

# SamSPECTRAL: A Modified Spectral Clustering Method for Clustering Flow Cytometry Data

Habil Zare and Parisa Shooshtari

March 30, 2012

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>How to run SamSPECTRAL?</b>	<b>2</b>
2.1	An example . . . . .	3
<b>3</b>	<b>Adjusting parameters</b>	<b>4</b>
3.1	Example . . . . .	5
<b>4</b>	<b>Reference</b>	<b>11</b>

# 1 Introduction

Data analysis is a crucial step in most of recent biological research areas such as microarray techniques, gene expression and protein classification. A classical approach for analysing biological data is to first group individual data points based on some similarity criterion, a process known as clustering, and then compare the outcome of clustering with the desired biological hypotheses. Spectral clustering is a non-parametric clustering method which has proved useful in many pattern recognition areas. Not only it does not require a priori assumptions on the size, shape and distributions of clusters, but it has several features that make it an appropriate candidate for clustering biological data:

- It is not sensitive to outliers, noise or shape of clusters.
- It is adjustable so we can make use of biological knowledge to adapt it for a specific problem or dataset.
- There is mathematical evidence to guarantee its proper performance.

However, because of the machine limitations, one faces serious empirical barriers in applying this method for large data sets. SamSPECTRAL is a modification to spectral clustering such that it will be applicable on large size datasets.

## 2 How to run SamSPECTRAL?

SamSPECTRAL is an R package source that can be downloaded from Bio-Cunductor. In Linux, it can be installed by the following command:

```
R CMD INSTALL SamSPECTRAL_x.y.z.tar.gz
```

where x.y.z. determines the version.

The main function of this package is `SamSPECTRAL()` which is loaded by using the command `library(SamSPECTRAL)` in R. Before running this function on a data set, some parameters are required to be set including: `normal.sigma` and `separation.factor`. This can be best done by running the

algorithm on some number of samples (Normally, 2 or 3 samples are sufficient). Then the function `SamSPECTRAL()` can be applied to all samples in that data set to identify cell populations in each sample data.

## 2.1 An example

This example shows how `SamSPECTRAL` can be run on flow cytometry data. If  $f$  is a flow frame (which is normally read from an FCS file using `flowCore`), then the object "small" in the following example should be replaced by `expr(f)`.

```
> library(SamSPECTRAL)
> data(small_data)
> full <- small
> L <- SamSPECTRAL(full,dimension=c(1,2,3),normal.sigma = 200,separation.factor=1)
> plot(full, pch='.', col= L)
```

`SamSPECTRAL` is done. The results are in  $L$ , a vector that provides a numeric label for each event. All events with equal label are in one component and isolated outliers are labelled by `NA`. The following piece of code is not a part of the analysis and it is included only for more clear presentation of the results. The code computes the frequency of events in each component and adds a legend to the figure.

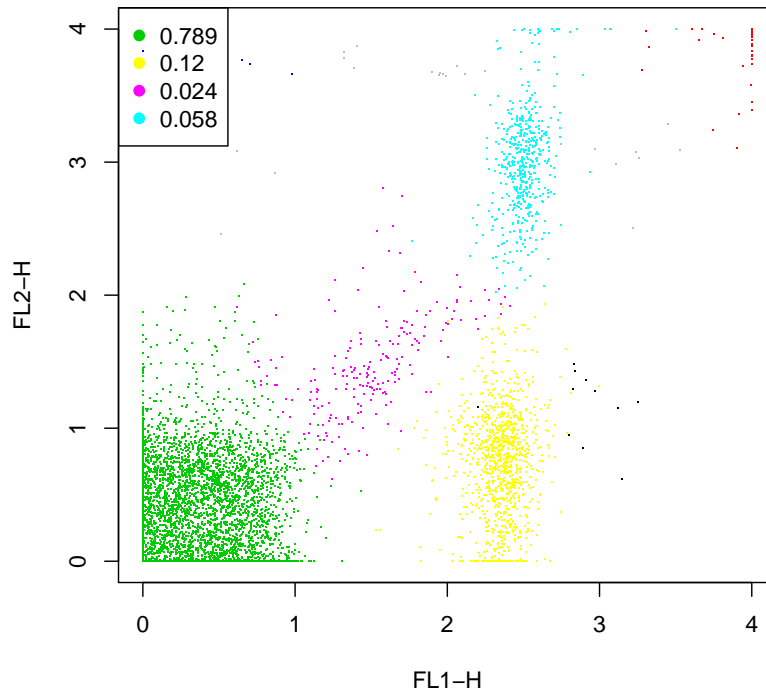
```
> # Computing the frequency:
> plot(full, pch='.', col= L)
> frequency <- c()
> minimum.frequency <- 0.01
> # components smaller than this threshold, will not be appear
> frequency.large <- c()
> labels <- as.character(unique(L))
> for(label in labels){
+   if(!is.na(label)){
+     frequency[label] <- length(which(L==label))/length(L)
+     if(frequency[label] > minimum.frequency)
+       frequency.large[label] <- frequency[label]
+   }
+ }
> print(frequency)
```

```

      3          7          6          5          8          2
0.7887777778 0.1196666667 0.0238888889 0.0577777778 0.0026666667 0.0052222222
      1          4
0.0012222222 0.0004444444

> # Adding legend
> legend(x="topleft",as.character(round(frequency.large,3)),col=names(fr

