Spectrum

Christopher R John

2019-04-08

Spectrum is a fast adaptive spectral clustering method for single or multi-view data. Spectrum uses a new type of density aware kernel that strengthens local connections in dense regions in the graph. It uses a recently developed tensor product graph data integration and diffusion procedure for integrating different data sources and reducing noise. Spectrum examines either eigenvector variance or distribution when determining the number of clusters (K). Gaussian and non-Gaussian structures can be clustered with Spectrum with automatic selection of K.

Contents

1. Data types and requirements

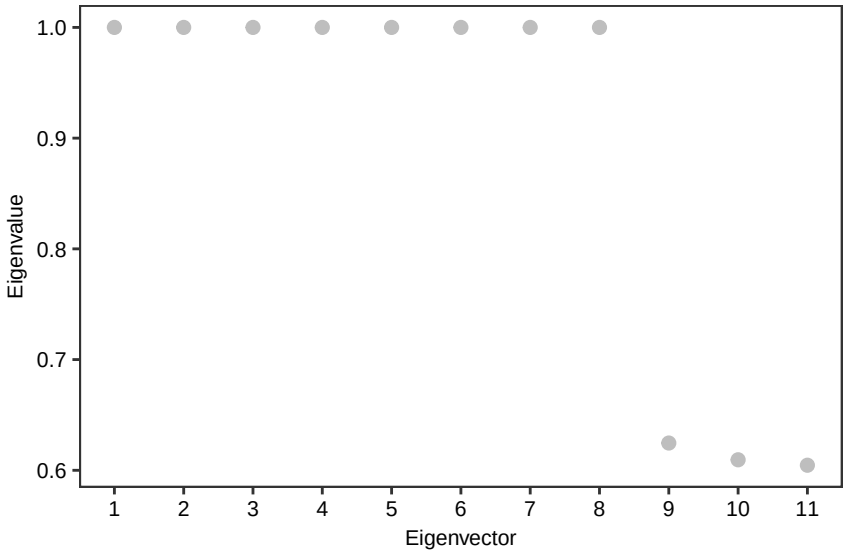
- Data must be in a data frame (single-view) or list of data frames (multi-view)
- Data must have samples as columns and rows as features
- Data should be normalised appropiately so the samples are comparable
- Data should be transformed appropiately so different features are comparable
- Data must be on a continuous scale
- For spectral clustering without finding K automatically, set method to 3 and use the fixk parameter
- For more than 10,000 samples, you can Spectrum in ultra-fast mode (section 7)
- Multi-view data must have the same number of samples in each view and column IDs must be in the same order

2. Single-view clustering: Gaussian blobs

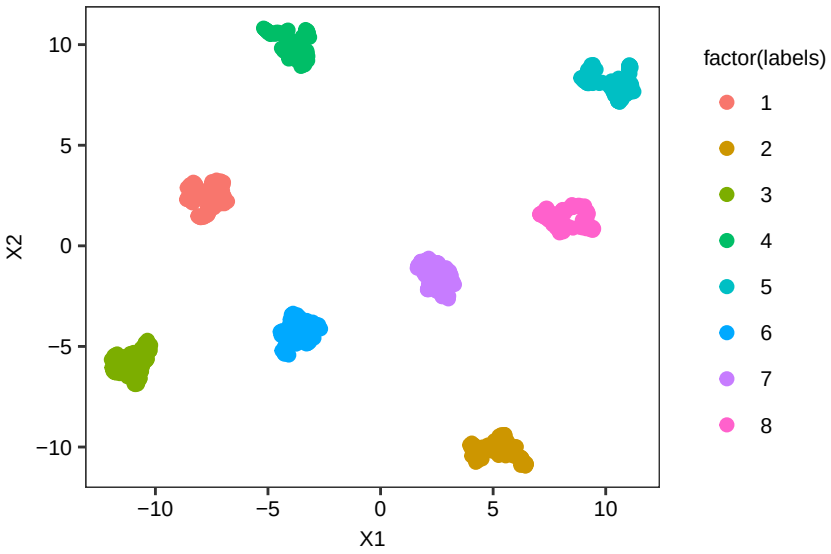
Here we cluster a simulated dataset consisting of several Gaussian blobs. This could represent a number of real world problems, for example, clustering a single omic' data platform (RNA-seq, miRNA-seq, protein, or single cell RNA-seq). Method 1 is set as the default when using Spectrum which uses the eigengap method to find K. We recommend this for most Gaussian clustering tasks. Method 2 which uses the new multimodality gap method which can detect K for non-Gaussian structures as well.

