Diluvian Clustering - A fast, effective algorithm for clustering compositional and other data

Nicholas W. M. Ritchie
National Institute of Standards and Technology[1]
Materials Measurement Science Division
100 Bureau Drive, Gaithersburg, MD 20899-8372
(301) 975-3929 nicholas.ritchie@nist.gov

Diluvian Clustering is an unsupervised grid-based clustering algorithm well suited to interpreting large sets of noisy compositional data. The algorithm is notable for its ability to identify clusters which are either compact or diffuse and clusters which have either a large number or a small number of members. Diluvian Clustering is fundamentally different from most algorithms previously applied to cluster compositional data in that its implementation does not depend upon a metric. The algorithm reduces in two-dimensions to a case for which there is an intuitive, real-world parallel. Furthermore, the algorithm has few tunable parameters. By eliminating the dependence on an explicit metric, it is possible to derive reasonable clusters with disparate variances like those in real-world compositional data sets. The algorithm is computationally efficient. While the worst case scales as $O(N^2)$ most cases are closer to $O(N)$ where N is the number of discrete data points. On a mid-range 2014 vintage computer, a typical 20 000 particle, 30 element data set can be clustered in a fraction of a second.

Introduction

Modern microscopes under computer automation can collect data sets consisting of thousands to millions of individual compositional data items. Two examples from the field of scanning electron microscopy with energy dispersive x-ray spectrometer (SEM-EDS) are automated particle analysis and x-ray spectrum imaging. The custom automated particle analysis system at the National Institute of Standards and Technology (NIST) is based on a TESCAN MIRA3 LMU with four 30 mm^2 Pulsetor silicon drift detectors[2] mounted at approximately 90 degree azimuthal increments. This configuration produces a summed output count rate (stores) of approximately 200 000 cps at 25 keV and 1 nA on bulk copper. The four spectra are typically

---

[1] Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

[2] DISCLAIMER: Certain commercial equipment, instruments, or materials are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

summed before quantification for particle work as summing tends to mitigate many absorption related issues with complex shaped samples. This system is capable of collecting about 10 000 unique particle spectra per hour under automation. Newbury [Newbury, 2005] has demonstrated useful x-ray spectrum images consisting of 300 000 point spectra in approximately 3 minutes. Spectrum images consisting of over 10^9 discrete spectra have recently been demonstrated [Vandecreme, 2014].

With tools like these, it is easy to generate data sets which stretch the limits of human patience and capacity to analyze and interpret using traditional methods. X-ray spectra consist of hundreds of thousands of individual x-ray measurements yet we are able to reduce this abundance of data into a handful of numbers which represent the composition. Similarly, x-ray spectrum data sets represent large collections of individual measurements which need to be reduced to facilitate interpretation. Instead of composition, the relevant quantities are the populations of compositionally similar spectra where the definition of similarity is determined by the homogeneity of the materials, the quality of the spectra and other sample dependent characteristics. At NIST, we have developed tools to help interpret spectra and tools to help interpret large spectrum data sets.

The most popular techniques for analyzing spectrum images and other large compositional data sets include visualization tools like x-ray spectrum images [Mott, 1995], statistical tools like Sandia's AXSIA [Kotula, 2003] (commercially available as Thermo Scientific's COMPASS tool) and the EM algorithm [Dempster, 1977], hierarchical clustering tools [Wilson, 2012], non-hierarchical clustering tools like k-means [MacQueen, 1967] and supervised learning models like support vector machines [Cortes, 1995]. Each of these techniques has strengths and weaknesses and have been used with varying degrees of success by this author and others.

This article proposes a novel algorithm called Diluvian Clustering that has significant advantages over each of these algorithms in certain data domains. The algorithm is fast, robust and consistent. It requires minimal configuration and produces clusters similar to those a human might identify manually. This algorithm is similar to the classic GRIDCLUS algorithm [Schikuta, 1996]. It goes beyond GRIDCLUS by providing a mechanism to split clusters based on variations in bin density. The name "Diluvian Clustering" is, for reasons that will become clear, inspired by the Biblical myth of the great flood that covered all the lands of the earth before receding to reveal the olive branch.

Method

The Diluvian Clustering algorithm will be applied to a set of discrete micro-particle data collected on a scanning electron microscope with an energy dispersive x-ray spectrometer. The particle data is quantified against measured elemental standard spectra using a filter-fit linear least squares algorithm [Schamber 1977, Goldstein 2003]. The resulting k-ratios are normalized to mitigate particle related effects such as variation in intensity due to mass and shape. Since the data is collected from micro-particles, it is inappropriate to apply bulk matrix correction algorithms (ZAF or $\varphi(\rho z)$)[Goldstein 2003] to compensate for differences in backscatter, energy loss, absorption, and secondary fluorescence due to material differences.

Alternatively, algorithms like principal component analysis (PCA) or multivariate curve resolution (MCR) could be used to perform dimensionality reduction. However, quantifying the data against measured standards has the advantage that the resultant clusters can be readily interpreted in mean, in variance and as individual particle data items.

Identifying the elements present in a set consisting of hundreds or thousands of spectra can be a challenge. Tools like the max-pixel spectrum [Bright, 2004] can summarize large sets of spectra quickly and help to identify the elements present. The max-pixel spectrum is derived from a set of spectra. Each channel in the derived spectrum contains the most intense value present at that channel in any spectrum in the set. The resulting spectrum shows all the characteristic peaks present in any spectrum. Normalizing the spectra to a fixed total number of counts before adding them to the max-pixel spectrum helps to mitigate mass effects in particle spectra. We refer to the max-pixel spectrum computed from normalized particle spectra as the *max-particle spectrum*.

To verify the identifications, we have extended the concept of the max-pixel spectrum to the max-residual spectrum. The max-residual is like the max-pixel except the input spectra are not the individual raw spectra but instead the residual spectrum. The residual is computed from the original spectrum during the fit process by subtracting each element's k-ratio scaled characteristic peak shape. The max-residual is particularly sensitive to missed elements. In practice, the max-residual spectrum can highlight a missing element even when that element interferes with other elements.

Usually, the process of identifying elements and quantifying the spectra is iterative. First, the max-pixel spectrum is calculated and all likely elements are identified. Then standard spectra are fit to each spectrum in the data set, the residuals are computed, the max-residual is computed and the max-residual is examined for omitted elements. If an elements is identified in the max-residual, a standard for this element is added to the fit. The process is repeated until the max-residual consists of continuum and Poisson noise. In practice, this ideal is rarely achieved. Slight chemical shifts and detector calibration shifts sometimes lead to slight s-curves under the characteristic peaks in the residual spectra. The s-curves then also appear in the max-residual. However, these artifacts in the max-residual spectrum are in practice easily differentiated from missed elements. The max-pixel and max-residual spectra help to ensure all major and minor elements are identified even if an element is present in only one spectrum.

The result of the quantification is a table of data in which each row represents a particle or pixel and each column represents the relative quantity of an element. The dimensionality of this table may still be high. Some fly ash samples may contain 30 or more distinct elements and tens of thousands of individual micro-particles.

At this point, we can ignore the fact that we are working with compositional data and simply consider the data to be a matrix of numbers. The i-th particle is represented by a vector x_i consisting of M values $x_{i,k}$. There is an entire sub-field in computer science dedicated to finding patterns in matrices – data mining [Aggarwal 2014, Gan 2007].

After extensive evaluation of classic data mining algorithms, we discovered that they typically fail to handle our large particle data sets well. The ideal algorithm would have a minimum number of adjustable parameters and would be able to identify clusters with membership ranging from one to many. The algorithm should produce clusters similar to careful manual inspection.