```



### 3 Adjusting parameters

For efficiency, one can set  $m = 3000$  to keep the running time below 1 minute by a 2 GHz processor and normally the results remained satisfactory for flow cytometry data. The separation factor and scaling parameter ( $\sigma$ ) are two main parameters that needed to be adjusted. The general way is to run SamSPECTRAL on one or two random data samples of a flow cytometry data

set and try different values for  $\sigma$  and separation factor. Then, the selected parameters were fixed and used to apply SamSPECTRAL on the rest of data samples. An efficient strategy is explained by the following example.

### 3.1 Example

First we load data and store the transformed coordinates in a matrix called *full*:

```
> data(small_data)
> full <- small
```

The objects needed for creating this vignette can be directly computed or loaded from previously saved workspace to save time. The later increases the speed of building this vignette.

```
> run.live <- FALSE
```

The following parameters are rarely needed to be changed for flow cytometry data:

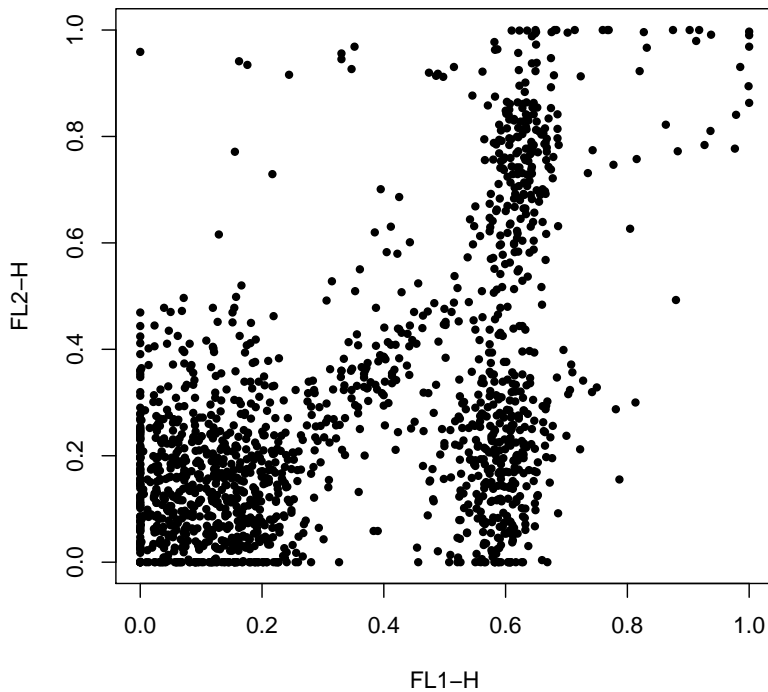
```
> # Parameters:
> m <- 3000;
> community.weakness.threshold <-1; precision <- 6;
> maximum.number.of.clusters <- 30
```

The following piece of code, scales the coordinates in range [0,1]:

```
> for (i.column in 1:dim(full)[2]){#For all columns
+   ith.column <- full[,i.column]
+   full[,i.column] <- (ith.column-min(ith.column)) / (max(ith.colu
+ }#End for (i.column.
> # Therefore,
> space.length <- 1
```

