Spectrum

Christopher R John

2019-03-15

Spectrum is a fast self-tuning spectral clustering method for single or multi-view data. Spectrum uses a new type of adaptive density aware kernel that strengthens local connections in dense regions in the graph. For integrating multi-view data and reducing noise it uses a recently developed tensor product graph data integration and diffusion system. Spectrum contains two techniques for finding the number of clusters (K); the classical eigengap method and a multimodality gap procedure. The multimodality gap analyses the distribution of the eigenvectors of the graph Laplacian to decide K. Spectrum is suited for clustering a wide range of complex data.

Contents

1. Data requirements

- Data should be in a data frame (single view) or list of data frames (multi-view)
- Data should 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
- Data may be small (10s-100s of samples) to large (1000s-10000s of samples)
- Multi-view data must have the same number of samples and samples must be in exactly the same order

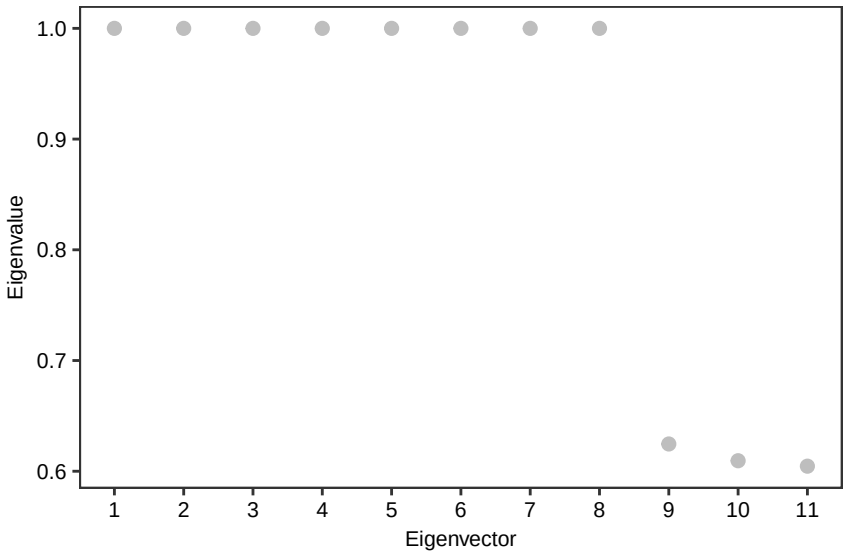
2. Single-view clustering: Gaussian blobs

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 standard Gaussian clustering tasks. Method 2 which uses the new multimodality gap method can handle non-Gaussian structures as well, although it is a little slower because of the kernel tuning.

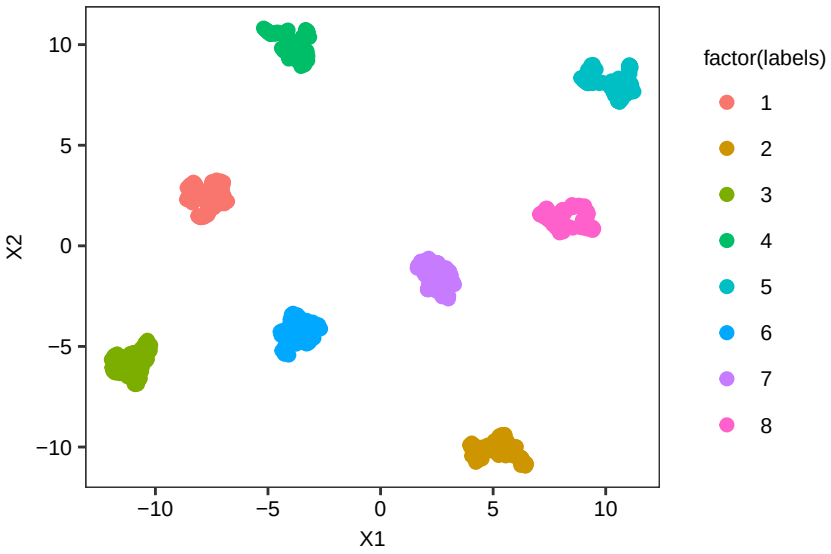
The first plot will show the eigenvalues where the greatest gap in used to decide K, the second plot shows the results from running UMAP on the affinity matrix. We could also run a t-SNE on the affinity matrix at this stage, by changing the 'visualisation' parameter or a PCA by setting the 'showpca'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