Classic algorithms, like k-means, use a metric function to quantify the distance between data points. Often one of a class of metrics called the Minkowski [Aggarwal2014, Gan 2007] metrics are used.

$$d_p(x_i, x_j) = \left[\sum_{k=1}^{d} \left|x_{i,k} - x_{j,k}\right|^P\right]^{\frac{1}{P}}$$

When P=1, the Minkowski metric is often called the L1-norm or the Manhattan distance. When P=2, the metric is called the L2-norm or the Euclidean distance. In the limit of $P \to \infty$, the metric is equivalent to the maximum value of $\left|x_{i,k} - x_{j,k}\right|$. The k-means algorithm can cluster compositional data but it has unfortunate shortcomings. For one, the user must specify k, the number of clusters, a priori. The algorithm divides the data into k clusters in a manner that locally minimizes the sum Minkowski metric. There is a pseudo-random character to k-means clustering. Reapplying the algorithm to the same data set is likely to produce different results each time. There are algorithms which extend the k-means by varying k to find some optimal number of clusters. However, the algorithm typically is good at identifying major clusters but very poor at identifying distinct low population clusters.

Hierarchical clustering successively bifurcates the data into clusters that maximize the distinction between clusters. Again, the Minkowski metric is typically used. After sufficient bifurcations, each data item is its own unique cluster. Hierarchical clustering can help to identify the clusters that an operator might discern manually.

The proposed Diluvian Clustering algorithm is very different from metric-based clustering algorithms in that it starts by binning the data in a multi-dimensional histogram. Each dimension in the data is represented by a dimension in the histogram. The bin widths in each dimension can be defined consistently or independently. The width of the bin can be selected based on the anticipated spread in the data. This can be determined by measuring a homogeneous class of particles and computing the standard deviation for each element/column.

In D-dimensions with M bins per dimension, there are M^D distinct bins. This number can grow exceedingly large. For 30 dimensions with 10 bins per dimension, the number of bins (10^{30}) would seem to be unmanageable except that almost all are empty. If there are N data points, then at most N bins will be non-empty. The histogram can be recorded efficiently in a sparse format.

The clustering algorithm proceeds to group adjacent histogram bins into islands. In one dimension, it is trivial to define adjacency. As shown in Figure 1, the bins adjacent to the i-th bin are the two bins i+1 and i-1. In two dimensions (i, j), there are 4 bins that differ by one in one index ($(i+1, j)$, $(i-1, j)$, $(i, j+1)$, $(i, j-1)$) plus an additional 4 bins ($(i+1, j+1)$, $(i+1, j-1)$, $(i-1, j+1)$, $(i-1, j-1)$) that differ by one in two indices. Call the first 4 bins the 1-adjacent bins and the second set the 2-adjacent bins. In general, d-adjacency can be defined as those bins that differ by one in d or fewer indices and are equal in all the rest. In general, there are

$$A(D, d) = \sum_{m=1}^{d} 2^m C(D, m)$$

bins that are d-adjacent in D-dimensions where C(n,k) is the binomial coefficient "n choose k". For the extreme cases, d=1 and d=D, $A(D, 1) = 2\,D$ and $A(D, D) = 3^D - 1$. Thus the order of adjacency can determine whether the number of adjacent bins grows linearly with D or exponentially. The distance from the initial bin to the furthest d-adjacent bin increases as $\sqrt{d}$.

Regardless which definition of adjacency selected, the fraction of adjacent bins decreases with D. If there are N bins in each of D dimensions then there are a total of N^D bins of which a maximum fraction of ${(3^D - 1)}/{N^D}$ are d-adjacent. Thus although there would seem to be many more adjacent bins, the relative number of adjacent bins is decreasing with higher dimension. The likelihood of two randomly generated data points being located in adjacent bins becomes infinitesimally small as the dimensionality increases. In a sense, this becomes a philosophical problem rather than a practical one. What does it mean if two data points are very similar but actually not related because of some other unmeasured characteristic?

The first step in this clustering algorithm is to identify islands of d-adjacent non-empty bins. The test for d-adjacency is relatively simple to implement and the application involves at most $\frac{1}{2}\,N\,(N-1)$ tests of each non-empty bin against the other bins. The result of this test is P islands of bins. Each island represents a connected set of histogram bins. In two dimensions, you can envision the entire process like a checkers board with checker pieces stacked on squares to represent the occupancy count. Adjacent bins look like mountainous islands. Bins without adjacent bins look like tiny pedestal islands. Empty bins are the water between the islands. While it is harder to visualize higher dimensions, the intuition developed in two dimensions can readily be extended.

We could simply call each island a cluster and be done. However, there are situations in which there are end-member classes and a range of mixtures between the end-members. Using the checkerboard metaphor, this might look like a long thin island with peaks on each end (2 end-members) or a triangular island with three peaks at the corners (3 end-members) and so on. It is beneficial to be able to subdivide the islands to identify the end members.

Continuing with the 2-D island metaphor, the approach taken is to look for high mountains with valleys in between. One way to identify high mountains is to imagine flooding the islands and slowly lowering the water level. As the water level drops, mountain peaks will appear. Two peaks are divided into distinct clusters (sub-islands) if there is a deep enough valley between the two peaks. There are many ways to define the notion of "deep enough". One mechanism that is both robust and scales well with the number of data items is based on the observation that the filling of the histogram bins is a homogeneous Poisson process. This suggests that each bin will fill with an average rate, λ, and that the time between events will have a probability density function (PDF) equal to $\lambda \exp(-\lambda\, t)$. At a time t, the average population of the bin will be $N = \lambda t$ with a variance of $\sqrt{N} = \sqrt{\lambda\, t}$. Thus $\sqrt{N}$ represents a realistic scale for comparing a bin with adjacent bins.

To determine whether two peaks should be joined by a connecting bin, compare the height of the connecting bin to the lesser of the two peak heights (see Figure 1-B). If the height of the connecting bin is greater than $N - S\sqrt{N}$, then join the two peaks with the connecting bin. The choice of S is somewhat arbitrary but in practice $S = 1$ or 2 seems to produce consistently reasonable results. Larger values of S produce fewer clusters.

If the bin does not connect the two peaks (it is a valley), then there is a decision to make. To which cluster should the connecting bin be assigned? This algorithms implements a very simple test based on the height of each cluster and the distance (in index space) between the cluster peak and the connecting bin. The algorithm considers the slope from the connecting bin to the cluster peak and assigns the connecting bin to the cluster with the largest slope. This simple algorithm favors assigning the bin to taller clusters and closer peaks.

The Diluvian Clustering algorithm can be implemented efficiently by collecting all the individual data points in the multi-dimensional histogram. The non-empty bins are then sorted by population. This sorted set is then traversed from highest population to lowest population and clusters are constructed bin-by-bin. When a bin is adjacent to more than one cluster, the clusters are tested pairwise to determine whether the pair should be merged based on the depth of the valley between them. After one pass through the sorted bin set, all the bins are assigned to clusters based on adjacency. The computational efficiency of this process depends not upon the number of data items but on the number of filled bins which is typically much smaller.

Data

The Diluvian Clustering algorithm is routinely used in our laboratory for particulate samples. We have two scanning electron microscopes that are configured for automated particle analysis. In addition to the TESCAN mentioned earlier, we also have an ASPEX (since acquired by FEI) Personal SEM with Automated Particle Analysis software. The Personal SEM has a single 30 mm^2 silicon drift detector and is typically operated at 25 keV and 1 nA probe current producing a total output count rate on bulk copper of about 20 kcps.

