P^{th} nearest neighbour. In the case of multiple datasets, this yields a new set of T similarity matrices, $Y = \{A_1, A_2, A_3 \dots, A_T\}$, while if we have just one platform, $T = 1$, we have a single similarity matrix A.

The adaptive density-aware kernel is inspired by the widely used self-tuning kernel by Zelnik-Manor[21] and the more recent density-aware kernel by Zhang[22]. A drawback of the Zelnik-Manor kernel[21], is that it does not adapt to the local density of the samples; however, it does adapt to scale using local σ^i and σ^j parameters. The problem with the Zhang kernel[22] is that it falls back on the usage of a global σ parameter for the entire dataset, which is difficult and time consuming to tune. It also relies on a global radius parameter, ε, that defines the density around the i^{th} and j^{th} samples when calculating the common nearest neighbours, $CNN(s_i s_j)$. ε, like a global σ, is sensitive to the scale of the specific dataset. Therefore, our aim was to create a kernel that self-tunes to the scale and density of each new dataset by relying on the sample's local statistics.

The adaptive density-aware kernel automatically tunes to the scale and density of each pair by examining the distance from each sample to their P^{th} nearest neighbour and their overlap within the C^{th} nearest neighbour neighbourhood of each sample. The $CNN(s_i s_j)$ calculation tells us how many samples are shared between the C^{th} nearest neighbour neighbourhood of both samples. If these samples are close together in a region of high density, their connectivity will become stronger. Therefore, this kernel adds weight to locally dense connections. C and P are free parameters in this kernel. We set them to default values of 7 and 3, respectively. These parameters were used for all experiments in this manuscript, including those on simulated data, and perform well on a range of genome wide expression data.

Combining multi-view data and tensor product graph diffusion. For combining the similarity matrices (graphs) in the set Y, Spectrum uses a recent technique from the machine learning literature[19] that involves calculating a cross view tensor product graph (TPG) from each pair of individual graphs. The technique has so far not been applied in genome wide data analysis. Cross view TPGs capture higher order information of the data. The cross view TPGs are integrated using linear combinations to form a single graph. Graph diffusion is then performed to reveal the underlying data structure. Shu et al.,[19] give a computationally efficient algorithm for this. The process is mathematically analogous to the TPG approach but can be calculated using a non TPG which makes the process much faster. Spectrum can also use a slight modification of this method for a single data type, more specifically:

1. Combine similarity matrices from the set Y. If we are dealing with a single similarity matrix, $T = 1$, then this step is skipped, but steps 2-5 are the same:

$$A = \sum_{i=1}^{T} A_i \qquad (2)$$

2. Sparsify A by keeping only the Z^{th} nearest neighbours of each sample s_i and setting the rest to 0. Let R_i be the set of Z nearest neighbour samples for s_i. In this method we set $Z = 10$:

$$A(i,j) = \begin{cases} A_{ij} & \text{j} \in R_i \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

3. Row normalise A, so that each row sums to 1:

$$A(i,j) = A(i,j) / \sum_j A(i,j) \qquad (4)$$

4. Perform graph diffusion iterations. Let $Q^1 = A$, and I be the identity matrix for A. Then for the t^{th} iteration from 2 ... $iters$, where $iters = 5$:

$$Q^t = AQ^{t-1}A^T + I \qquad (5)$$

5. We then take the final similarity matrix as $A^* = Q^T$. This ends the procedure. A^* can now be used as the input for spectral clustering.

The parameters $Z = 10$ and $iters = 5$ are set in alignment with previous work[19].

Spectral clustering of similarity matrix. Starting with A^*, Spectrum uses the Ng spectral clustering method[18], but with the eigengap heuristic to estimate the number of clusters and Gaussian Mixture Modelling (GMM) to cluster the final eigenvector matrix. More specifically:

1. Using D, the diagonal matrix whose (i, i) element is the sum of A^*'s i^{th} row, construct the normalised graph Laplacian L:

$$L = D^{-1/2}A^*D^{-1/2} \qquad (6)$$

2. Perform the eigendecomposition of L and thus extract its eigenvectors $x_1, x_2, \ldots\ x_N$ and eigenvalues $\lambda_1, \lambda_2 \ldots \lambda_{N+1}$.

3. Evaluate the eigengap for eigenvalues, starting with the second eigenvalue, $n = 2$, and choose the optimal k, denoted by k^*, for which the eigengap is maximised:

$$k^* = \underset{n}{\operatorname{argmax}}(\lambda_n - \lambda_{n+1})$$

4. Get the $x_1, x_2, \ldots\ x_{k^*}$, k^* largest eigenvectors of L, then form the matrix, $X = [x_1, x_2, \ldots\ x_{k^*}] \in \mathbb{R}^{N \times k^*}$ by stacking the eigenvectors in columns.

5. Form the matrix Y from X by renormalizing each of X's rows to have unit length:

$$Y_{ij} = \frac{X_{ij}}{(\sum_j X_{ij}^2)^{1/2}} \qquad (7)$$

6. Now each row of Y is treated as a sample, s_i, then all samples are clustered into k^* clusters using GMM. Spectrum uses the implementation of GMM from the ClusterR CRAN package.

A heuristic for finding K that analyses eigenvector distributions. How to select the optimal K is an open question in the field of spectral clustering. A natural way to solve this problem when spectral clustering is analysis of the eigendecomposition of the graph Laplacian. The classical eigengap method is effective for Gaussian clusters, however, its rule must be modified to detect non-Gaussian structures thus limiting its applicability (Figure S2). We found a new heuristic for finding K that could be used for Gaussian or non-Gaussian structures and as a complementary method to analyse genome wide expression datasets. The method examines the multimodality of the eigenvectors of the graph Laplacian and looks for a point beyond which there is no more substantial decrease in multimodality.

Intuitively, the degree of multimodality defines how informative a given eigenvector is, and when we pass the optimal K moving along the sorted eigenvectors, $V = \{x_1, x_2, \dots x_N\}$, we expect a large drop in useful information. Multimodality is quantified using the well-known dip test statistic[31]. The method works better if the nearest neighbour parameter of the kernel is tuned from $P = 1, \dots, 10$. This is done by selecting the kernel that gives the maximum multimodality gap. Analysing the multimodality drop was inspired by work by Xiang et al.,[20] where the authors select out the most informative eigenvectors using an Expectation Maximisation technique, then use GMM and the BIC to choose K. However, in our experiments, we found this method to be unreliable. We now detail an alternative procedure for finding K and tuning the kernel based on the analysis of drops in eigenvector multimodality.