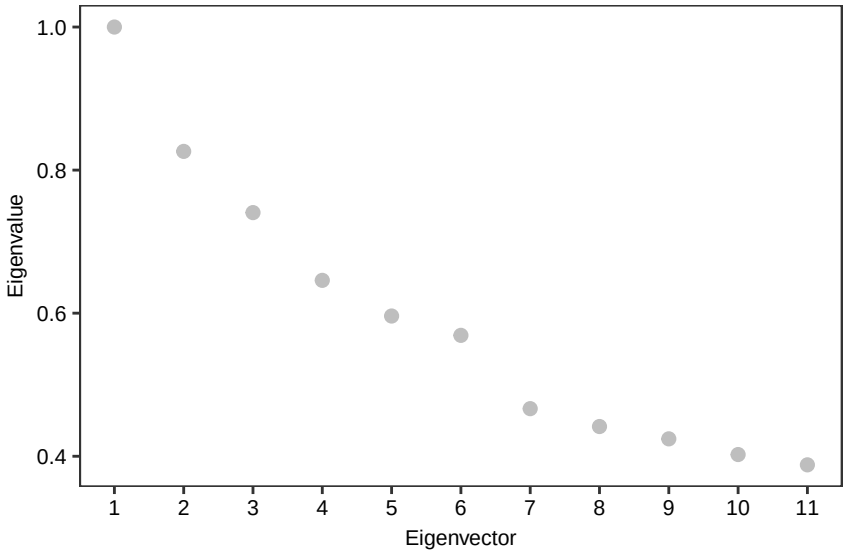
3. Single-view clustering: Brain cancer RNA-seq

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

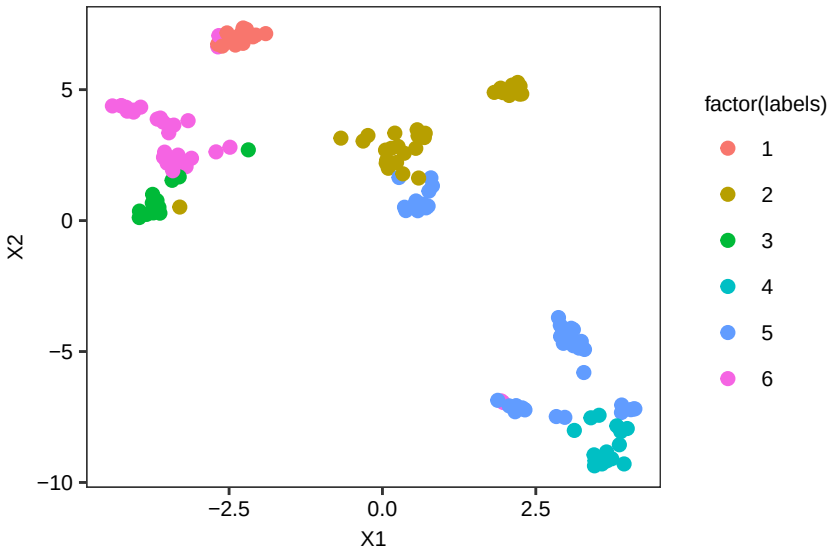
The first plot will show the eigenvalues where the greatest gap in used to decide K, the second plot shows the results from running UMAP on the diffused affinity matrix.

```
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.
```



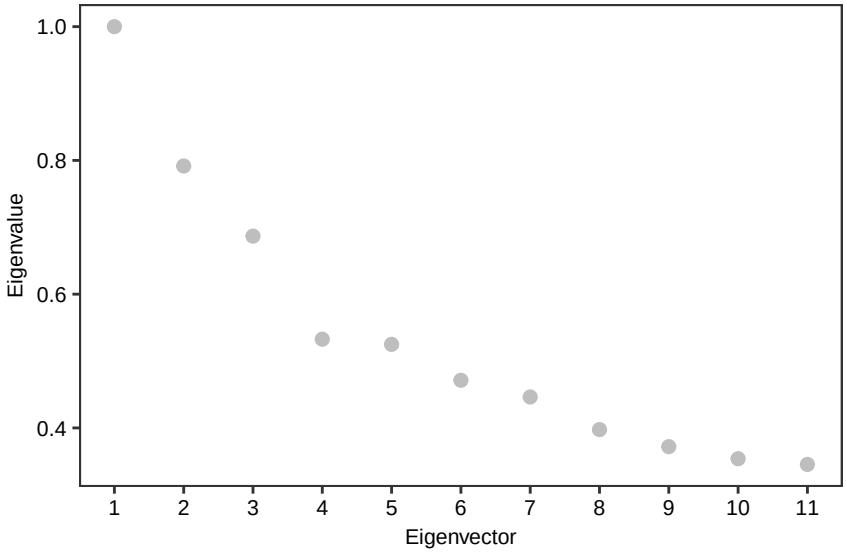
4. Multi-view clustering: Brain cancer multi-omics

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 framework to combine heterogeneous data sources.