The first plot will show the eigenvalues, where the greatest gap is used to decide K, the second plot shows the results from running UMAP on the similarity matrix. We could also run a t-SNE on the similarity matrix at this stage, by changing the 'visualisation' parameter or a PCA by setting the 'showpca' parameter to TRUE.

```
library(Spectrum)
test1 <- Spectrum(blobs,showdimred=TRUE,fontsize=8,dotsize=2)
#> ***Spectrum***
#> detected views: 1
#> method: 1
#> kernel: density
#> calculating kernel 1
#> done.
#> combining kernels if > 1 and making KNN graph...
#> done.
#> diffusing on tensor graph...
#> done.
#> calculating graph laplacian...
#> getting eigendecomposition of L...
#> done.
#> examining eigenvalues to select K...
#> optimal K: 8
#> doing GMM clustering...
#> done.
#> running UMAP on similarity...
```




```
#> done.
#> finished.
```



Spectrum generates a number of outputs for the user including the cluster each sample is within in the 'assignments' vector contained in the results list (see below code). Use a '$' sign to access the results contained in the list's elements and for more information, see ?Spectrum. Cluster assignments will be in the same order as the input data.

```
names(test1)
#> [1] "assignments"          "eigenvector_analysis" "K"
#> [4] "similarity_matrix"    "eigendecomposition"
```

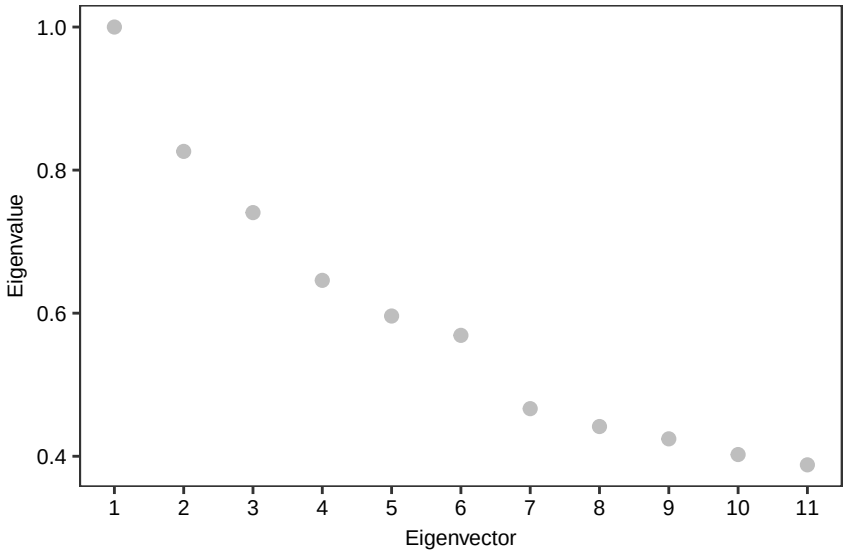
3. Single-view clustering: Brain cancer RNA-seq

Here we cluster a brain cancer RNA-seq dataset with 150 samples again using the eigengap method. The sample size has been reduced because of the CRAN package guidelines.