Let the set of dip test statistics be $Z = \{z_1, z_2, z_3 \dots, z_N\}$, calculated from the eigenvectors, $V = \{x_1, x_2, \dots x_N\}$. Note that larger values of z_i correspond to greater eigenvector multimodality. To calculate the multimodality difference between consecutive values, we use, $d_i = z_i - z_{i-1}$. Since we require two values to get d_i, the calculation must begin at $i = 2$, which corresponds to the first pair of eigenvectors. Let the set of d_i values calculated from Z be, $D = \{d_1, d_2, d_3 \dots, d_{maxK-1}\}$, the steps for this are as follows:

1. Find the optimal kernel, A^*. Each kernel is calculated using equation 1 and the nearest neighbour parameter P is tuned via a search from $P = 1, \dots, 10$. To do this, calculate the P^{th} graph Laplacian (equation 6) from the P^{th} kernel. Obtain the eigenvectors of these, V_P. Calculate Z_P from these eigenvectors, then D_P. Get $\min(D_P)$ for each P, yielding a set, $D_{min} = \{d_1, d_2, \dots\ d_{10}\}$. A^* is the kernel that corresponds to $min\{D_{min}\}$.
2. Get the KNN graph of A^*, row normalise, then perform diffusion iterations (equations 3-5).
3. Calculate the graph Laplacian, L (equation 6).
4. Perform eigendecomposition of L yielding the eigenvectors, $V = \{x_1, x_2, \dots\ x_N\}$.
5. Calculate the dip test statistics Z for the eigenvectors in V, then calculate the differences of these, D.
6. Pass D into the algorithm described below for finding the last substantial drop in multimodality. Let k^* be the optimal K found by this method.
7. Continue with equations 4-6 corresponding to the Ng spectral clustering method, with GMM to cluster the final eigenvector matrix, with K set to k^*.

One could select K using the maximum multimodality eigengap. However, we found that this simpler method is susceptible to getting stuck in local minima (Fig. S6). This naturally led to making an algorithm to find the last substantial drop. For finding k^* from set Z, we now describe this straightforward algorithm that reads along the elements of Z, to find a point where there is no more substantial decrease in multimodality.

Finding last substantial multimodality gap. This algorithm will search D, storing in memory the biggest difference in multimodality. Let that be d_{min}. A more negative d_i corresponds a bigger drop in the elements of Z. The method then examines if there are any more points ahead of this drop (up until $< c_{max}$ points) that are $\leq \frac{d_{min}}{f}$, where f is the minimum magnitude that the drop must be to replace d_{min}. If a new d_{min} is found, this new difference is stored in memory. This process continues until no more substantial drops are found with the threshold c_{max} to stop the search. More specifically:

1. Skip the first element of D, d_1, as this corresponds to the drop from the 1st to 2nd eigenvector which is non-informative. Then, store in memory d_2, as the greatest drop by default. Call this d_{min}. Initialise a counter to $c = 1$, for keeping count of how many indices ahead we are of the stored d_{min}.
2. Iterate from $d_3 \dots d_{maxK-1}$ and with each iteration check if $\frac{d_{min}}{f} > d_i$. If so let d_i be the new d_{min}, otherwise continue. If $c > c_{max}$, break the loop and accept the current stored d_{min} as the solution.
3. The optimal number of classes is $k^* = i - 1$, where i is the index of D corresponding to the d_i taken as d_{min} in step 2.

The parameters used in this study for the multimodality drop procedure were $c_{max} = 7$ and $f = 2$, values we empirically selected based on our experience.

Generating simulated datasets for analysis

Gaussian cluster simulations were all performed using the CRAN clusterlab package[9], following the standard operating procedure. In the case of non-Gaussian structures, found throughout the Supplementary Figures, either the CRAN mlbench or clusterSim packages were used to simulate the data, using the default settings.

Downloading and processing of real data for analysis

TCGA datasets. The seven multi-omic TCGA datasets[1-7] were downloaded from the Broad Institute (http://gdac.broadinstitute.org/). Pre-normalised data was used for each platform (mRNA, miRNA, and protein) and for every dataset each one was filtered in the same manner, using the coefficient of variation (CV) to select the top 50% most variable features. The data was then $\log_2$ transformed to reduce the influence of extreme values. Code for data pre-processing is found in the following

GitHub repository (https://github.com/crj32/spectrum_manuscript). The processed multi-omic data is in Supplementary File 1. RNA-seq datasets were taken from the same studies as the multi-omic data and filtering of features was done in the same manner. However, more patients were included in the RNA-seq analyses because we did not have to unify the patient IDs between platforms. The RNA-seq data is included in Supplementary File 2. Code for performing log-rank tests in also in the github as well as commands for running methods.

Single-cell RNA-seq datasets. The seven single-cell RNA-seq datasets[24-30] were obtained from the Hemberg lab website (https://hemberg-lab.github.io/scRNA.seq.datasets/). For each dataset we used $\log_2$ normalised counts and selected the top 100 most variable genes for analysis. Code for data pre-processing is found in the manuscript's GitHub repository (https://github.com/crj32/spectrum_manuscript). Additionally, we include the data ready for analysis in Supplementary File 3. Code for calculating NMI is in the github as well as commands for running methods.

Author contributions

C.R.J conceived and designed the approach. C.R.J and D.W wrote the manuscript. C.R.J and D.W wrote the code. C.R.J performed data analyses. All authors reviewed and edited the manuscript. M.B, C.P, and M.L supervised the project.

Additional information

The authors declare that they have no competing interests.

References

1 Agrawal, N. *et al.* Integrated genomic characterization of papillary thyroid carcinoma. *Cell* **159**, 676-690 (2014).

2 Akbani, R. *et al.* Genomic classification of cutaneous melanoma. *Cell* **161**, 1681-1696 (2015).

3 Ceccarelli, M. *et al.* Molecular profiling reveals biologically discrete subsets and pathways of progression in diffuse glioma. *Cell* **164**, 550-563 (2016).

4 Ciriello, G. *et al.* Comprehensive molecular portraits of invasive lobular breast cancer. *Cell* **163**, 506-519 (2015).