The first data set was collected from a sample of Irish Sea sediment, a candidate for a future Standard Reference Material. The sediment powder was dispersed on a carbon planchet and then analyzed using an ASPEX Personal SEM. The instrument was configured to measure particles from 0.3 μm to 160 μm in average diameter. A total of 2 682 image fields were searched and a total of 23 787 particles were sized and a 2 second spectrum was collected from each. The electron beam was rastered over the particle in a series of chords through the particle center while the spectrum was collected. 783 particle spectra were discarded for having too few counts (<2 000 counts attributed to characteristic peaks). Typically, a spectrum with few counts is symptomatic of a particle consisting of mostly hydrocarbon. Marine sediment typically contains a very rich variety of particle compositions. Particles in samples like this may agglomerate and so there are often spectra representing multiple adjacent particles.

The elemental data excluding carbon and oxygen (28 elements) was binned with 11 bins per element each bin representing [0, 5 %), [5, 15%), [15%, 25%), …,[85 %, 95 %), and [95 %, 100%). The first and last bin were narrower (5 % wide) than the others (10 % wide) because 0% of something and 100% of something are meaningful special cases. This special importance is emphasized in the clustered data by making these bins narrower. Carbon was excluded because the substrate was made of carbon and it is not possible to distinguish between carbon in the particle and carbon in the substrate. Oxygen was excluded because the variance in oxygen is high because of mass and geometry effects. Excluding oxygen usually improves clustering performance. The elemental data excluding carbon and oxygen was renormalized to 100%. A peak-to-valley threshold of $S = 2$ was used to split clusters. The 23 004 particles populated 3 155 bins for an average bin population of 7.29 particles per bin. Only one bin in $4.6 \cdot 10^{25}$ bins (3 155 out of 11^{28} bins) was actually populated. As shown in Table 1, the algorithm discerned 4 clusters with over 1 000 particles, 28 clusters with over 100 particles, 50 clusters with over 10 particles and 111 clusters with over 2 particles. An additional 422 particles were identified as unique. 25.2 % of the particles were clustered in the most populous cluster and 79.5 % of the particles with sufficient counts were clustered in the 10 most populous clusters.

The most populous cluster, DV3[Fe=75.1±8.9,Si=9.6±4.5,Al=3.4±2.0,K=1.2±1.0], consisted of 5 988 particles in 324 distinct bins. The peak bin in this cluster contained 1 850 particles and the average bin contained 18.5 particles. The second most populous cluster, DV3[Ti=71.9±11.6,Si=9.2±7.8,Ba=3.8±2.7,La=3.2±1.9], consisted of 4 188 particles in 325 bins and its most populated bin contained 528 particles. These two clusters covered a relatively large number of bins. In contrast, cluster 4, DV3[Fe=90.2±2.9,Si=3.2±1.1,Co=1.2±0.7,Al=1.1±0.6], consisted on 1 580 particles in 9 bins with the peak bin containing 1 385 particles. These characteristics of the clusters are evident in the eight ternary diagrams in Figure 2. These ternary diagrams show the three most prevalent elements in each cluster. Clusters 1, 2 and 3 clearly represent distinct classes of particles but the clusters are diffuse. Cluster 4 represents a more precisely define (more homogeneous) type of particle, iron oxide particles. This demonstrates a strength of the Diluvian Clustering algorithm – the data itself define the extent of the clusters.

A number of clusters contained primarily aluminum, silicon and iron. Figure 3 shows these clusters. Ternary A shows a handful of different clusters ranging from pure silicon to pure iron in an assortment of colors. To eye, there does seem to be some logic to the assignment of particles to clusters although there are clusters which are not obviously distinct. There are clusters which in the ternary presentation seem to overlap. It should be remembered that we are only presenting three elements out of twenty-eight in the ternary and the distinction between the clusters can result from elements which are not being displayed. Ternary B shows the same clusters all colored red. This is to make the point that this clustering algorithm is making far deeper use of the data than we can by cursory visual examination of the ternary image. Inspection only segregates the data into two distinct clusters (silicon-rich and iron-rich) and a trail of particles in between. Ternary C, which display two of the same elements, shows what happens if we set the valley depth threshold (S) very high and essentially disable the portion of the algorithm that splits the islands into sub-clusters. Before splitting, this island contains 21 845 of the total 23 004 particles. This stresses the importance of splitting the island into sub-clusters. The high connectivity of the particles in this data set is typical of many complex environmental samples. There are many distinct types of particles in the data set. However, preparing an ideal sample for analysis is an unsolved problem. Particles agglomerate either fundamentally (as inclusions for example) or as a result of the sample preparation process. In large data sets, these agglomerates which quantify between the end-member constituents can form a trail of connecting bins like a sand-bridge between two islands. Disconnecting the islands, as the algorithm does, makes sense from this perspective.

In addition to the dominant clusters, there were a handful of interesting minor clusters. One of particular interest was the lanthanum-cerium-neodymium phosphate cluster, DV3[Ce=33.4±5.0,Nd=16.7±4.7,La=14.3±4.1,P=11.0±3.4], shown in the Figure 4. The ternary shows a distinct albeit diffuse cluster consisting of various admixtures of lanthanum, cerium and neodymium. The spectrum on the right is the sum spectrum over the 143 particles in the cluster. This cluster was not anticipated but the algorithm identified it. The cluster represents a known class of minerals called monazites. The ability to discern clusters we don't anticipate is one of the strengths of unassisted clustering. This algorithm's ability to identify them has proven invaluable on many internal and external samples.

The second example data set is contrived. Seven engineered glasses were selected (see Table 2) and ground into particle samples. Each glass was dispersed on a mount consisting of double-sided carbon tape on a 10 mm diameter pin mount. The samples were analyzed separately and each data set was quantified. In addition to the glass particles, there was a small fraction of contamination particles. The seven individual glass data sets were then combined into a single data set. The provenance of each particle in the combined data set was retained. The combined data set was clustered using the Diluvian Clustering algorithm and the clusters were compared to the source data.

As shown in Table 3 and Figure 5, the algorithm was very effective at extracting the known clusters. In all cases, the Diluvian Clustering algorithm assigned each distinct of glass to a cluster of its own. Very few particles were not assigned to the correct cluster as demonstrated by the low false negative rates. Very few particles were assigned to the wrong cluster as demonstrated by the low false positive rates. The most challenging glasses were the K2496 and K2469 which differ primarily in the amount of Ti present. The algorithm was highly effective at differentiating even these challenging glasses.