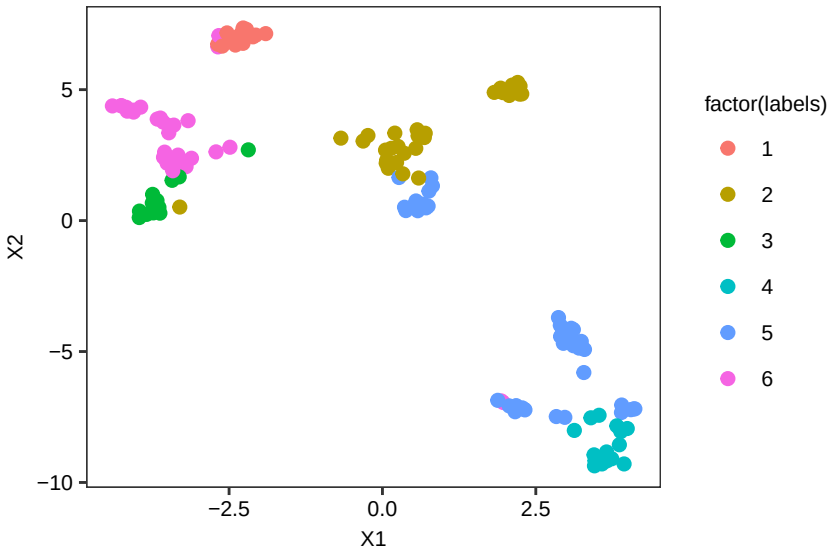
The first plot will show the eigenvalues where the greatest gap in used to decide K, the second plot shows the results from running UMAP on the diffused similarity matrix.

```
library(Spectrum)
RNAseq <- brain[[1]]
test2 <- Spectrum(RNAseq,showdimred=TRUE,fontsize=8,dotsize=2)
#> ***Spectrum***
#> detected views: 1
#> method: 1
#> kernel: density
#> calculating kernel 1
#> done.
#> combining kernels if > 1 and making KNN graph...
#> done.
#> diffusing on tensor graph...
#> done.
#> calculating graph laplacian...
#> getting eigendecomposition of L...
#> done.
#> examining eigenvalues to select K...
#> optimal K: 6
#> doing GMM clustering...
```

```
#> done.
#> running UMAP on similarity...
```




```
#> done.
#> finished.
```



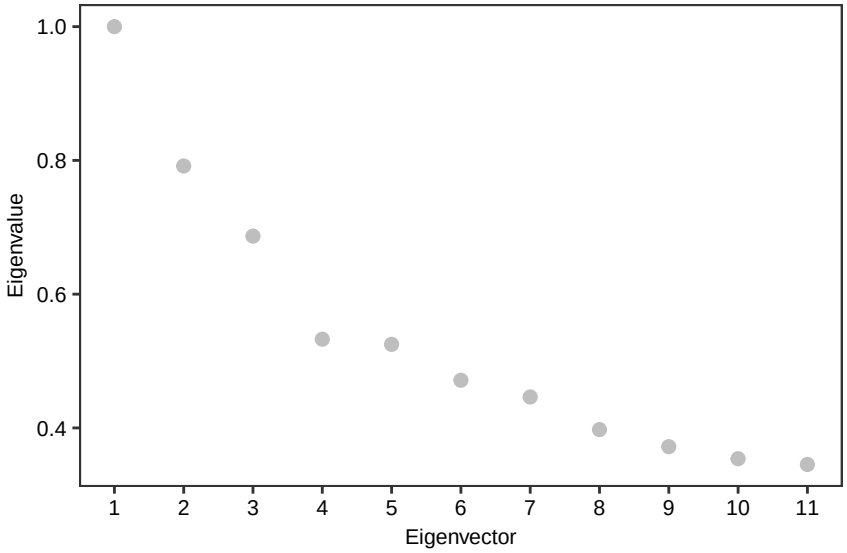
4. Multi-view clustering: Brain cancer multi-omics

Here we cluster multi-omic cancer data with three different platforms (or views): mRNA, miRNA, and protein expression data. This example uses Spectrum's tensor product graph data integration method to combine heterogeneous data sources.