5 Fishbein, L. *et al.* Comprehensive molecular characterization of pheochromocytoma and paraganglioma. *Cancer Cell* **31**, 181-193 (2017).

6 Network, C. G. A. R. Comprehensive molecular characterization of clear cell renal cell carcinoma. *Nature* **499**, 43 (2013).

7 Network, C. G. A. R. Comprehensive molecular characterization of urothelial bladder carcinoma. *Nature* **507**, 315 (2014).

8 Lefaudeux, D. *et al.* U-BIOPRED clinical adult asthma clusters linked to a subset of sputum omics. *Journal of Allergy and Clinical Immunology* **139**, 1797-1807 (2017).

9 John, C. R. *et al.* M3C: A Monte Carlo reference-based consensus clustering algorithm. *bioRxiv*, 377002 (2018).

10 Dudoit, S. & Fridlyand, J. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome biology* **3**, research0036. 0031 (2002).

11 Nguyen, H., Shrestha, S., Draghici, S. & Nguyen, T. PINSPlus: A tool for tumor subtype discovery in integrated genomic data. *Bioinformatics* (2018).

12 Wang, B. *et al.* Similarity network fusion for aggregating data types on a genomic scale. *Nature methods* **11**, 333 (2014).

13 Shen, R., Olshen, A. B. & Ladanyi, M. Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis. *Bioinformatics* **25**, 2906-2912 (2009).

14 Ramazzotti, D., Lal, A., Wang, B., Batzoglou, S. & Sidow, A. Multi-omic tumor data reveal diversity of molecular mechanisms that correlate with survival. *Nature communications* **9**, 4453 (2018).

15 Kiselev, V. Y. *et al.* SC3: consensus clustering of single-cell RNA-seq data. *Nature methods* **14**, 483 (2017).

16 Butler, A., Hoffman, P., Smibert, P., Papalexi, E. & Satija, R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology* **36**, 411 (2018).

17 Wang, B., Zhu, J., Pierson, E., Ramazzotti, D. & Batzoglou, S. Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nature methods* **14**, 414 (2017).

18 Ng, A. Y., Jordan, M. I. & Weiss, Y. in *Advances in neural information processing systems.* 849-856.

19 Shu, L. & Latecki, L. J. in *Asian Conference on Machine Learning.* 362-377.

20 Xiang, T. & Gong, S. Spectral clustering with eigenvector selection. *Pattern Recognition* **41**, 1012-1029 (2008).

21 Zelnik-Manor, L. & Perona, P. in *Advances in neural information processing systems.* 1601-1608.

22 Zhang, X., Li, J. & Yu, H. Local density adaptive similarity measurement for spectral clustering. *Pattern Recognition Letters* **32**, 352-358 (2011).

23 Yan, D., Huang, L. & Jordan, M. I. in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* 907-916 (ACM).

24 Baron, M. *et al.* A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure. *Cell systems* **3**, 346-360. e344 (2016).

25 Camp, J. G. *et al.* Multilineage communication regulates human liver bud development from pluripotency. *Nature* **546**, 533 (2017).

26 Darmanis, S. *et al.* A survey of human brain transcriptome diversity at the single cell level. *Proceedings of the National Academy of Sciences* **112**, 7285-7290 (2015).

27 Li, H. *et al.* Reference component analysis of single-cell transcriptomes elucidates cellular heterogeneity in human colorectal tumors. *Nature genetics* **49**, 708 (2017).

28 Muraro, M. J. *et al.* A single-cell transcriptome atlas of the human pancreas. *Cell systems* **3**, 385-394. e383 (2016).

29 Patel, A. P. *et al.* Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science* **344**, 1396-1401 (2014).

30 Pollen, A. A. *et al.* Low-coverage single-cell mRNA sequencing reveals cellular heterogeneity and activated signaling pathways in developing cerebral cortex. *Nature biotechnology* **32**, 1053 (2014).

31 Hartigan, J. A. & Hartigan, P. M. The dip test of unimodality. *The annals of Statistics* **13**, 70-84 (1985).

Figure legends

Figure 1. Spectrum provides fast and effective clustering for single and multi-omic TCGA data. The clustering performance score is the sum of the -$\log_{10}$(p values) from a Cox proportional hazards regression model using a log-rank test to assess significance across seven TCGA datasets (ranks are also included in Table I). (A) Spectrum relative performance just on RNA-seq datasets. (B) Spectrum relative performance on multi-omic datasets (mRNA, miRNA, protein). (C) Runtime analysis for faster single-omic clustering algorithms. (D) Runtime analysis for slower single-omic clustering algorithms. (E) Runtime analysis for faster multi-omic clustering algorithms. (F) Runtime analysis for slower multi-omic clustering algorithms.

Figure 2. The adaptive density-aware kernel demonstrates an advantage in multi-omic analysis. On the right-hand side of the panel are the results for the Zelnik-Manor kernel[21], while the density-aware kernel results are shown on the left-hand side. (A) Spectrum clustering assignments from the brain cancer dataset[3], UMAP was run on the integrated similarity matrices for mRNA, miRNA, and protein data to generate the plots. (B) Eigenvalues for the eigenvectors of the graph Laplacians. The maximum eigengap was taken in each case as the optimal K. (C) Survival curves with p values from a Cox proportional hazards regression model using a log-rank test to assess significance between clusters.

Figure 3. Spectrum performs well at identifying cell types in single cell RNA-seq data. For each dataset normalised mutual information (NMI) between the cell labels and the detected clusters is shown. (A) Heatmap of similarity matrix from Spectrum for the Pollen dataset[27]. (B) t-SNE of the Pollen data[27] overlaid with clustering assignments. (C) Same as B, but from the Li data[25]. (D) Same as B, but from the Camp data[23]. (E) Same as B, but from the Baron data[24]. (F) Same as B, but from the Muraro data[28].

Figure 4. Demonstration of a new heuristic for finding K when spectral clustering. (A) PCA showing a synthetic dataset with five Gaussian clusters. (B) Using the multimodality drop method to find the ground truth K, the last substantial drop is in-between eigenvectors five and six, this refers to an optimal K of five (see methods). (C) Plots showing the individual eigenvectors of the graph Laplacian generated from the synthetic data. The distribution of eigenvector elements for the sixth eigenvector is far more unimodal than the fifth, demonstrating a solution of five is preferable.

