X-ray diffraction Data Clustering

Jie Yong

June 21, 2016

With the help of modern computers, scientific data acquisition is much faster than before and scientists have accumulated tremendous amount of data. While Some data are useful but a large portion of the data is redundant. Clustering can group similar data into the same group thus reduce time and rescourses for scientists to gain insight on huge amount of data.

In this example, we will do clustering on Fe-Ga-Pd ternary X-ray diffraction spectra. Each spectrum is taken for slightly different composition of this ternary compound. We want to know what the typical patterns which corresponds to different phases are.

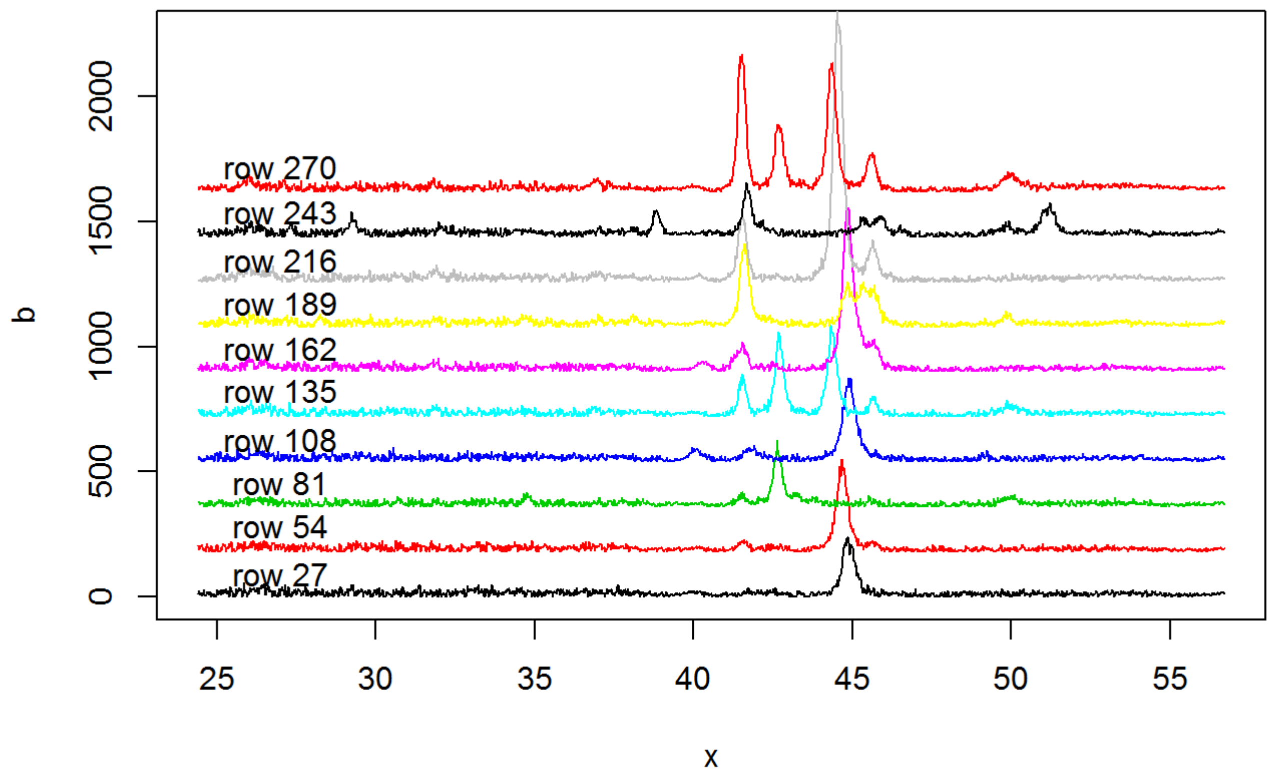
The work is done with collobaration of:

- Gilad Kusne, Research Scientist, National Institute of Science and Technology
- Ichiro Takeuchi, Professor of Materials Science and Technology in University of Maryland.