To perform faithful sampling, we run:

```
> # Sample the data and build the communities
> society <- Building_Communities(full,m, space.length, community.weakness.t
> plot(full[society$representatives, ], pch=20)
```

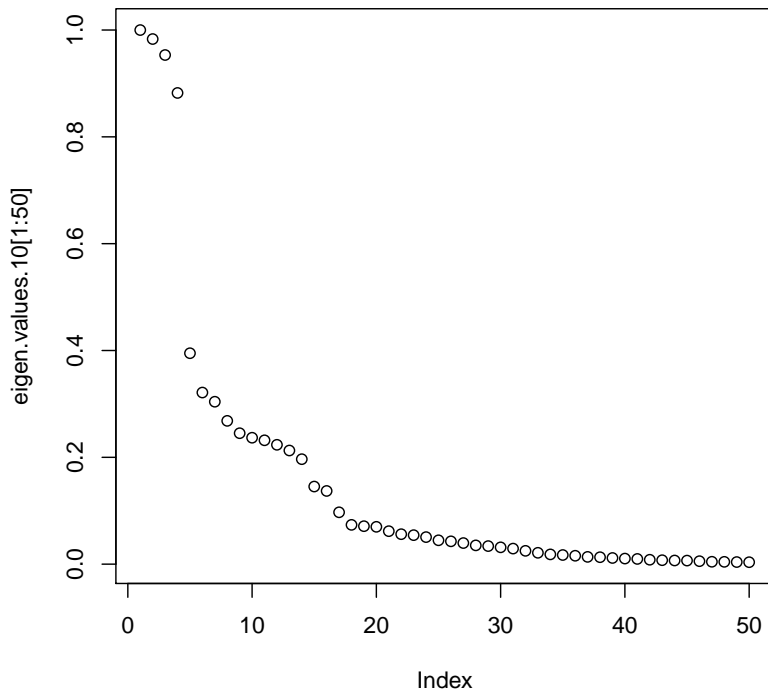


We intend to first find an appropriate value for  $\sigma$  and then set separation factor. Note that  $\text{normal.sigma} = \frac{1}{\sigma^2}$ , therefore, decreasing  $\text{normal.sigma}$  is equivalent to increasing  $\sigma$  and visa versa. We start with  $\text{normal.sigma} = 10$ :

```

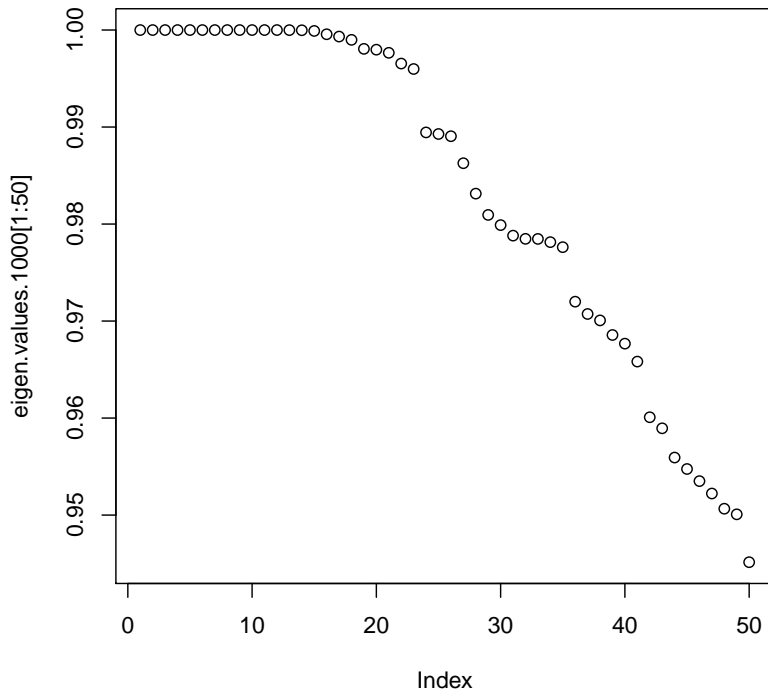
>     normal.sigma <- 10
>     # Compute conductance between communities
>     conductance <- Conductance_Calculation(full, normal.sigma, space.length, s
>     # Compute the eigenspace:
>     if (run.live){
+         clust_result.10 <- Civilized_Spectral_Clustering(full, maximum.num
+             eigen.values.10 <- clust_result.10@eigen.space$values
+     } else
+         data("eigen.values.10")
>     plot(eigen.values.10[1:50])