Supplemental figure legends

Figure S1. Spectrum performs well at identifying synthetic Gaussian clusters using eigenvalues. In each case a Principal Component Analysis (PCA) of the synthetic data is shown alongside the

eigenvalues of the eigenvectors of the data's graph Laplacian. (A) Results from synthetic data analysis with K=2. (B) Analysis for K=3. (C) Analysis for K=4. (D) Analysis for K=5. In all cases the maximum eigengap method identified the correct number of clusters.

Figure S2. Spectrum performs well at identifying non-Gaussian clusters using eigenvalues with a modified decision rule. In each case a PCA of the synthetic data is shown alongside the eigenvalues of the eigenvectors of the data's graph Laplacian. (A) Results from synthetic data analysis for K=2. (B) Analysis for K=3. In both cases the first non-zero eigengap method identified the correct number of clusters.

Figure S3. Spectrum performs well at reducing noise when clustering data generated in a multi-omic data simulation. (A) Three Gaussian clusters are simulated, each sample belongs to the same cluster in each simulation, but a degree of variability has been added. (B) Individual platform clustering could not detect the optimal K in every case due to noise. (C) Cluster assignments shown in PCAs using individual platforms with no data integration. (D) Spectrum data integration method results in finding K=3 on the combined data.

Figure S4. The locally adaptive density-aware kernel outperforms the Zelnik Manor kernel on a non-Gaussian dataset. (A) Spectrum results with the adaptive density-aware kernel. (B) Spectrum results with the Zelnik Manor non density-aware kernel.

Figure S5. Spectrum performs well on simulated Gaussian data resembling single-cell RNA-seq. (A) PCA of the K=10 synthetic dataset. (B) Eigenvalues of the eigenvectors of the data's graph Laplacian, a maximum drop is located between 10 and 11 correctly designating this as the optimal K decision by the algorithm. (C) PCA of the K=20 synthetic dataset. (D) Eigengap method confirms K=20.

Figure S6. Spectrum results from clustering the Patel single cell RNA-seq dataset. (A) Results from t-SNE analysis of the Patel data[29] overlaid with Spectrum clustering assignments. Normalised mutual

information (NMI) between the cell labels and the detected clusters is shown. (B) Same as A, but for the Darmanis dataset[26].

Figure S7. Runtime analysis for single-cell RNA-seq clustering methods. These analyses were run on a single core of an Intel Core i7-6560U CPU @ 2.20GHz laptop computer with 16GB of DDR3 RAM. Simulated datasets had 1000 features and varying number of samples (N). Spectrum was run without data compression (FASP).

Figure S8. Demonstration of the multimodality gap method for recognising non-Gaussian structures. Z refers to the dip test statistic z_i for the ith eigenvector, where a larger value indicates stronger multimodality. (A) Results using multimodality gap method on a two concentric circle dataset. The optimal K was identified because the gap between two and three is the last substantial one (see methods). (B) Results from analysing a three concentric circle dataset, again the correct K was identified. (C) Results from analysing a two half moon dataset, the correct K was detected. (D) Results from analysing a four cluster smiley face, the correct K was also found.

Figure S9. Demonstration of the multimodality gap method for recognising Gaussian clusters. Z refers to the dip test statistic z_i for the ith eigenvector, where a larger value indicates stronger multimodality. (A) Results from using the multimodality gap method on a K=2 Gaussian synthetic dataset. The optimal K of two is identified by the method because the eigenvalue gap between eigenvectors two and three is the last substantial one. (B) Results from using the multimodality gap method on a K=3 Gaussian synthetic dataset, the correct K is identified. (C) Results from using the multimodality gap method on a K=4 Gaussian synthetic dataset, the correct K is identified. (D) Results from using the multimodality gap on a K=5 Gaussian synthetic dataset, if we took the maximum multimodality gap the method fails, therefore we improved the method to find 'the last substantial gap'. This is found by a greedy algorithm described in the methods.

Figure S10. Kernel tuning can improve the multimodality gap method. Z refers to z_i the dip test statistic for the ith eigenvector, where a larger value indicates stronger multimodality. (A) Results

from the multimodality Spectral clustering method on spirals with kernel tuning of the K parameter that defines the number of nearest neighbours to use when calculating the local sigma (see methods). The correct K is found, as the last substantial gap is in-between eigenvectors two and three. (B) Same dataset, but the kernel tuning was not performed, and the correct K was not found. (C) Tuning results from changing the K parameter (NN). D refers to D_{min} the biggest difference between any consecutive elements of z in the set of dip statistics Z for a given value of K. A lower D indicates a bigger difference found for a given K and kernel, therefore indicating a preferred kernel.

Tables

Table I. Spectrum multi-omic clustering performance relative to other algorithms. P values are from a Cox proportional hazards regression model using a log-rank test to test the significance of the survival time differences between clusters. In brackets next to the p values are the ranks for each dataset**.** The first final row is the summed -log10(p values) for that column (higher is better), the second is the sum of the ranks (lower is better). PCPG stands for Pheochromocytoma and Paraganglioma. For all datasets, the three data types used were mRNA, miRNA, and protein.

Dataset	N	Spectrum	PINSplus	iClusterPlus	SNF	CIMLR
Bladder[7]	338	0.0042 (3)	0.31 (5)	0.0022 (2)	0.00022 (1)	0.0047 (4)
Brain[3]	425	3.76E-16 (1)	0.0053 (4)	1.72E-07 (3)	4.17E-11 (2)	0.013 (5)
Breast[4]	634	1.47E-07 (1)	2.85E-05 (4)	1.78E-05 (3)	0.94 (5)	2.04E-07 (2)
Kidney[6]	240	0.91 (5)	0.038 (2)	0.24 (4)	0.045 (3)	0.0026 (1)
PCPG[5]	80	0.043 (1)	0.18 (4)	0.093 (3)	0.09 (2)	0.54 (5)
Skin[2]	338	0.0014 (1)	0.96 (5)	0.4 (3)	0.51 (4)	0.0029 (2)
Thyroid[1]	219	0.049 (1)	0.09 (2)	0.67 (5)	0.18 (4)	0.17 (3)
P value score		30.21	10.56	16.40	17.49	17.06
Rank score		13	26	23	21	22

Table II. Spectrum single cell RNA-seq clustering performance relative to other algorithms. Values used for scoring each algorithm refer to Normalised Mutual Information (NMI) of given cell type labels versus those defined by the clustering algorithm. The bracketed values are the rank for that algorithm relative to the others for each dataset. The first final row corresponds to the summation of the columns NMI values, the second corresponds to the summation of the ranks.