First, load the X-ray diffraction data and their corresponding composition data.

```
XRD <- read.table("data/FeGaPd_Exp_PP.txt",header=TRUE)
#Load composition data
com <- read.table("data/FeGaPd_Exp_CMP.txt",header=TRUE)
x <- seq(24.40,56.70, by=0.02)
```

The following diaplay some sample data:



```
## png
##   3
```

```
## png
##   2
```

Fo how X-ray diffraction works, please refer to this Wikipedia article: article (https://en.wikipedia.org/wiki/X-ray_crystallography)

The total number of spectra is 278. It is too large for scientists to handle. Let us say we want to reduce this number to less than 10 (number of clusters).

The section sets the parameters for different clustering methods. You can change it ro play with different parameters.

```
cluster_No = 5
method="euclidean" # choose distance metric
```

The section will make a plot function so we can easily visualize the clustering results.

```r
plot_results = function (cluster_No, results, titles){
  triangle_X=1.00-com[ ,2]/2-com[ ,1]
  triangle_Y=0.866*com[ ,2]
  plot(triangle_X, triangle_Y, col=rainbow(cluster_No)[results],
       pch=19, cex=1, xlim=c(-0.05,0.60), ylim=c(-0.05,0.55))

  segments(0, 0, 0.3, 0.6*0.866)
  segments(0, 0, 0.60, 0)
  segments(0.60, 0,0.3, 0.6*0.866)
  text(0, -0.03, colnames(com[1]), cex=1)
  text(0.6, -0.03, colnames(com[3]), cex=1)
  text(0.3, 0.44, colnames(com[2]), cex=1)
  title_name = paste(titles)
  text(0.3, 0.55, title_name, cex=0.9)
}
```

Distance Metric

This section Calculates different distance metric: Euclidean distance and cosine distance.

```r
library(stylo)
XRD_euclidean_dist=dist(XRD, method)
#cosine distance
XRD_table <- t(XRD)%*%as.matrix(XRD)
XRD_cosine_dist = dist.cosine(XRD_table)
```

Clustering

We will Use k-medoidos, Hierarchical Clustering or spectral clustering for clustering methods

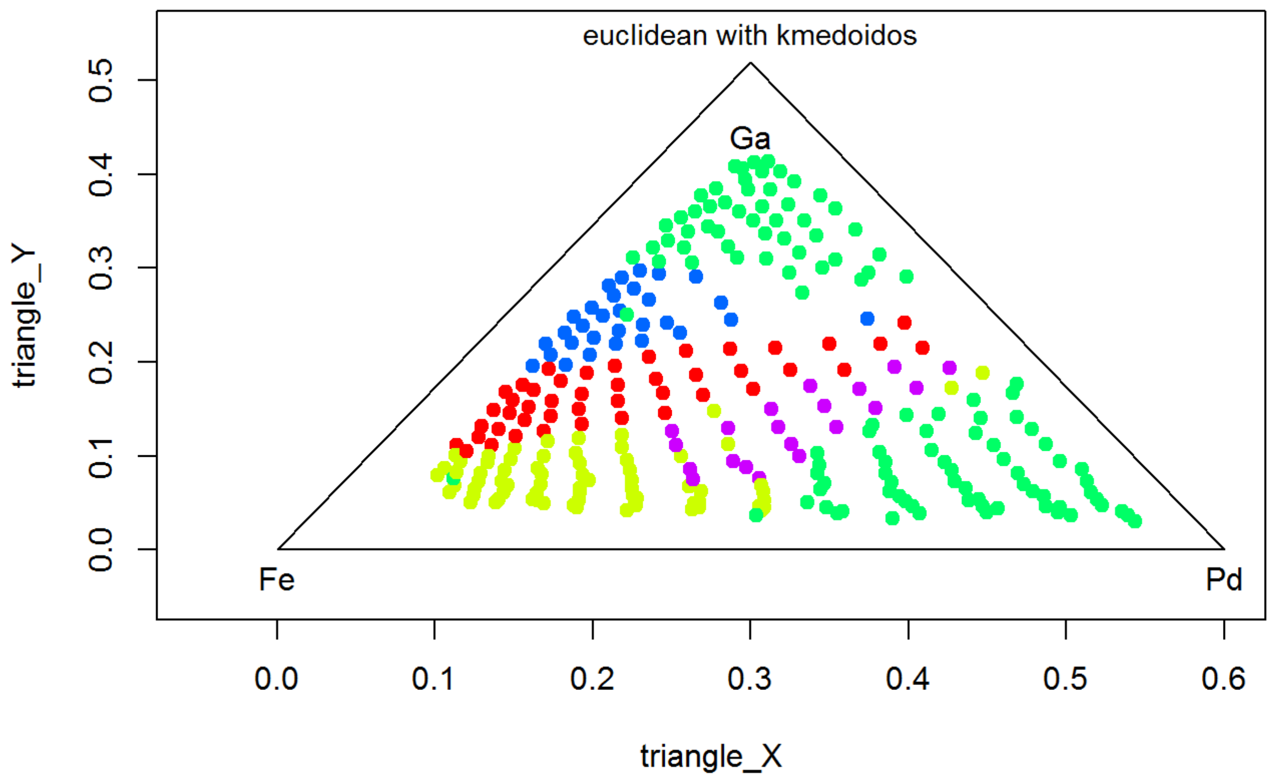
```r
library(cluster)
library(kernlab)
#k-medoidos
results_kmedoidos = pam(XRD_euclidean_dist, cluster_No)$cluster
results_kmedoidos_cosine = pam(XRD_cosine_dist, cluster_No)$cluster
#hc
hc=hclust(XRD_euclidean_dist, method = "ward.D2")
results_hc <- cutree(hc, k = cluster_No)
# spectral clustering
results_spec <- specc(XRD, centers = cluster_No)
```