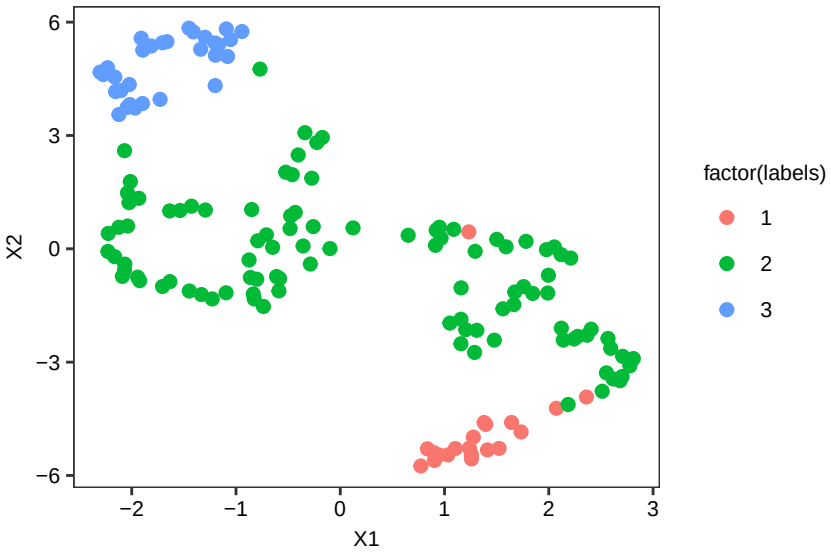
The first plot will show the eigenvalues where the greatest gap in used to decide K, the second plot shows the results from running UMAP on the combined similarity matrix. Running UMAP or t-SNE on the integrated similarity matrix provides a new way of multi-omic data visualisation.

```
library(Spectrum)
test3 <- Spectrum(brain,showdimred=TRUE,fontsize=8,dotsize=2)
```

```
#> ***Spectrum***
#> detected views: 3
#> method: 1
#> kernel: density
#> calculating kernel 1
#> done.
#> calculating kernel 2
#> done.
#> calculating kernel 3
#> done.
#> combining kernels if > 1 and making KNN graph...
#> done.
#> diffusing on tensor graph...
#> done.
#> calculating graph laplacian...
#> getting eigendecomposition of L...
#> done.
#> examining eigenvalues to select K...
#> optimal K: 3
#> doing GMM clustering...
#> done.
#> running UMAP on similarity...
```




```
#> done.
#> finished.
```



5. Single-view clustering: Non-Gaussian data, 3 circles

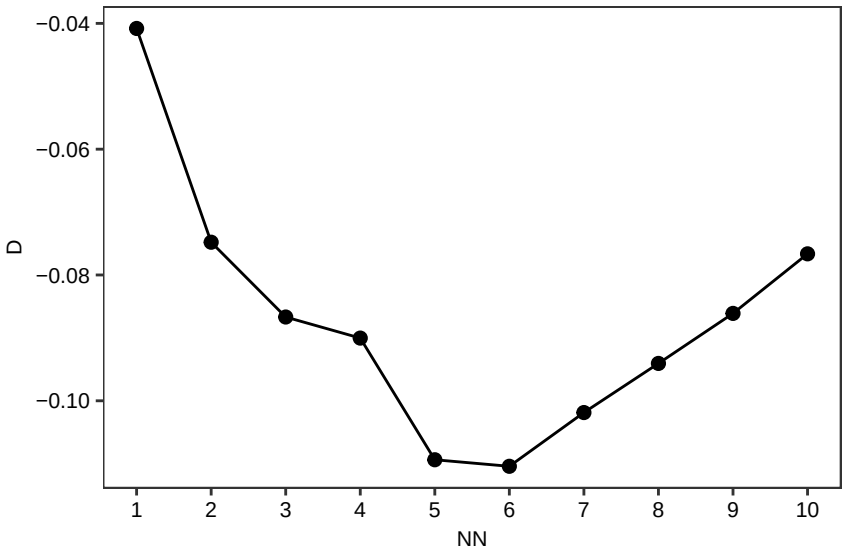
For analysing non-Gaussian structures and determining K automatically, it is much better to use our method that examines the multimodality of the eigenvectors and searches for the last substantial drop. This method, by default, also tunes the kernel by examining the multimodality gaps, although this is not always necessary. The method can handle Gaussian clusters too, multimodality is quantified using the well known dip-test statistic (Hartigan et al., 1985).

The first plot shows the results from tuning the kernel's nearest neighbour parameter (NN). D refers to the greatest difference in dip-test statistics between any consecutive eigenvectors of the graph Laplacian for that value of NN. Spectrum automatically searches for a minimum (corresponding to the maximum drop in multimodality) to find the optimal kernel. In the next plot, we have the dip test statistics (Z) which measure the multimodality of the eigenvectors.