Dataset	Cells	Spectrum	Seurat	SIMLR	SC3	MUDAN
Camp[25]	777	0.85 (3)	0.87 (1)	0.67 (4)	0.51 (5)	0.86 (2)
Li[27]	561	0.88 (2)	0.86 (3)	0.5 (4)	0.89 (1)	0.86 (3)
Patel[29]	430	0.76 (5)	0.78 (4)	0.79 (3)	0.84 (1)	0.81 (2)
Pollen[30]	301	0.96 (1)	0.86 (4)	0.79 (5)	0.89 (2)	0.87 (3)
Darmanis[26]	466	0.72 (3)	0.78 (1)	0.74 (2)	0.74 (2)	0.7 (4)
Baron[24]	8569	0.84 (1)	0.76 (3)	0.79 (2)	0.76 (3)	0.79 (2)
Muraro[28]	2126	0.88 (1)	0.83 (3)	0.73 (5)	0.86 (2)	0.82 (4)
NMI score		5.89	5.74	5.01	5.49	5.71
Rank NMI score		16	19	25	16	20

Figure 1

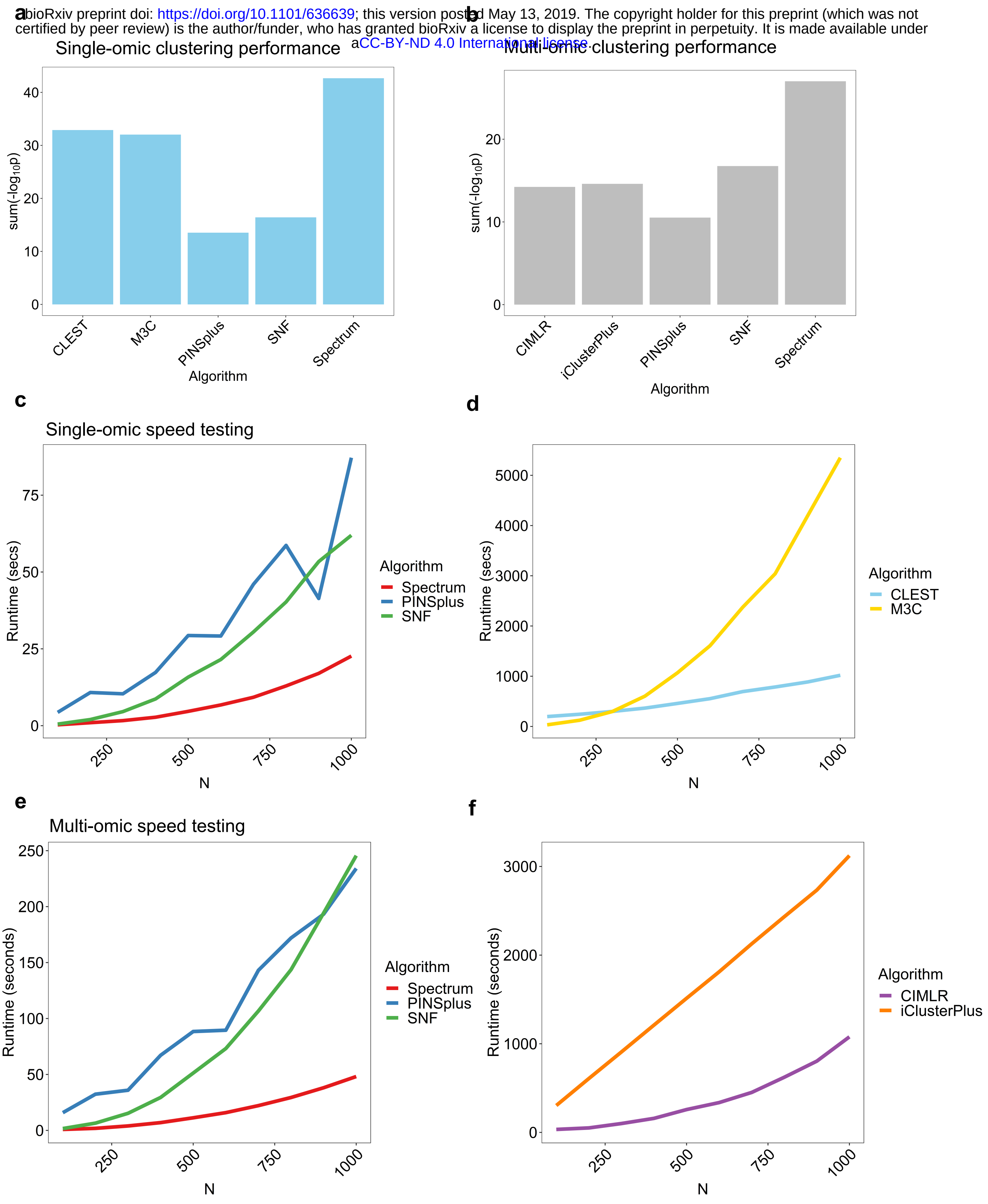
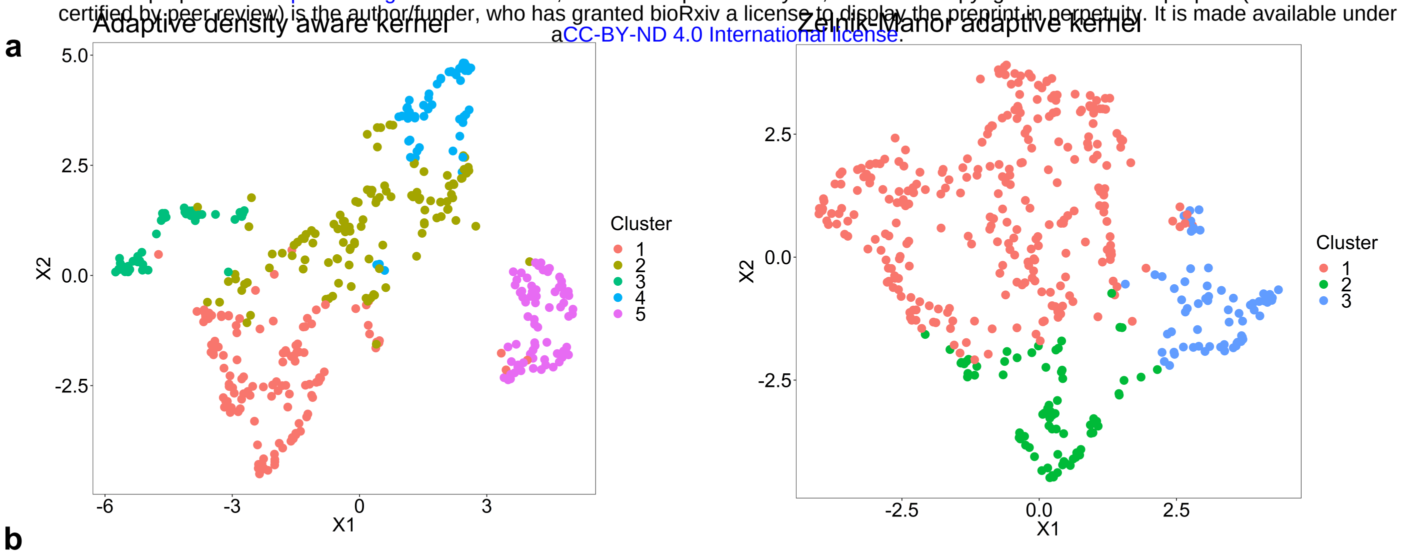