Discussion

The Advantages of the Diluvian Clustering Algorithm

Clustering high dimensionality data is difficult. Zimbek [Zimbek, 2014] identifies five different aspects that become increasingly challenging as the number of dimensions increase. These challenges are often collectively called the "curse of dimensionality." Grid-based algorithms handle some of these aspects better than other classes of algorithms. Zimbek's aspect 1 concerns the difficulty of optimization which increases exponentially as the number of dimensions increases. Grid based algorithms scale with the number of non-empty bins which is likely to increase with the number of independent dimensions. Zimbek's aspect 2 concerns the utility of Lp-norms to express differences diminishing as the dimensionality increases. Grid based algorithms don't depend on Lp-norms. Zimbek's aspect 3 concerns the introduction of irrelevant attributes. If one dimension contains random values then this attribute alone can disperse data points among distinct bins. Grid based algorithms can handle this gracefully so long as the number of bins is sufficiently smaller than the number of data items in a cluster defined by eliminating the irrelevant attribute. Metric-based algorithms don't handle irrelevant attributes well. Zimbek's aspect 4 concerns correlated attributes. If two or more dimensions are correlated then the intrinsic dimensionality of the data is smaller but the number of filled bins should not be significantly different. The number of filled bins is determined by the intrinsic dimensionality rather than the actual dimensionality. Grid algorithms handle correlated attributes easily. Zimbek's aspect 5 concerns the relative volume of a hypersphere in high dimensions. In a grid-based algorithm, this plays out in the definition of adjacency. If the algorithm requires 1-adjacency, then for each bin there are $2 \cdot D$ adjacent bins. One the other hand for D-adjacency, there are $3^D - 1$ adjacent bins. In my experience, the biggest practical difference is between 1-adjacency and 2-adjacency with 1-adjacency being optimal. Each added dimension serves to permit inherent noise in the data to adversely influence the clustering process. Figure 6 demonstrates how the algorithm's effectiveness diminishes for adjacencies larger than 1.

Other advantages of the Diluvian Clustering algorithm are:

1) Insensitivity to noise – When the bin width is chosen to be similar or greater than the natural inherent precision of the data then similar data points are highly likely to reside in the same or adjacent bins. The bin width should be smaller than the inherent structure of similar cluster's variance.
2) Insensitivity to dispersion within clusters – A cluster may be tight or diffuse. So long as the cluster contains at least a few members then each member is likely to be in the same bin or an adjacent bin so long as the bin width is selected according to 1).
3) Insensitivity to number of cluster members – This algorithm is equally capable of creating clusters with one member as 1 000. If a datum is unique then it will be given its own cluster.
4) Moderate number of tunable parameters – There are two tunable parameters, the binning and the valley depth. There is a natural way to select the bin widths for each dimension based on the inherent precision of the data. Regardless, the algorithm is relatively insensitive to the bin widths. Choosing a small S makes it more likely that clusters will be divided leading to more clusters.
5) Insensitivity to the shape of the cluster – There are no inherent assumptions (except those imposed by the bin shape) about the shape of the cluster such as are imposed by a distance metric.
6) Insensitivity to the ordering of input data – The algorithm is completely independent of the ordering of the input data.
7) Moderate memory requirements – The maximum memory requirement is determined not by the data size but by the number of filled bins.
8) Speed – The algorithm is extremely fast. The algorithm scales linearly with the number of date points and as the square of the number of filled bins. $O(M) + O(N^2)$
9) The intuitive nature of the algorithm – The algorithm is simple and intuitive enough to be described and understood quickly. The intuitive nature of the algorithm makes understanding why it discerned certain clusters relatively easy.

Criticisms of Diluvian Clustering:

1. The binning of the data imposes a rigid structure on the data which may not reflect the true structure of the data. Wide bins can group distinct clusters into a single cluster. Narrow bins can artificially split clusters particularly those with smaller numbers of members.
2. Running this algorithm on distinct data sets will produce different clusters making direct inter-comparison difficult. This can be addressed by merging the data sets, clustering the merged data and then splitting the data to recover the cluster populations in the original data.

Conclusions

In the author's experience, Diluvian Clustering has proven to be a very fast effective way to cluster particle microanalytical data sets. It is also well suited to compositional and other bounded data sets. The clusters discerned have proven to be useful for identifying real significant patterns in the data. Many of these patterns might never have been discovered by manually examination of the data or would have required many hours of intense effort. Often the identified clusters can be assigned to known types of materials or minerals. In some cases, the relative populations in the dominant clusters can be used to characterize a sample. In other cases, the singleton and low count clusters can be used to identify rare or unusual particles which can serve as a fingerprint of a class of sample.

Acknowledgements

Thanks to Jeff Davis and Jaideep Bhargava for collecting some of the glass data. Thanks to Abigail Lindstrom for preparing samples and for collecting some of the glass data.

References

Newbury, D. E. (2005), *X-ray spectrometry and spectrum image mapping at output count rates above 100 kHz with a silicon drift detector on a scanning electron microscope*. Scanning, **27** 227–239. doi: 10.1002/sca.4950270503.

Vandecreme, A., Bajcsy, P., Ritchie, N. W. M., & Scott, J. H. (2014) *Interactive Analysis of Terabyte-sized SEM-EDS Hyperspectral Images*, Microsc Microanal **20-S3** 654-655 doi: 10.1017/S1431927614004991.

Mott, R.B., Waldman, C.G., Batcheler, R. & Friel, J.J. (1995). *Position-tagged spectrometry: A new approach for EDS spectrum imaging*. In proceedings Microscopy and Microanalysis 1995, Bailey, G.W., Ellisman, M.H., Hennigar, R.A. & Zaluzec, N.J. (Eds.), 592–593.

Kotula, P., Keenan, M. & Michael J. R. (2003) *Automated Analysis of SEM X-Ray Spectral Images: A Powerful New Microanalysis Tool* Microsc. Microanal. **9** 1–17

Dempster, A., Laird, N., and Rubin, D. (1977) *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society, Series B (Methodological), 39(1) p 1-38.

Wilson, N. C., MacRae, C. M. Torpy, A., Davidson, C. J., Vicenzi, E. P. (2012) *Hyperspectral Cathodluminescence Examination of Defects in a Carbonado Diamond*, Microsc Microanal **18** Number 6, 2012,

MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations*. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability p 281-297 University of California Press.

Cortes, C., Vapnik, V. (1995) *Support-vector networks*. Machine Learning, **20**, Issue 3, 273-297

Schikuta, E. (1996) *Grid-clustering: a fast hierarchical clustering method for very large data sets.* in Proceedings 15th Int. Conf. on Pattern Recognition, 101—105

Schamber, F. H. (1977) *A Modification of the Linear Least Squares Fitting Method Which Provides Continuum Suppression*. in X-Ray Fluorescence Analysis of Environmental Samples, Dzubay, T. (Ed).

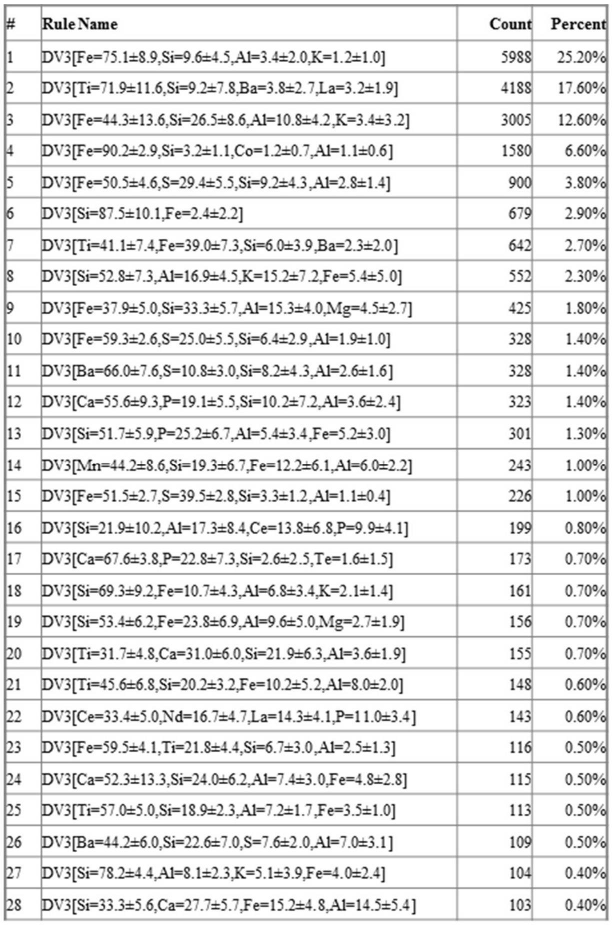
Goldstein, J.I., Newbury, D.E., Joy, D.C., Lyman, C.E., Echlin, P., Lifshin, E., Sawyer, L. & Michael, J.R. (2003) *Scanning Electron Microscopy and X-ray Microanalysis*. New York: Kluwer Academic/Plenum Publishers.