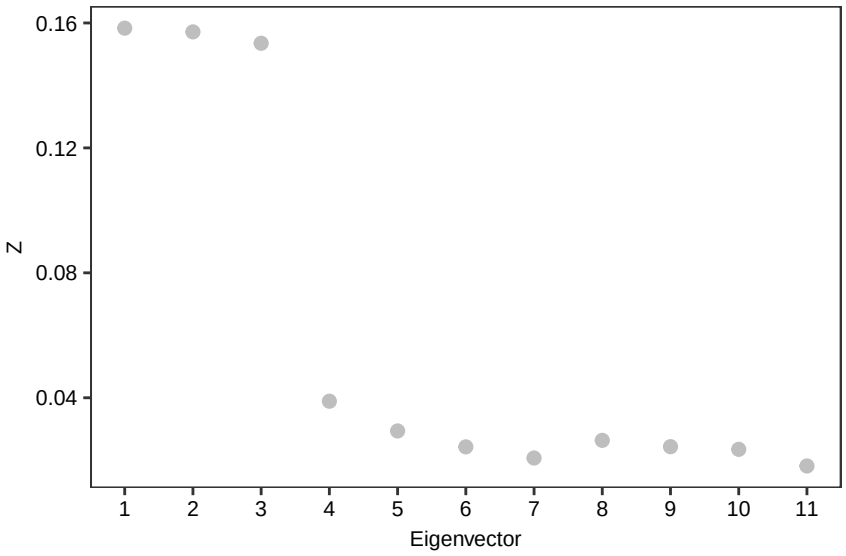
When the multimodality is higher, the eigenvectors are more informative and represent blocks of the data's similarity matrix. Thus, when there is a big gap this may correspond to the optimal K as we have no more blocks to find. However, Spectrum has its own greedy algorithm to search for 'the last substantial gap' instead of the greatest gap. This is because searching for the greatest gap sometimes gets stuck in local minima without including all informative eigenvectors. The parameters for the search can be adjusted with the 'thresh' and 'frac' parameters. The last plot shown here is PCA to visualise the data, run just on the input data for the user.

```
library(Spectrum)
test4 <- Spectrum(circles,showpca=TRUE,method=2,fontsize=8,dotsize=2)
#> ***Spectrum***
#> detected views: 1
#> method: 2
#> kernel: density
#> calculating kernel 1
#> finding optimal NN kernel parameter by examining eigenvector distributions
#> tuning kernel NN parameter: 1
#> tuning kernel NN parameter: 2
#> tuning kernel NN parameter: 3
#> tuning kernel NN parameter: 4
#> tuning kernel NN parameter: 5
#> tuning kernel NN parameter: 6
```

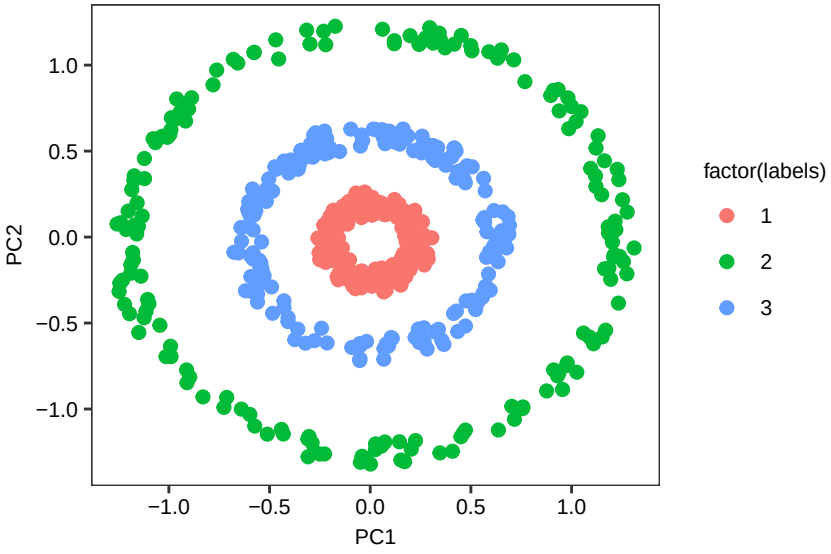
```
#> tuning kernel NN parameter: 7
#> tuning kernel NN parameter: 8
#> tuning kernel NN parameter: 9
#> tuning kernel NN parameter: 10
#> optimal NN:6
#> done.
#> combining kernels if > 1 and making KNN graph...
#> done.
#> diffusing on tensor graph...
#> done.
#> calculating graph laplacian...
#> getting eigendecomposition of L...
#> done.
#> examining eigenvector distributions to select K...
#> finding informative eigenvectors...
#> done.
```




```
#> optimal K: 3
#> doing GMM clustering...
#> done.
```




```
#> finished.
```