a
Single-omic clustering performance
sum(-log$_{10}$P)
0
10
20
30
40
CLEST
M3C
PINSplus
SNF
Spectrum
Algorithm
b
Multi-omic clustering performance
sum(-log$_{10}$P)
0
10
20
CIMLR
iClusterPlus
PINSplus
SNF
Spectrum
Algorithm
c
Single-omic speed testing
Runtime (secs)
0
25
50
75
250
500
750
1000
N
Algorithm
Spectrum
PINSplus
SNF
d
Runtime (secs)
0
1000
2000
3000
4000
5000
250
500
750
1000
N
Algorithm
CLEST
M3C
e
Multi-omic speed testing
Runtime (seconds)
0
50
100
150
200
250
250
500
750
1000
N
Algorithm
Spectrum
PINSplus
SNF
f
Runtime (seconds)
0
1000
2000
3000
250
500
750
1000
N
Algorithm
CIMLR
iClusterPlus

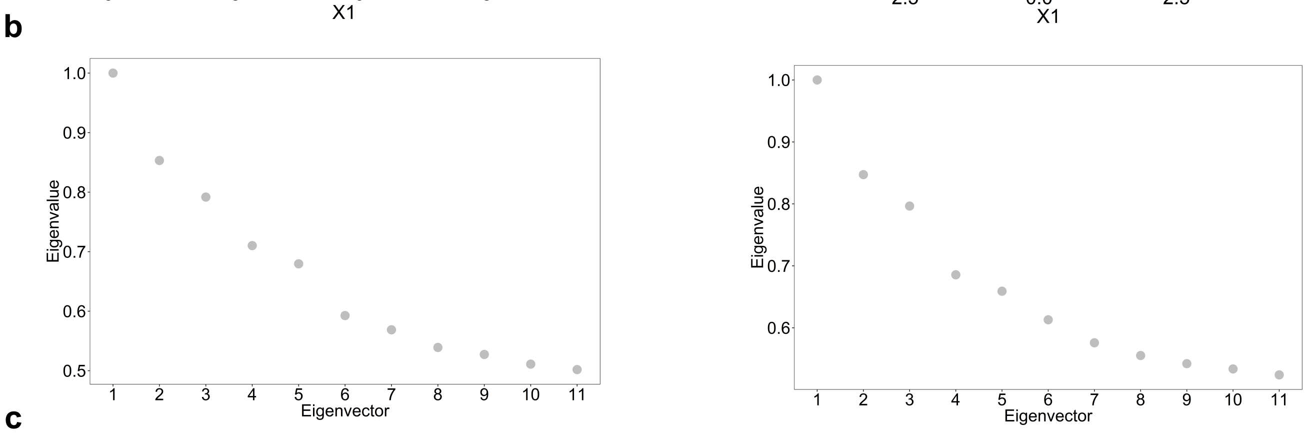
Figure 2

a



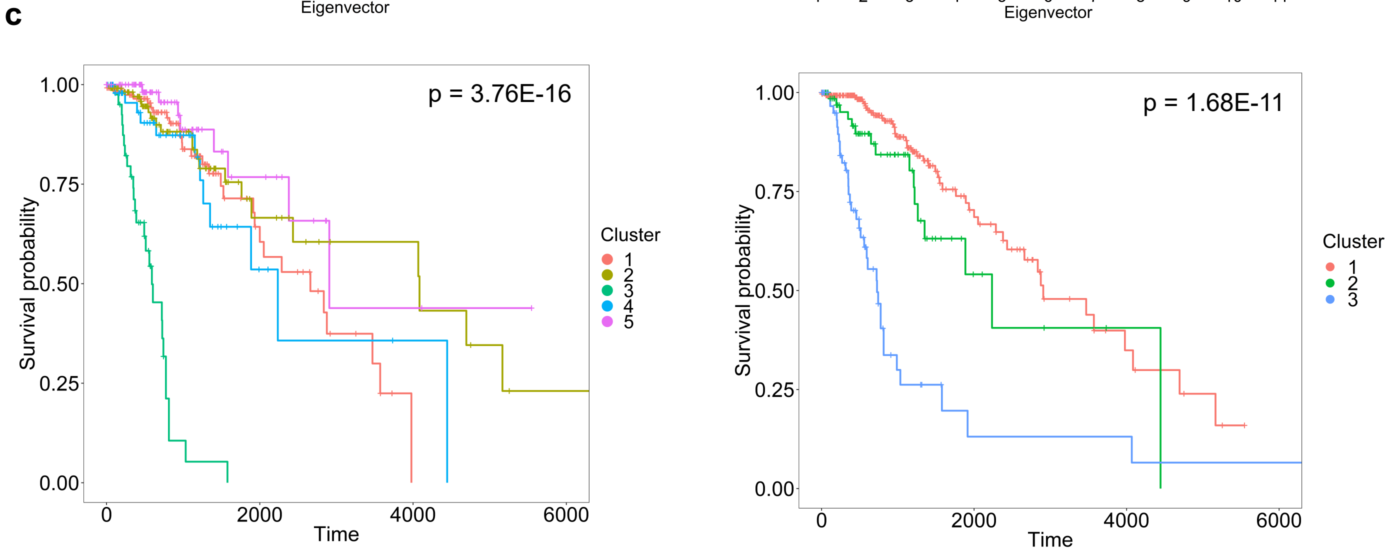
Adaptive density aware kernel
X2
X1
Cluster
1
2
3
4
5
Zelnik-Manor adaptive kernel
Cluster
1
2
3