Clustering Visualization

Finally, we diaplay the clustering results.

This one shows euclidean distance with kmedoidos clustering.

```r
par(mfrow=c(1,1))
plot_results(cluster_No, results_kmedoidos, "euclidean with kmedoidos")
```



```
dev.copy(png,'figures/euclidean_with_kmedoidos.png')
```

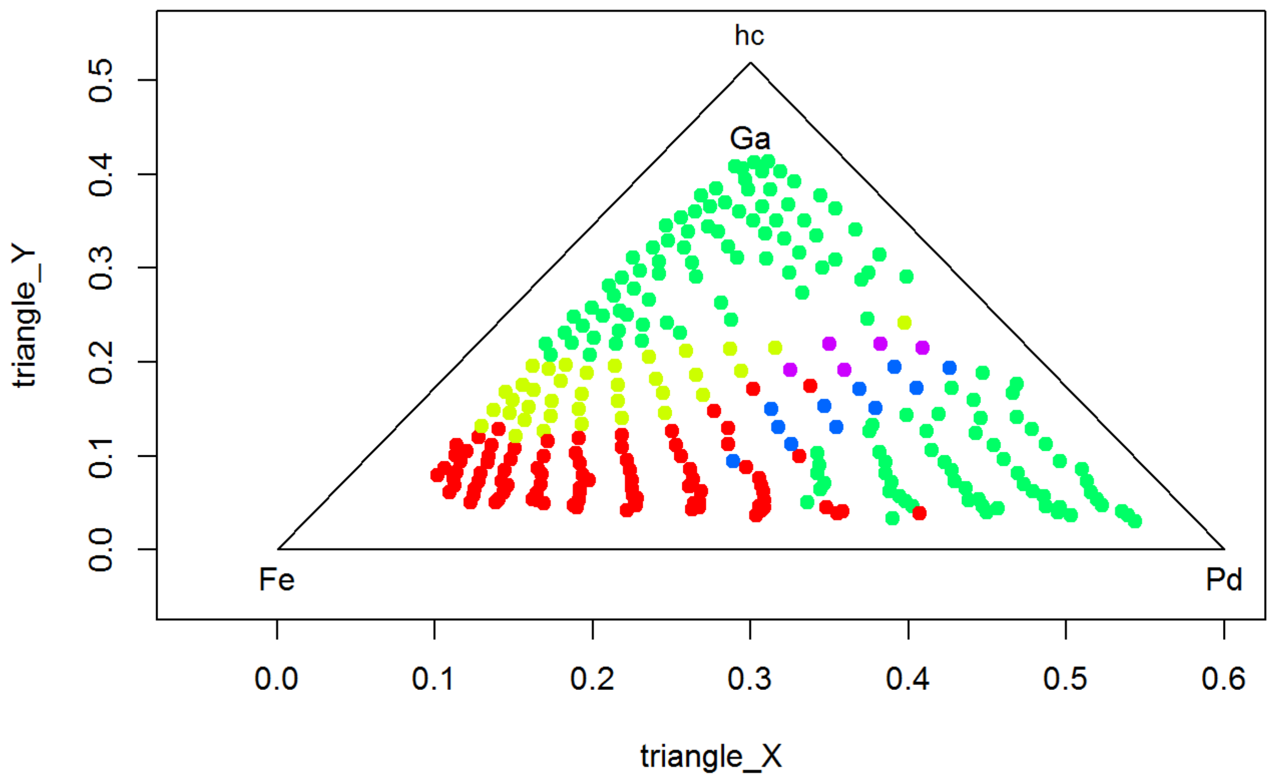
```
## png
##   3
```

```
dev.off()
```

```
## png
##   2
```