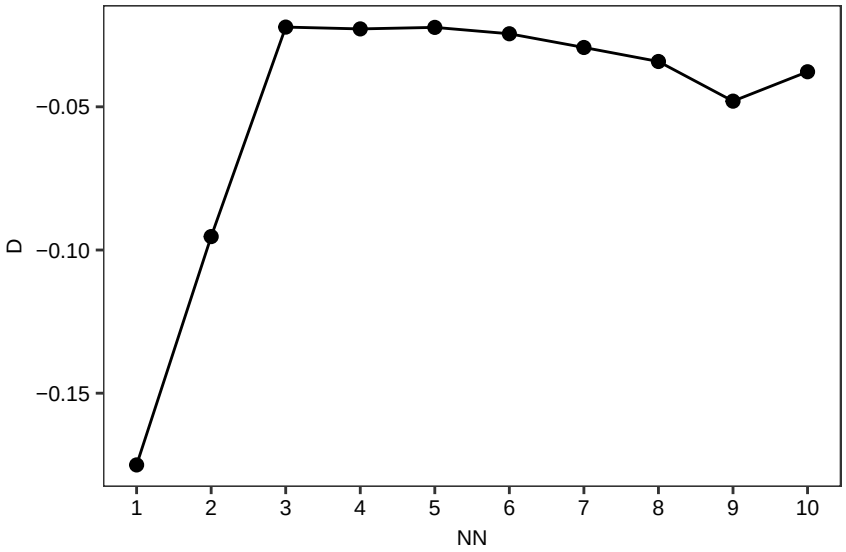


6. Single-view clustering: Non-Gaussian data, spirals

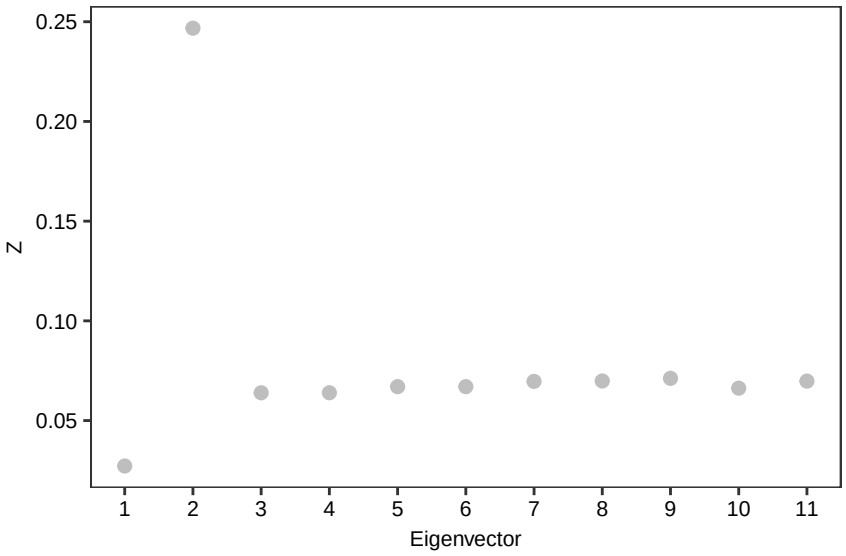
Same as the last example, but for the spirals dataset. In this example, kernel tuning is required to detect the optimal K.

```
library(Spectrum)
test5 <- Spectrum(spirals,showpca=TRUE,method=2,fontsize=8,dotsize=2)
#> ***Spectrum***
#> detected views: 1
#> method: 2
#> kernel: density
#> calculating kernel 1
#> finding optimal NN kernel parameter by examining eigenvector distributions
#> tuning kernel NN parameter: 1
#> tuning kernel NN parameter: 2
#> tuning kernel NN parameter: 3
```