b



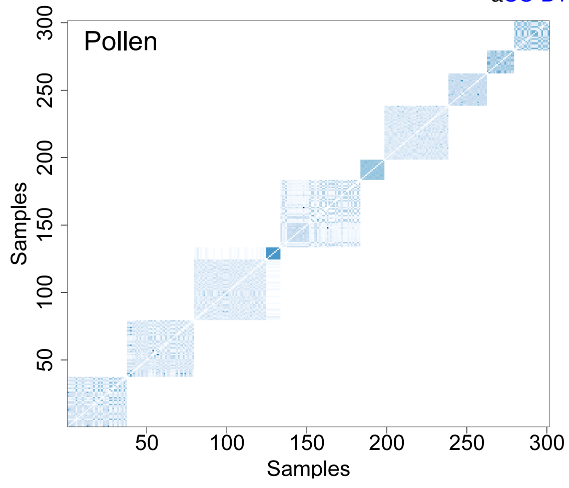
Eigenvalue
Eigenvector

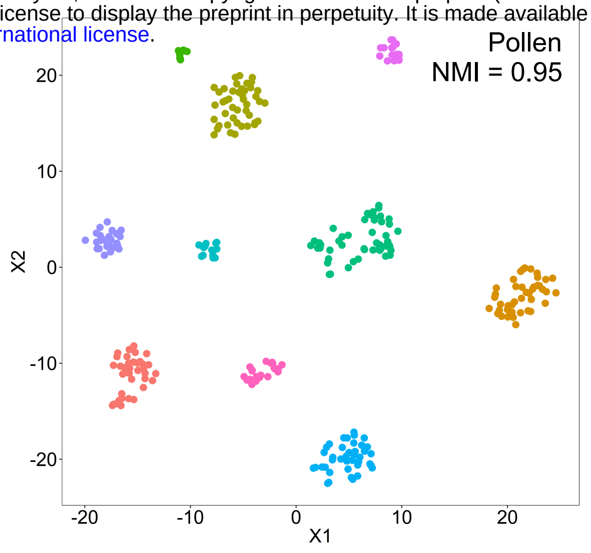
c

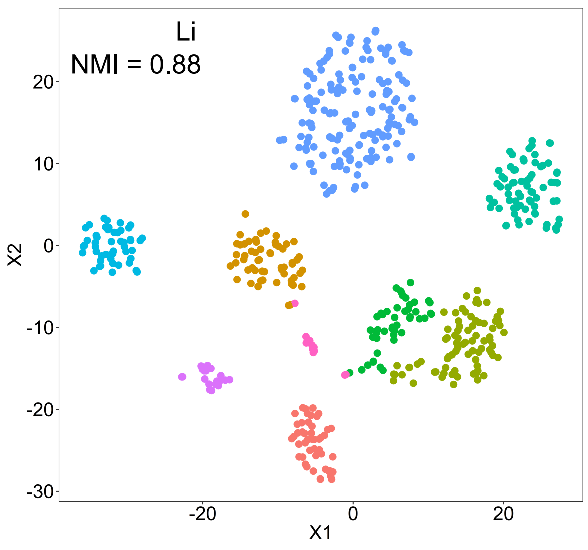


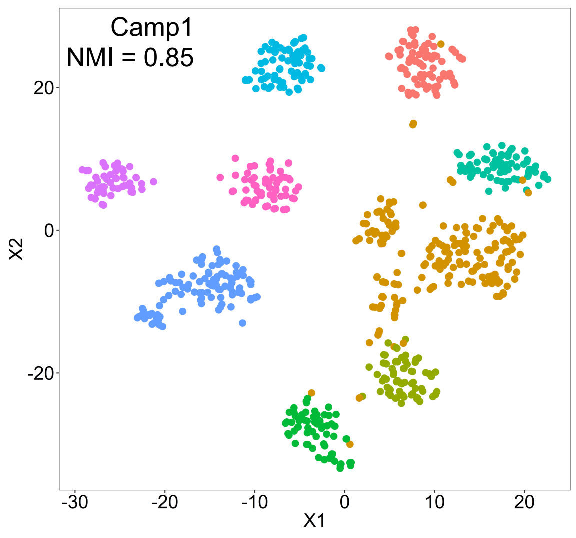
p = 3.76E-16
Survival probability
Time
Cluster
1
2
3
4
5
p = 1.68E-11
Cluster
1
2
3