This one shows euclidean distance with Hierarchical Clustering.

```
par(mfrow=c(1,1))
plot_results(cluster_No, results_hc,"hc")
```



```
dev.copy(png,'figures/hc.png')
```

```
## png
##   3
```

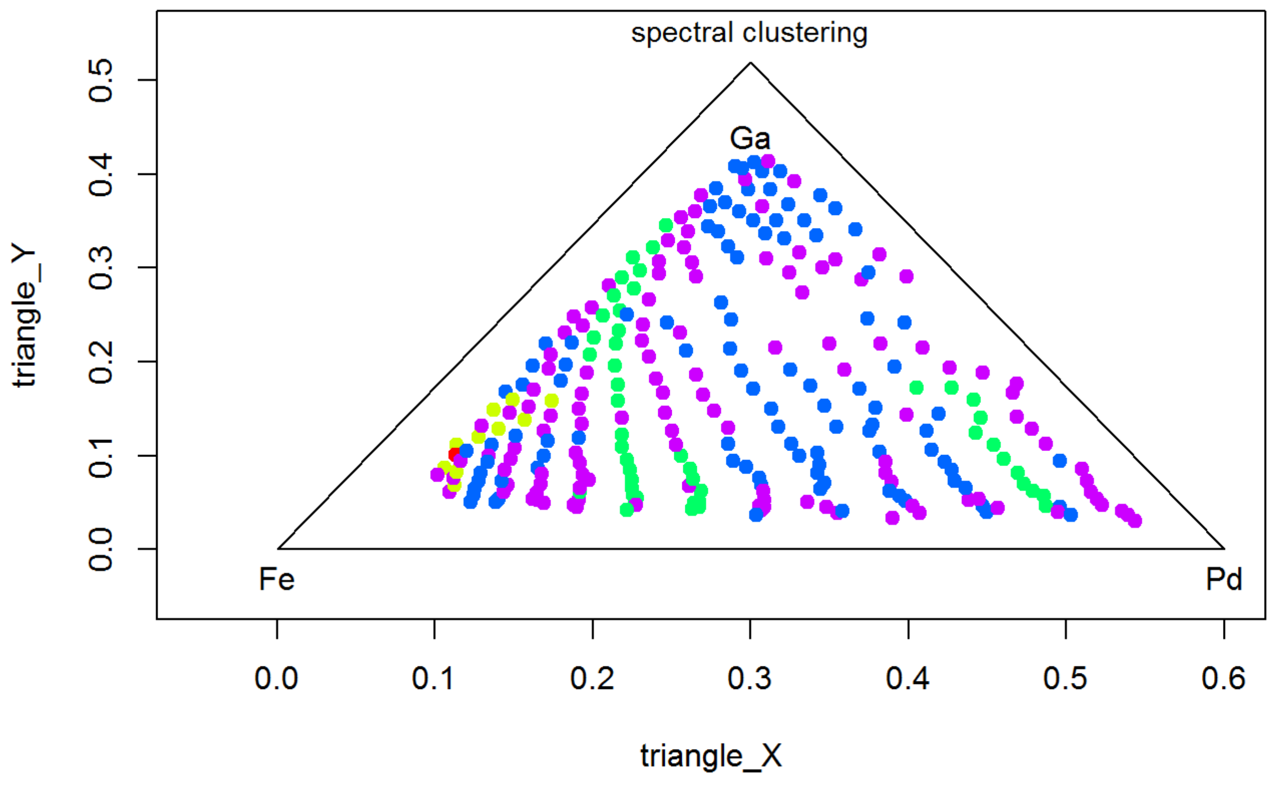
```
dev.off()
```

```
## png
##   2
```

```
par(mfrow=c(1,1))
```