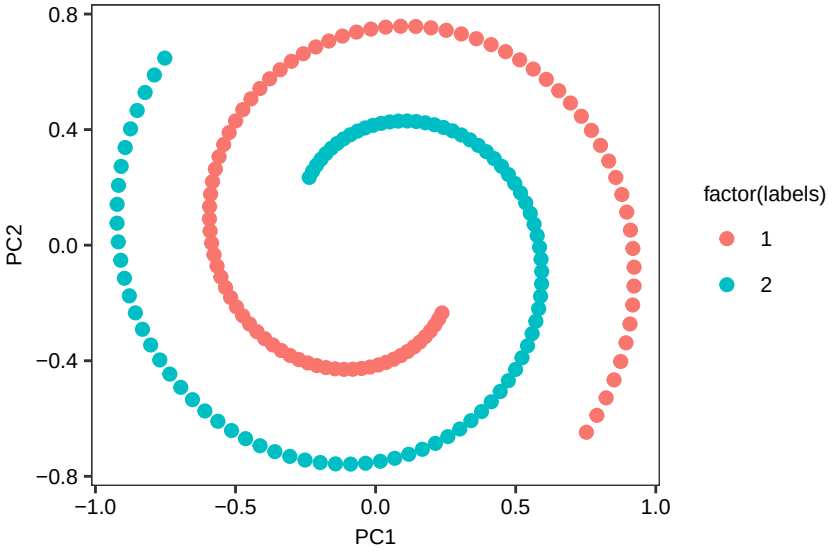
```
#> tuning kernel NN parameter: 4
#> tuning kernel NN parameter: 5
#> tuning kernel NN parameter: 6
#> tuning kernel NN parameter: 7
#> tuning kernel NN parameter: 8
#> tuning kernel NN parameter: 9
#> tuning kernel NN parameter: 10
#> optimal NN:1
#> done.
#> combining kernels if > 1 and making KNN graph...
#> done.
#> diffusing on tensor graph...
#> done.
#> calculating graph laplacian...
#> getting eigendecomposition of L...
#> done.
#> examining eigenvector distributions to select K...
#> finding informative eigenvectors...
#> done.
```




```
#> optimal K: 2
#> doing GMM clustering...
#> done.
```




```
#> finished.
```



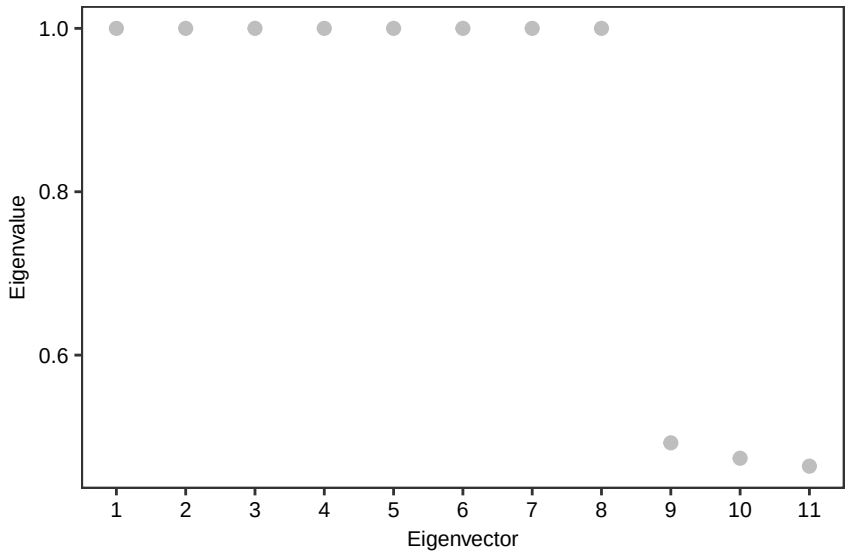
7. Ultra-fast single-view clustering: Gaussian blobs II

To enable clustering of very high numbers of samples (e.g. 10,000-100,000+) on a single core of a Desktop computer, Spectrum implements the Fast Approximate Spectral Clustering (FASP) method (Yan et al., 2009). FASP computes k centroids and then uses these for clustering, after running Spectrum will then assign the original data to clusters via their centroids. This option is recommended for very large datasets, the loss in accuracy is usually marginal, see Yan et al. (2009) for further details.