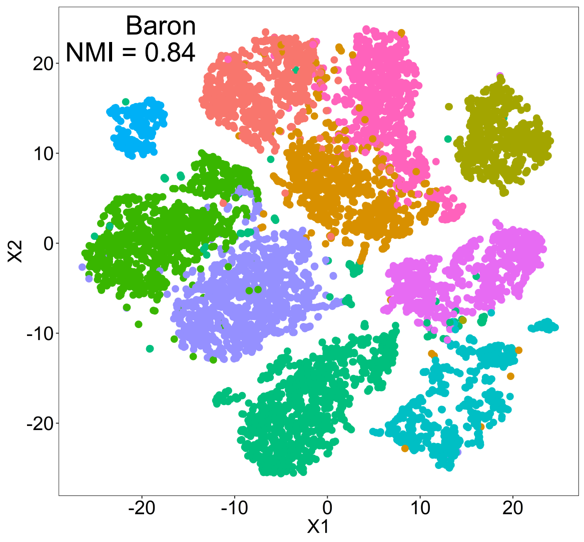
Figure 3

a


b


c


d


e


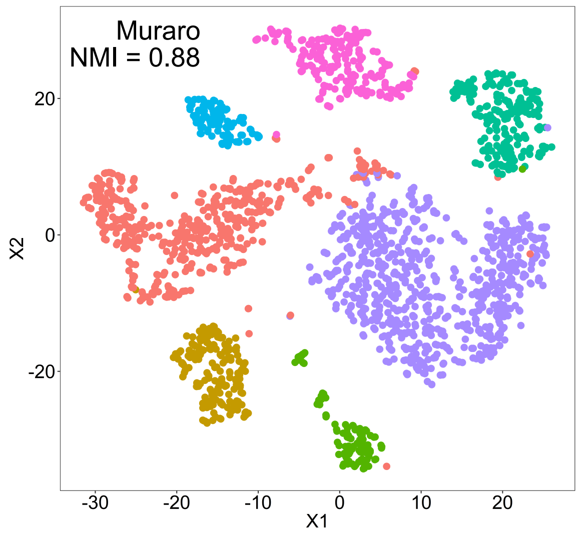
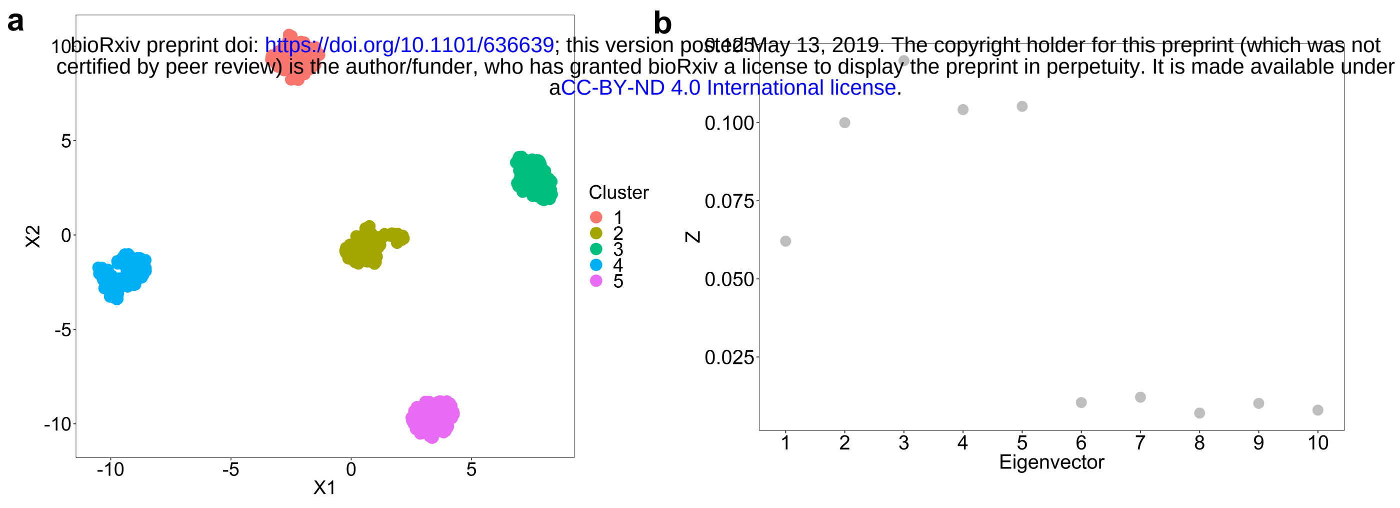
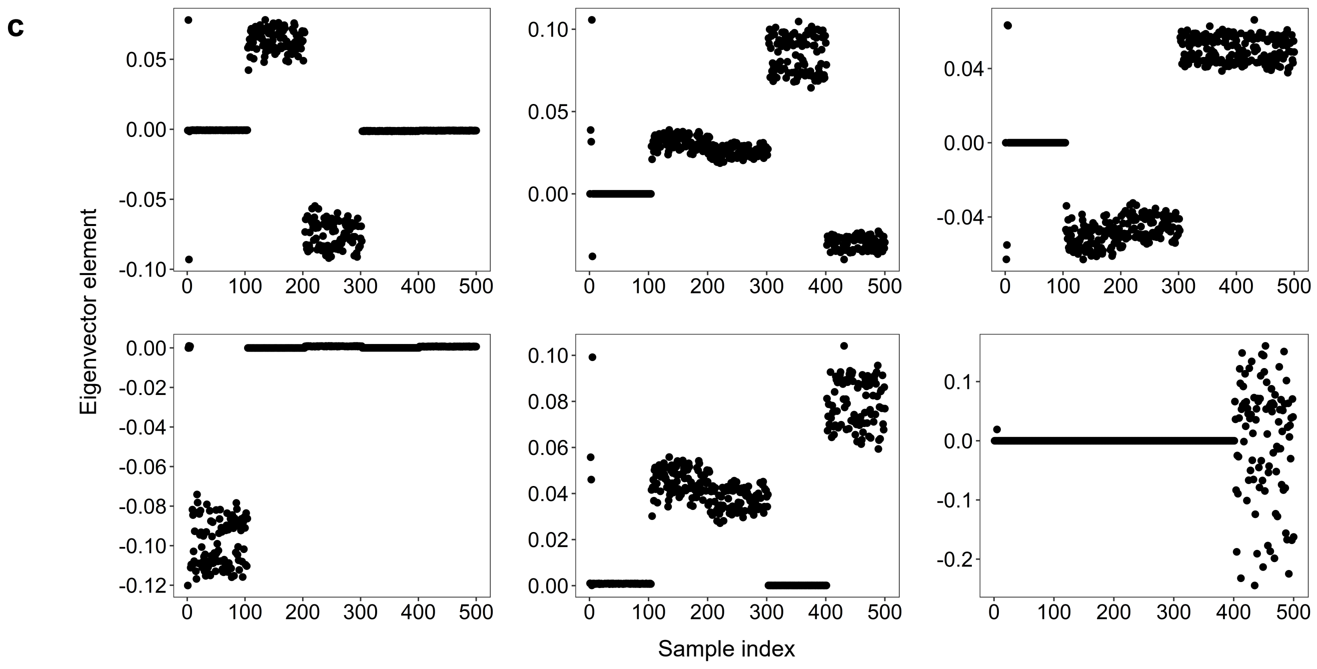
f


Figure 4


a
b
10
5
0
-5
-10
X2
-10
-5
0
5
X1
Cluster
1
2
3
4
5
0.100
0.075
0.050
0.025
Z
1
2
3
4
5
6
7
8
9
10
Eigenvector


c
Eigenvector element
Sample index
0.05
0.00
-0.05
-0.10
0.10
0.04
-0.04
-0.02
-0.06
-0.08
-0.12
0.08
0.06
0.02
0.1
0.0
-0.1
-0.2
0
100
200
300
400
500