Bright, D. S. & Newbury, D. E. (2004) *Maximum pixel spectrum: a new tool for detecting and recovering rare, unanticipated features from spectrum image data cubes*. Journal of Microscopy, **216**, Issue 2, 186–193 DOI: 10.1111/j.0022-2720.2004.01412.x

Aggarwal, C. C. & Reddy C. K (2014)., *Data Clustering: Algorithms and Applications*. Boca Raton, CRC Press

Gan, G., Ma, C., & Wu J. (2007) *Data Clustering: Theory, Algorithms and Applications*. ASA-SIAM Series on Statistics and Applied Probability, Philadelphia, SIAM

Zimbek, A. (2014), *Clustering High-Dimension Data.* chapter in Data Clustering: Algorithms and Applications. Aggarwal, C., Reddy, C., (eds), Boca Raton: CRC Press



#	Rule Name	Count	Percent
1	DV3[Fe=75.1±8.9,Si=9.6±4.5,Al=3.4±2.0,K=1.2±1.0]	5988	25.20%
2	DV3[Ti=71.9±11.6,Si=9.2±7.8,Ba=3.8±2.7,La=3.2±1.9]	4188	17.60%
3	DV3[Fe=44.3±13.6,Si=26.5±8.6,Al=10.8±4.2,K=3.4±3.2]	3005	12.60%
4	DV3[Fe=90.2±2.9,Si=3.2±1.1,Co=1.2±0.7,Al=1.1±0.6]	1580	6.60%
5	DV3[Fe=50.5±4.6,S=29.4±5.5,Si=9.2±4.3,Al=2.8±1.4]	900	3.80%
6	DV3[Si=87.5±10.1,Fe=2.4±2.2]	679	2.90%
7	DV3[Ti=41.1±7.4,Fe=39.0±7.3,Si=6.0±3.9,Ba=2.3±2.0]	642	2.70%
8	DV3[Si=52.8±7.3,Al=16.9±4.5,K=15.2±7.2,Fe=5.4±5.0]	552	2.30%
9	DV3[Fe=37.9±5.0,Si=33.3±5.7,Al=15.3±4.0,Mg=4.5±2.7]	425	1.80%
10	DV3[Fe=59.3±2.6,S=25.0±5.5,Si=6.4±2.9,Al=1.9±1.0]	328	1.40%
11	DV3[Ba=66.0±7.6,S=10.8±3.0,Si=8.2±4.3,Al=2.6±1.6]	328	1.40%
12	DV3[Ca=55.6±9.3,P=19.1±5.5,Si=10.2±7.2,Al=3.6±2.4]	323	1.40%
13	DV3[Si=51.7±5.9,P=25.2±6.7,Al=5.4±3.4,Fe=5.2±3.0]	301	1.30%
14	DV3[Mn=44.2±8.6,Si=19.3±6.7,Fe=12.2±6.1,Al=6.0±2.2]	243	1.00%
15	DV3[Fe=51.5±2.7,S=39.5±2.8,Si=3.3±1.2,Al=1.1±0.4]	226	1.00%
16	DV3[Si=21.9±10.2,Al=17.3±8.4,Ce=13.8±6.8,P=9.9±4.1]	199	0.80%
17	DV3[Ca=67.6±3.8,P=22.8±7.3,Si=2.6±2.5,Te=1.6±1.5]	173	0.70%
18	DV3[Si=69.3±9.2,Fe=10.7±4.3,Al=6.8±3.4,K=2.1±1.4]	161	0.70%
19	DV3[Si=53.4±6.2,Fe=23.8±6.9,Al=9.6±5.0,Mg=2.7±1.9]	156	0.70%
20	DV3[Ti=31.7±4.8,Ca=31.0±6.0,Si=21.9±6.3,Al=3.6±1.9]	155	0.70%
21	DV3[Ti=45.6±6.8,Si=20.2±3.2,Fe=10.2±5.2,Al=8.0±2.0]	148	0.60%
22	DV3[Ce=33.4±5.0,Nd=16.7±4.7,La=14.3±4.1,P=11.0±3.4]	143	0.60%
23	DV3[Fe=59.5±4.1,Ti=21.8±4.4,Si=6.7±3.0,Al=2.5±1.3]	116	0.50%
24	DV3[Ca=52.3±13.3,Si=24.0±6.2,Al=7.4±3.0,Fe=4.8±2.8]	115	0.50%
25	DV3[Ti=57.0±5.0,Si=18.9±2.3,Al=7.2±1.7,Fe=3.5±1.0]	113	0.50%
26	DV3[Ba=44.2±6.0,Si=22.6±7.0,S=7.6±2.0,Al=7.0±3.1]	109	0.50%
27	DV3[Si=78.2±4.4,Al=8.1±2.3,K=5.1±3.9,Fe=4.0±2.4]	104	0.40%
28	DV3[Si=33.3±5.6,Ca=27.7±5.7,Fe=15.2±4.8,Al=14.5±5.4]	103	0.40%

Table 1: *The 28 clusters with more than 100 particle members discerned in the Irish Sea sediment sample. The clusters are assigned names based on up to four primary elements, the average quantity of each element and the standard deviation among the particle members.*

Element	*Z*	*K1404*	*K1718*	*K2450*	*K2155*	*K2307*	*K2496*	*K2469*
Oxygen	8	28.0%	43.1%	42.7%	47.0%	45.5%	32.3%	30.6%
Sodium	11		14.8%		8.9%			
Aluminum	13			15.9%		7.9%		
Silicon	14	15.9%	29.1%	14.0%	32.7%	25.7%	22.9%	16.8%
Calcium	20		3.6%	21.4%		10.7%		
Titanium	22			6.0%		3.0%	1.8%	9.6%
Manganese	25				11.4%	3.2%		
Iron	26		10.5%					
Zinc	30	12.9%	101.0%			4.0%		
Zirconium	40	7.4%						
Barium	56	35.8%					43.0%	43.0%

Table 2: *Nominal compositions of the engineered glasses.*

	K1404	K1718	K2450	K2155	K2307	K2496	K2469	False Negative	False Positive
DV3[Si=18.3±2.7,Na=10.3±1.7,Mn=7.9±3.2]		7		9962	30			0.08%	0.4%
DV3[Si=14.8±3.3,Ca=8.3±2.7,Al=3.7±0.8,Mn=2.7±0.9]			1	8	9938	1	2	0.28%	0.1%
DV3[Ca=38.7±4.1,Al=15.4±0.9,Si=13.8±0.8,Ti=8.4±1.0]			7406					0.00%	0.0%
DV3[Si=23.6±2.7,Fe=11.3±2.7,Na=10.7±1.5,Ca=4.8±1.1]		4967						0.14%	0.0%
DV3[Ba=34.1±4.8,Zn=14.5±2.4,Si=8.4±0.8,Zr=4.3±0.7]	2497							0.00%	0.0%
DV3[Ba=68.3±2.4,Si=30.5±2.1,Al=0.7±0.3]						880	3	0.79%	0.3%
DV3[Ba=68.0±2.9,Si=19.7±2.0,Ti=12.0±2.3]						8	481	0.62%	1.7%
Nominal count	2497	4974	7406	9970	9966	887	484		