```



We observe that the eigen values curve does not have a "knee" shape. So we increase sigma:

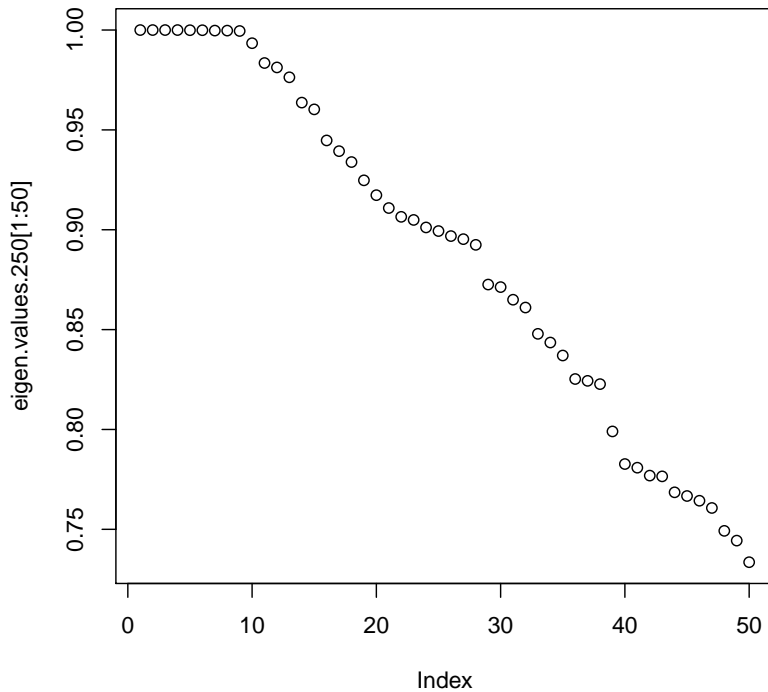
```
>     normal.sigma <- 1000
>     # Compute conductance between communities
>     conductance <- Conductance_Calculation(full, normal.sigma, space.length, s
>     # Compute the eigenspace:
>     if (run.live){
+         clust_result.1000 <- Civilized_Spectral_Clustering(full, maximum.n
+             eigen.values.1000 <- clust_result.1000@eigen.space$values
+     } else
+         data("eigen.values.1000")
>     plot(eigen.values.1000[1:50])
```



We observe that in the eigen values plot, "too many" values are close to 1 but for this example we do not expect 20 populations. So we decrease sigma:

```
>     normal.sigma <- 250
>     # Compute conductance between communities
>     conductance <- Conductance_Calculation(full, normal.sigma, space.length, s
>     # Compute the eigenspace:
>     clust_result.250 <- Civilized_Spectral_Clustering(full, maximum.number.of.
>     eigen.values.250 <- clust_result.250@eigen.space$values
>     plot(eigen.values.250[1:50])
```