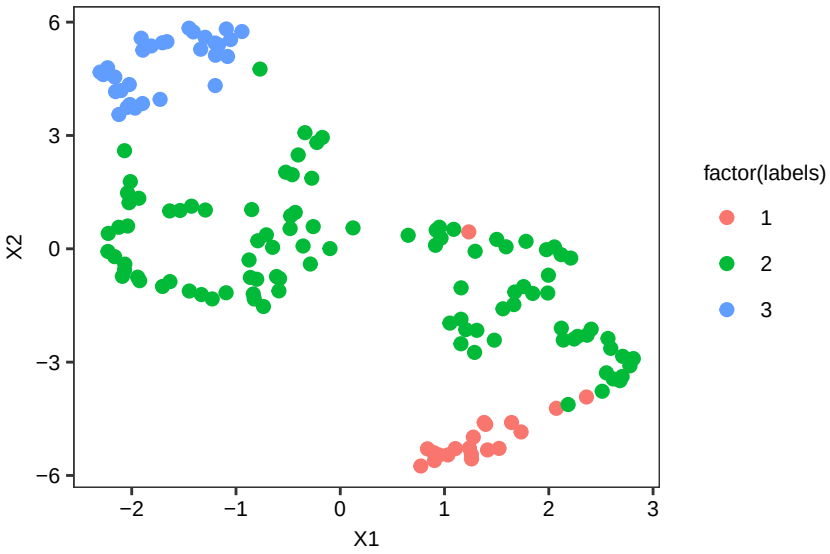
The first plot will show the eigenvalues where the greatest gap in used to decide K, the second plot shows the results from running UMAP on the combined affinity matrix. Running UMAP or t-SNE on the integrated affinity matrix provides a new way of multi-omic data visualisation.

```
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

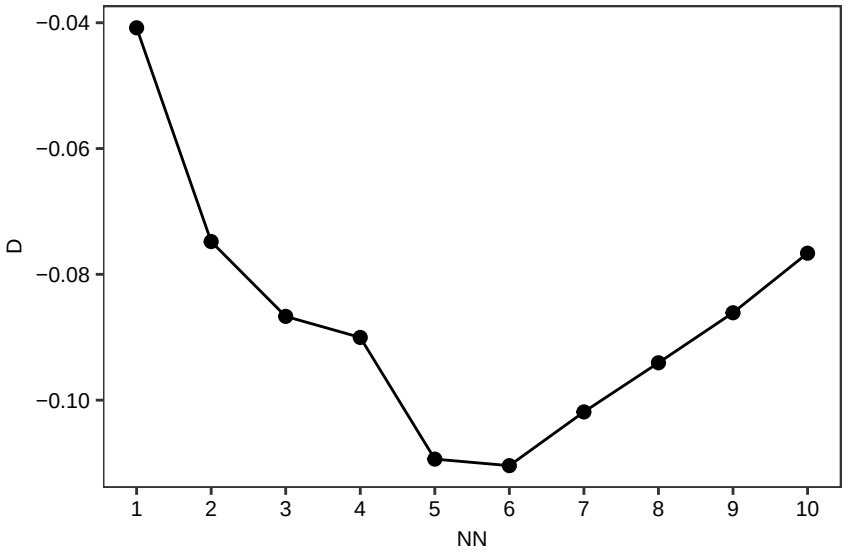
For non-Gaussian complex structures it is much better to use our method that examines the multimodality of the eigenvectors of the data's graph Laplacian and searches for the last substantial drop. This method also tunes the kernel by examining the multimodality. This method can handle Gaussian clusters too and multimodality is quantified using the well known dip-test statistic.

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 looks for a minimum (corresponding to the maximum drop in multimodality) to find the best kernel. In the next plot, we have the dip test statistics (Z) which measure the multimodality of the eigenvectors.

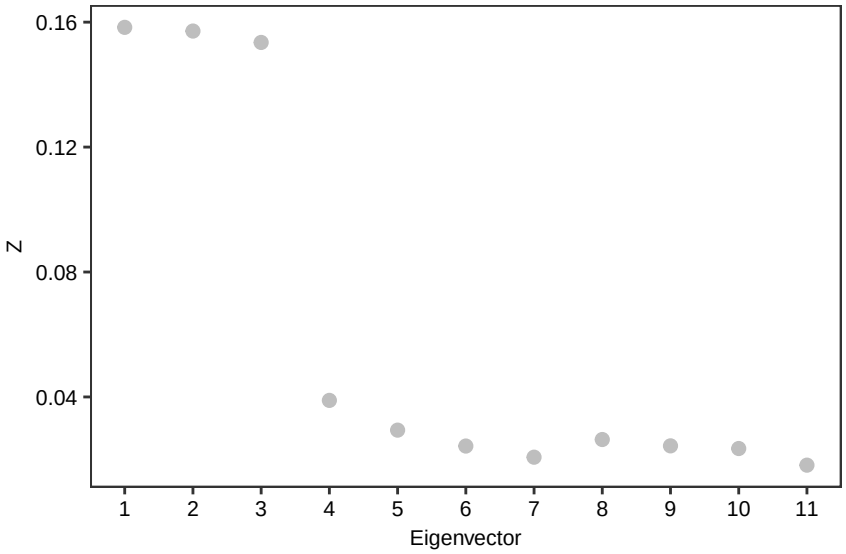
When the multimodality is higher the eigenvectors are more informative, so when there is a big gap, this may correspond to the optimal K. However, Spectrum has its own algorithm to search for the last substantial gap instead of the maximum. As searching for the greatest gap sometimes gets stuck in local minima. The last plot is PCA to visualise the data run just on the original data for the user, and is overlayed with cluster assignments.