This one shows spectral Clustering.

```
plot_results(cluster_No, results_spec,"spectral clustering")
```



```
dev.copy(png,'figures/spec.png')
```

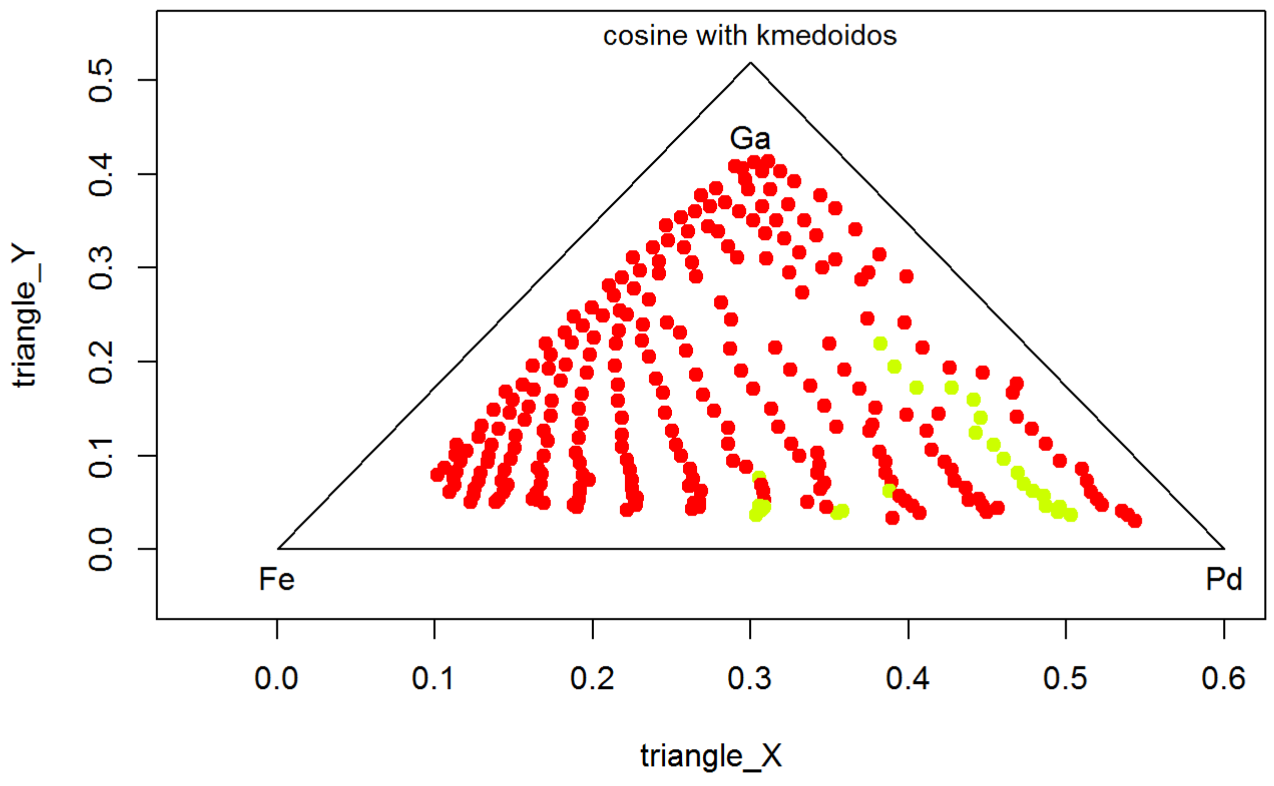
```
## png
##   3
```

```
dev.off()
```

```
## png
##   2
```

Last one is cosine distance metric with K-medoidos clustering.

```
plot_results(cluster_No, results_kmedoidos_cosine,"cosine with kmedoidos")
```



```
dev.copy(png,'figures/clustering.png')
```

```
## png
##   3
```

```
dev.off()
```

```
## png
##   2
```

```
par(mfrow=c(1,1))
```