Table 3: *The clustering results on the glass data set. The nominal count is the count of particles in the original data set. The table shows how the glasses were assigned to clusters. The last two columns report how many particles in the original data set were not clustered into the appropriate cluster (false negative) and the number of particles not in the original data set which were mistakenly assigned to this cluster (false positive.)*

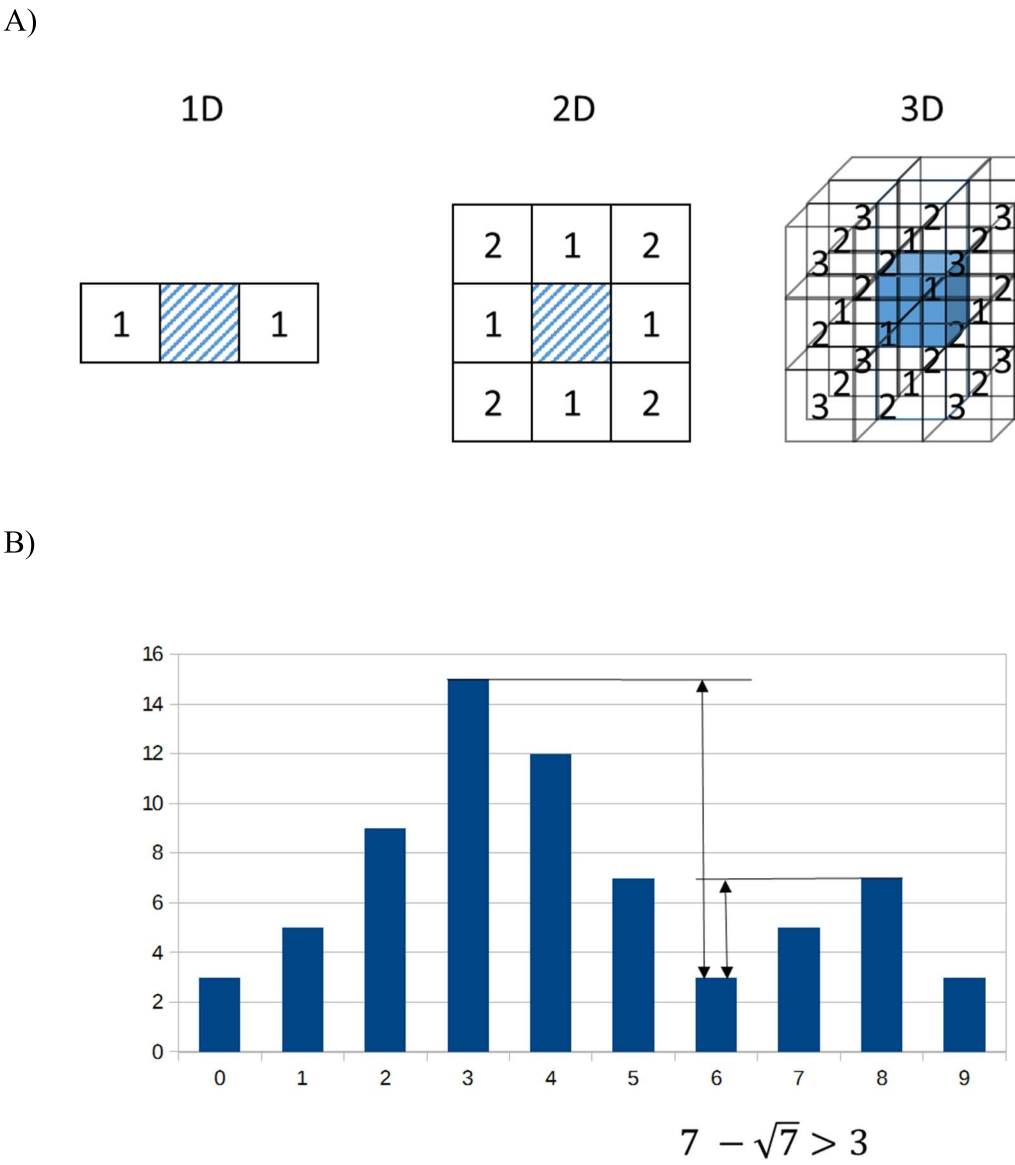
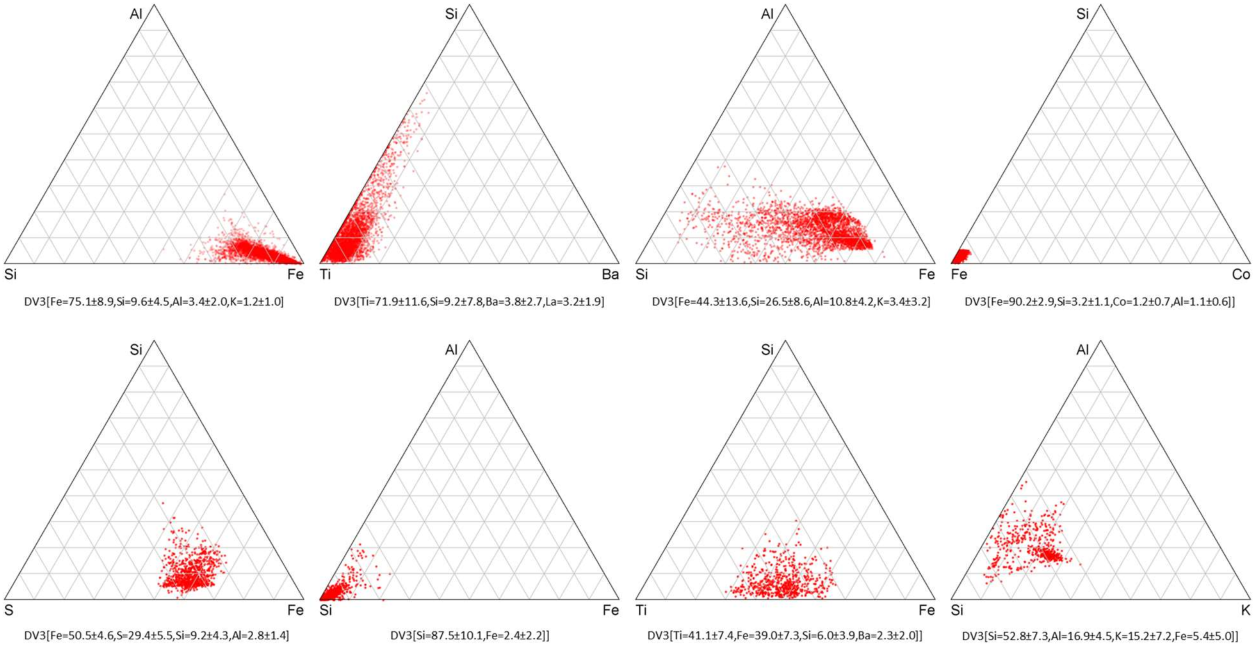


Figure 1: *A) These three figures demonstrate how adjacency is defined in 1-D, 2-D and 3-D. B) A contrived 1-D example of how a valley between two peaks is determined to be deep enough to permit splitting the two peaks into separate clusters. The test can be implemented in an equivalently manner in an arbitrary number of dimensions.*