```
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
```

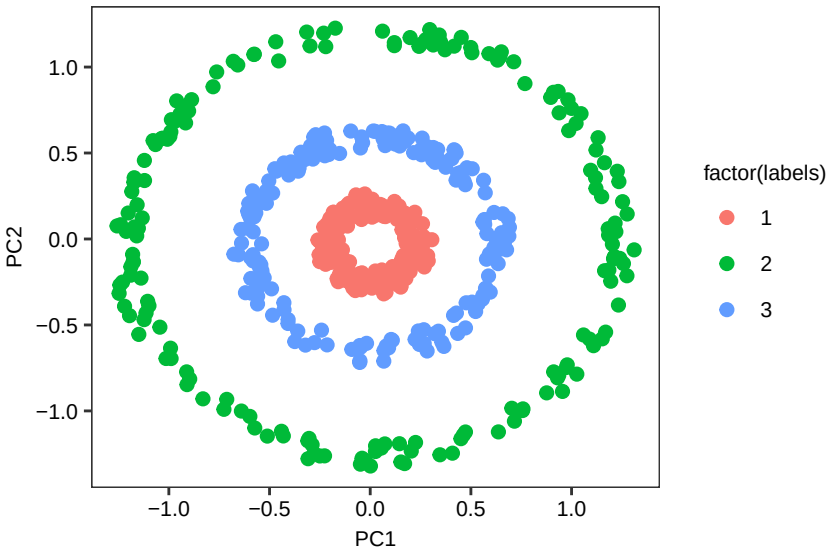
```
#> tuning kernel NN parameter: 10
#> optimal NN:6
#> done.
#> combining kernels if > 1 and making KNN graph...
#> done.
#> diffusing on tensor graph...
#> done.
#> calculating graph laplacian...
#> getting eigendecomposition of L...
#> done.
#> examining eigenvector distributions to select K...
#>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