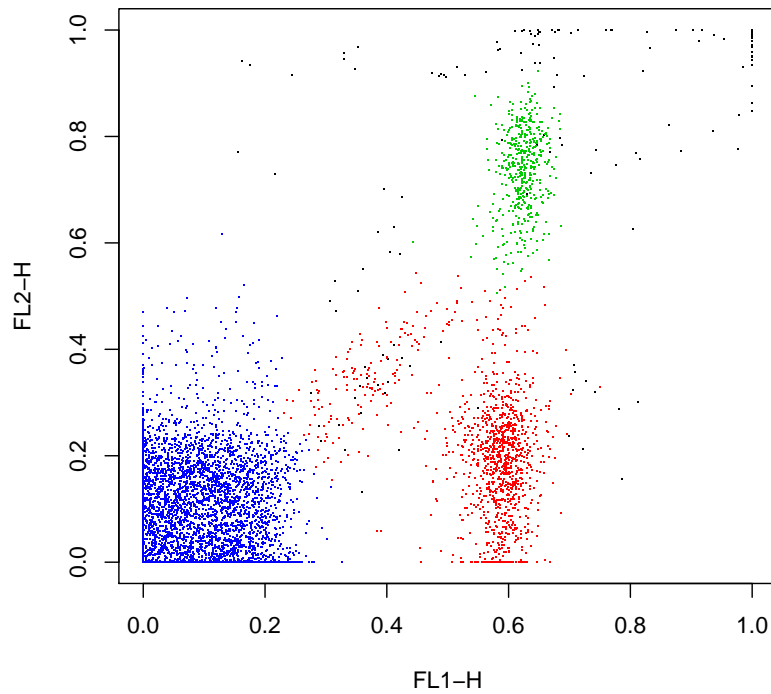
This is "a right" value for normal.sigma because the curve has now a knee shape. Even some variation to this parameter does not change the shape significantly (200 or 300 can be tried).

Now having sigma been adjusted, separation factor can be tuned:

```

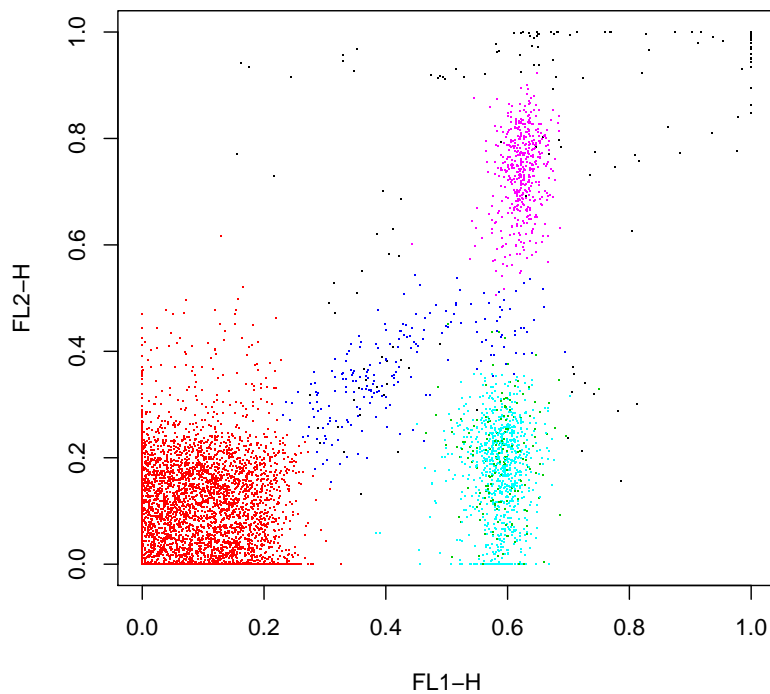
> # Extracting labels:
> labels.for_num.of.clusters <- clust_result.250@labels.for_num.of.clusters
> number.of.clusters <- clust_result.250@number.of.clusters
> L33 <- labels.for_num.of.clusters[[number.of.clusters]]
> # Setting septation factor:
> separation.factor <- 0.1
> # post-processing:
> component.of <- Connecting(full, society, conductance, number.of.clusters,
> # plotting:
> plot(full, pch='.', col= component.of)

```



This value is too small for the separation factor and a population is combined by mistake. Therefore, we increase separation factor to separate the components more:

```
> # Setting separation factor:
> separation.factor <- 0.5
> # post-possesing:
> component.of <- Connecting(full, society, conductance, number.of.clusters,
> # plotting:
> plot(full, pch='.', col= component.of)
```



This is the right value for separator factor as all population are now separated.

Now, we can fix these values for the parameters; `normal.sigma=250` and `separation.factor=0.5`. One can run the SamSPECTRAL algorithm on the rest of the data set without changing them, hopefully, obtaining as appropriate results.

## 4 Reference

Zare, H. and Shooshtari, P. and Gupta, A. and Brinkman R.B: **Data Reduction for Spectral Clustering to Analyse High Throughput Flow Cytometry Data.** *BMC Bioinformatics*, 2010, **11**:403.