DV3[Fe=75.1±8.9,Si=9.6±4.5,Al=3.4±2.0,K=1.2±1.0]

DV3[Ti=71.9±11.6,Si=9.2±7.8,Ba=3.8±2.7,La=3.2±1.9]

DV3[Fe=44.3±13.6,Si=26.5±8.6,Al=10.8±4.2,K=3.4±3.2]

DV3[Fe=90.2±2.9,Si=3.2±1.1,Co=1.2±0.7,Al=1.1±0.6]]

DV3[Fe=50.5±4.6,S=29.4±5.5,Si=9.2±4.3,Al=2.8±1.4]

DV3[Si=87.5±10.1,Fe=2.4±2.2]]

DV3[Ti=41.1±7.4,Fe=39.0±7.3,Si=6.0±3.9,Ba=2.3±2.0]]

DV3[Si=52.8±7.3,Al=16.9±4.5,K=15.2±7.2,Fe=5.4±5.0]]

Figure 2: *Ternary plots showing the primary three elements in each of the 8 most populous clusters in the Irish Sea sediment sample.*

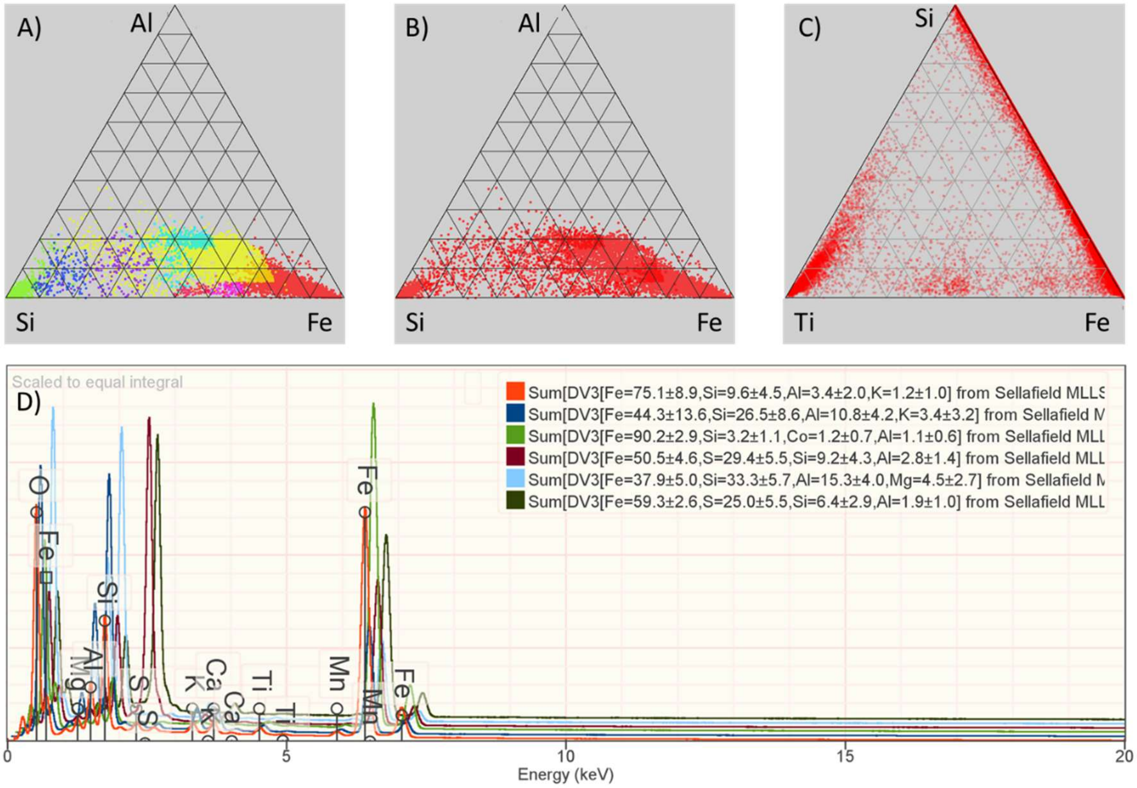


Figure 3: *Ternaries and sum spectra from the silicon, iron and aluminum rich clusters found in the Irish Sea sediment sample. A) A ternary in which each cluster has been colored distinctly. B) The same data points unclustered and all colored red. C) A ternary showing silicon, iron and titanium resulting from clustering the data but setting a very high threshold for splitting the islands into sub-clusters. D) The spectra associated with the six most prevalent clusters in A). The spectra are staggered to facilitated comparison.*

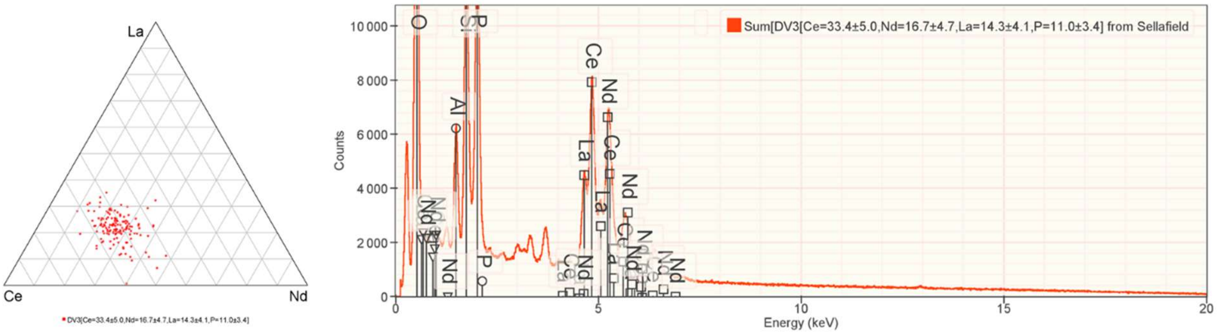


Figure 4: The ternary and the sum spectrum associated with an interesting un-anticipated cluster of particles, lanthanum-cerium-neodymium phosphate particles.