To perform this method the user needs to set the FASPk parameter to a suitable number for their data, this depends on the data to some degree. For example, if we had 50,000 samples, we might set FASPk to 1,000, to reduce the clustering data input size by 50x.

```
library(Spectrum)
test6 <- Spectrum(blobs,FASP=TRUE,FASPk=300,fontsize=8,dotsize=2)
#> ***Spectrum***
#> detected views: 1
#> method: 1
```

```
#> kernel: density
#> FASP method: TRUE
#> calculating kernel 1
#> done.
#> combining kernels if > 1 and making KNN graph...
#> done.
#> diffusing on tensor graph...
#> done.
#> calculating graph laplacian...
#> getting eigendecomposition of L...
#> done.
#> examining eigenvalues to select K...
#> optimal K: 8
#> doing GMM clustering...
#> done.
#> finished.
```



Looking at the results, we have found the same K as before (8), but with a reduced sample size and runtime. Observe the output below, where there is a new item in the list for the original sample assignments.

```
names(test6)
#> [1] "allsample_assignments" "centroid_assignments"  "eigenvector_analysis"
#> [4] "K"                     "similarity_matrix"     "eigendecomposition"
```

```
head(test6[[1]])
#> c1s1 c2s1 c3s1 c4s1 c5s1 c6s1
#>    6    8    7    4    3    2
```

8. Parameter settings

Generally speaking, Spectrum is set up to handle a wide range of data automatically with its self-tuning kernel on the default settings. However, we provide the following advice in more specific circumstances.

- For data containing non-Gaussian clusters; use method 2, which can automatically detect K for these structures. In some cases, the greedy search parameters, 'thresh' and 'frac', which define the cut-offs for searching for the last substantial gap in multimodality, may require manually changing for the data

type.

- For very high numbers of samples; switch FASP mode on and set 'FASPk' to e.g. 500-10,000. The 'FASPk' parameter should be high enough to get a good representation of the data.
- Kernel tuning for method 2 is optional, it can help, but for a large dataset it may be too time consuming.
- For method 2, it is better to set 'diffusion' to FALSE. We tested the method on several single cell RNA-seq datasets and it performs much better without graph diffusion.
- Although in our experiments we found our adaptive density aware kernel superior to the classic Zelnik-Manor et al. (2005) self tuning kernel, the user might want to experiment on their data with the Zelnik-Manor kernel (stsc) also included in Spectrum.
- For noisy overlapping Gaussian data, such as a lot of RNA-seq and multi-omics it is best to use method 1 (eigengap) to decide K. Method 2 tends to work better when there are clearer blocks in the similarity matrix.
- The kernel parameters N and NN2 can be experimented with which control the number of nearest neighbours used when calculating the local sigma or density parameter. However, the default parameters have been tested with a wide range of real and simulated data and generalise well.

9. Closing comments

Spectrum is an unique assembly of spectral clustering techniques designed to serve a broad community. Included are both innovations for the field, such as the adaptive density aware kernel and the multimodality gap procedure, as well as implementations of others state-of-the-art methods. As we have demonstrated in this vignette, it is suitable for a wide range of clustering tasks.

10. References

Hartigan, John A., and Pamela M. Hartigan. "The dip test of unimodality." The annals of Statistics 13.1 (1985): 70-84.

Yan, Donghui, Ling Huang, and Michael I. Jordan. "Fast approximate spectral clustering." Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009.

Zelnik-Manor, Lihi, and Pietro Perona. "Self-tuning spectral clustering." Advances in neural information processing systems. 2005.