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



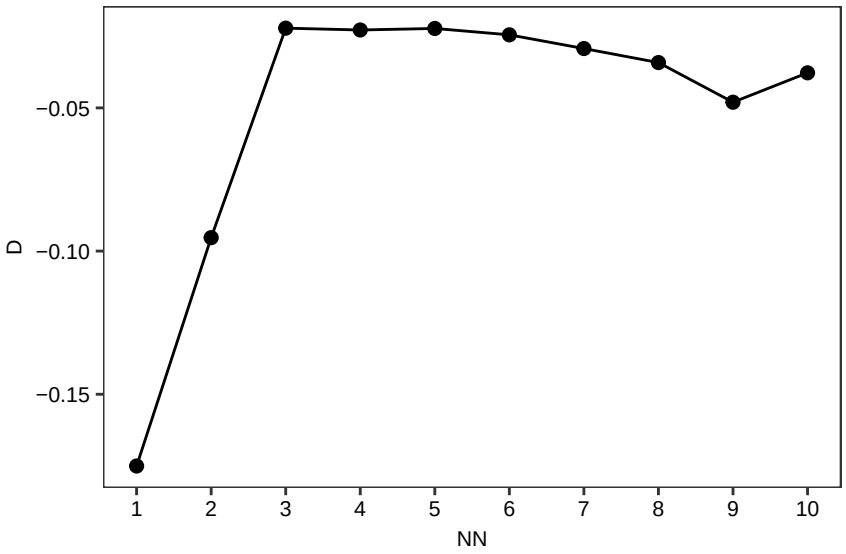

```
#> finished.
```



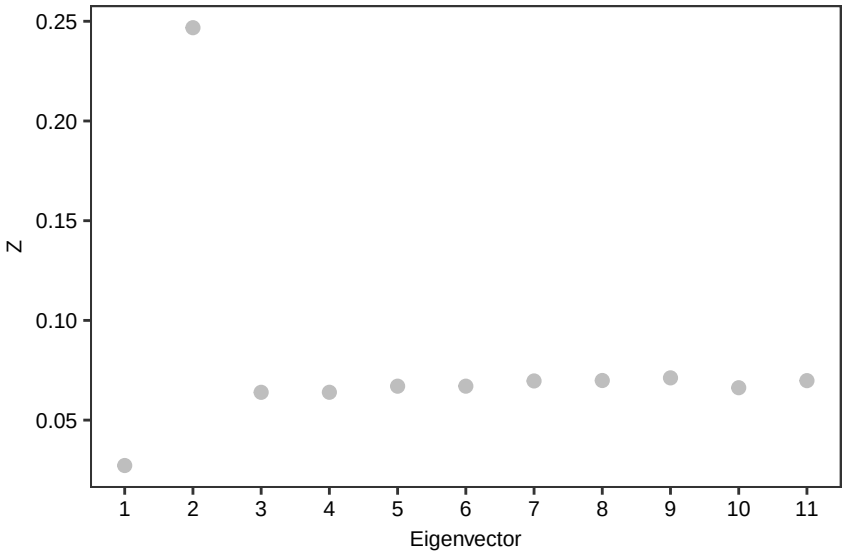
6. Single-view clustering: Non-Gaussian data, spirals

Same as the last example, but for the spirals dataset.

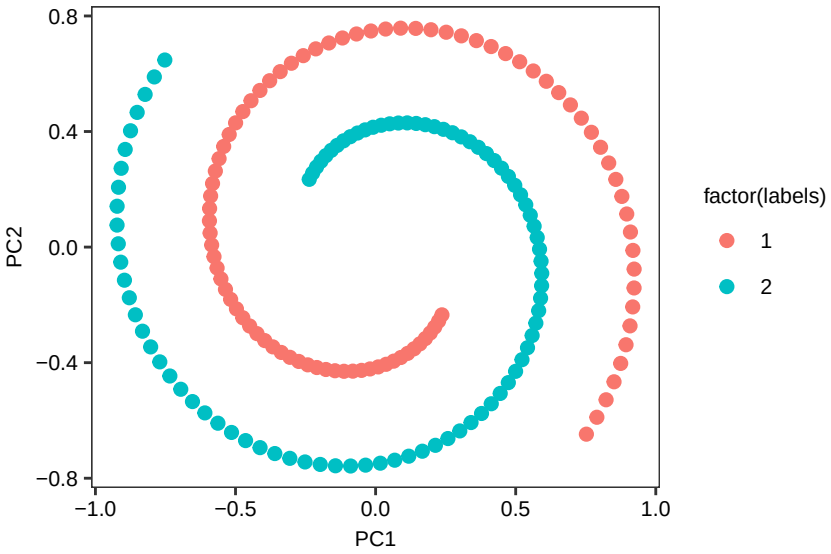
```
library(Spectrum)
test5 <- Spectrum(spirals,showpca=TRUE,method=2,fontsize=8,dotsize=2)
#> ***Spectrum***
#> detected views: 1
#> method: 2
#> kernel: density
#> calculating kernel 1
#> finding optimal NN kernel parameter by examining eigenvector distributions
#> tuning kernel NN parameter: 1
#>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

Spectrum is an advanced spectral clustering method automatically suitable for a wide range of data with its ability to self-tune to the data. Spectrum also has its own modifications for user customisation, as it has two kernels to choose from (adaptive density aware or self tuning spectral clustering), and two methods for selecting K (eigengap and multi-modality gap). The kernel parameters can be adjusted manually, but we recommend leaving them to default as Spectrum usually performs well automatically.

Included is a range of data visualisation options (t-SNE, kernel PCA, PCA, UMAP) for the cluster structure that can be run on the similarity matrix output or the original data, refer to the manual for more details.