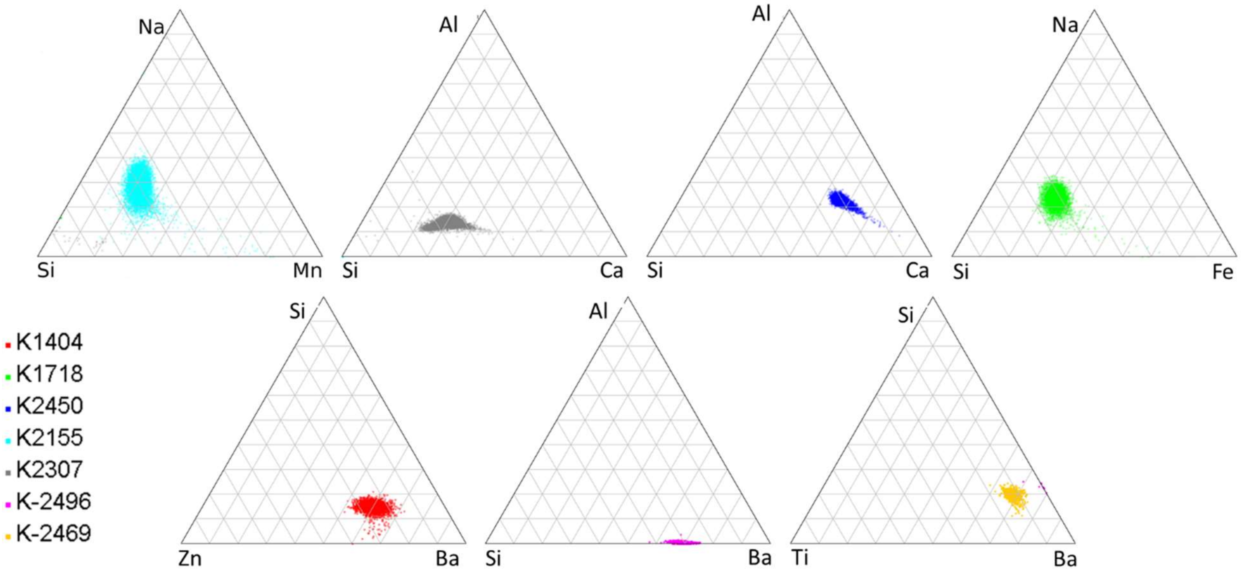


Figure 5: *To evaluate the effectiveness of the algorithm, a contrived data set was constructed from a collection of pure glass data sets. The contrived data set was clustered to determine whether the algorithm would correctly cluster the glasses according to their known composition. The largely consistent color of each ternary demonstrates that the algorithm clusters the particle data in a manner consistent with the known glass type.*

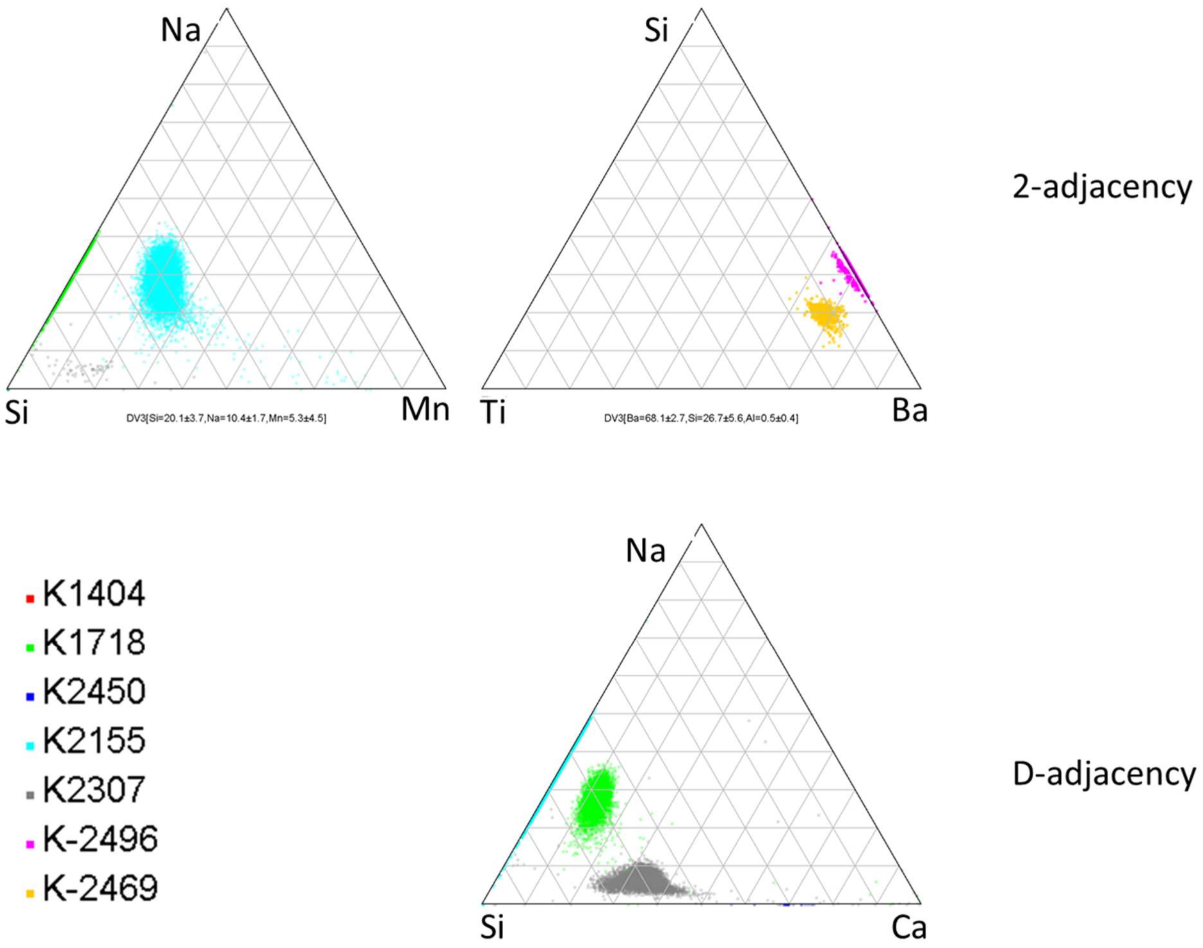


Figure 6: *The clusters in Figure 5 were determined using 1-adjacency. If the requirement is relaxed to 2-adjacency or even D-adjacency where D is the number of independent dimensions, then the performance of the algorithm to identify distinct clusters tends to be diminished.*

Appendix A: *A Java code fragment implementing the Diluvian Clustering algorithm.*

```
    final ArrayList<Cluster> result = new ArrayList<Cluster>();
    // Sort the bins tallest to shortest
    final SortedSet<Bin> bins = new TreeSet<Bin>(new Comparator<Bin>() {
       @Override
       public int compare(final Bin b0, final Bin b1) {
              final int res = -Integer.compare(b0.getHeight(), b1.getHeight());
              return res != 0 ? res : b0.compareTo(b1);
       }
    });
    bins.addAll(mBins);  // Data is collected in mBins
    // Join adjacent bins into islands
    for(final Bin bin : bins) {
       final ArrayList<Cluster> join = new ArrayList<Cluster>();
       // Determine which clusters the index is adjacent to
       for(final Cluster island : result)
              if(island.isNAdjacent(bin))
                     join.add(island);
       if(join.size() == 0)
              result.add(new Cluster(bin)); // Create a new cluster
       else {
              if(join.size() > 1) {
                     // Special case: bin intersects two Cluster objects
                     final int binHeight = bin.getHeight();
                     result.removeAll(join);
                     for(int i0 = join.size() - 2; i0 >= 0; --i0) {
                            final Cluster c0 = join.get(i0);
                            for(int i1 = join.size() - 1; i1 > i0; --i1) {
                                   final Cluster c1 = join.get(i1);
                                   final int minH = Math.min(c0.getHeight(),
c1.getHeight());
                                   if(binHeight > (minH - (mNSigma * Math.sqrt(minH))))
{
                                          // merge c1 into c0 and eliminate c1
                                          c0.mIndex.addAll(c1.mIndex);
                                          join.remove(c1);
                                   }
                            }
                     }
                     result.addAll(join);
              }
              { // Assign the Index to the cluster with the steepest climb
                     Cluster best = null;
                     double bestS = 0.0;
                     for(final Cluster c : join) {
                            final double cS = (c.getHeight() - bin.getHeight()) /

       bin.l2norm(c.getPeakPosition());
                            if((best == null) || (cS > bestS)) {
                                   best = c;
                                   bestS = cS;
                            }
                     }
                     best.mIndex.add(bin);
              }
       }
    }
    // Sort into max to min population order.
    Collections.sort(result, new Comparator<Cluster>() {
       @Override
       public int compare(final Cluster c1, final Cluster c2) {
              return -Integer.compare(c1.population(), c2.population());
```

```
   }
});
return Collections.unmodifiableList(result);
```