Data Clustering and Visualization in R

Jonathan Hundman

4/23/2020

Introduction

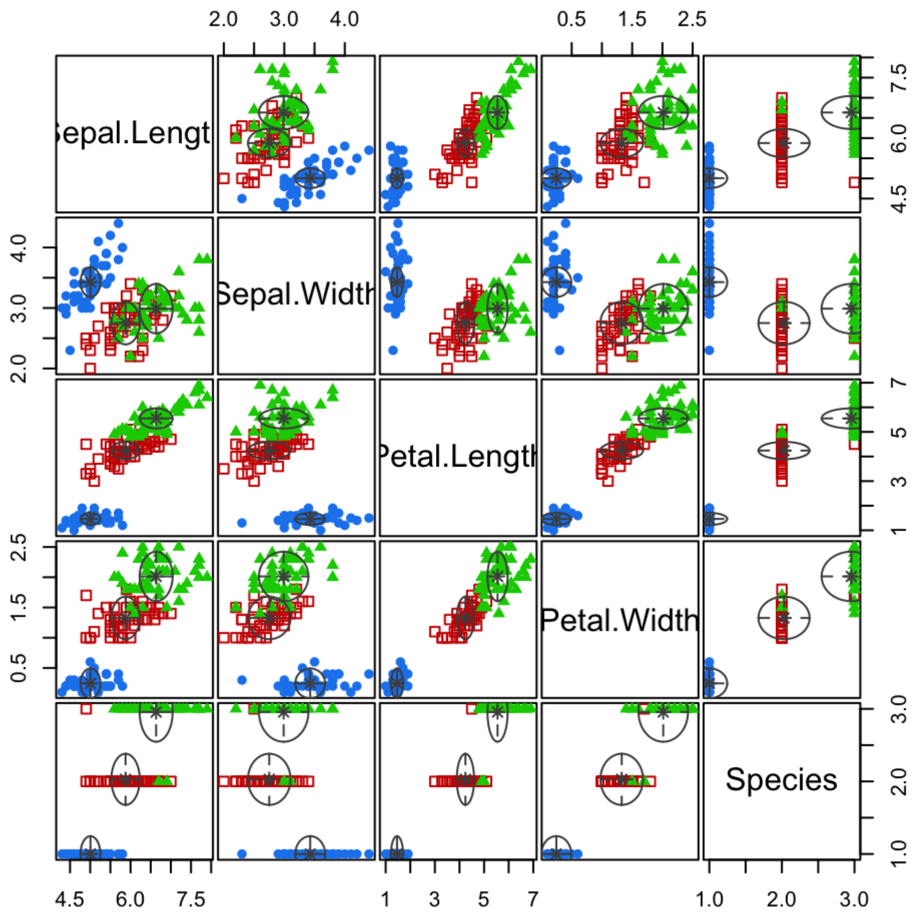
The goal of this analysis will be to explore some of the different clustering and clustering visualization libraries in R. With many different ways to approach the clustering problem, there are a lot of implementations throughout the R ecosystem. In addition, clustering can be difficult both to implement and display, especially with data that has higher dimensionality because of the difficulty to visualize it in more than two or even three dimensions. In order to explore these different implementations I will use the Iris dataset, as it has a long history of being used in statistical analysis. The iris dataset will also be useful because of its four dimensions. Then by the end of this analysis I will have provided an overview of some R libraries and some pros and cons of the different methods explored.

The iris dataset comes from the datasets library[1]. The data was originally collected by Edgar Anderson in 1936[2]. The data consists of three classes of iris flowers from the three species known as Setosa, Versicolor, and Virginica. For each of the three species four measurements were taken. The four measurements consist of the length and width for both the sepal and petals for each species. This data is useful because it is complex enough to where I can use libraries to display up to four dimensional clusters, and it is data which is relatively easy to cluster and see the differences between the three species. They aren't extremely distinct from each other but distinct enough where most clustering algorithms can perform fairly well.

Methods

Mclust and Built in Visualization

Before the exploration of new libraries I would first like to start at a baseline method learned in class. A clustering method learned in class was through the library mclust. The mclust library provides a function also named Mclust, which takes data, and clusters it into groups. You may also provide it arguments which change the way in which it clusters. Most commonly with the G argument you can control the number of clusters. This method works by estimating Gaussian mixture models through the use of the EM algorithm[3]. Below you can see what this clustering looks like.

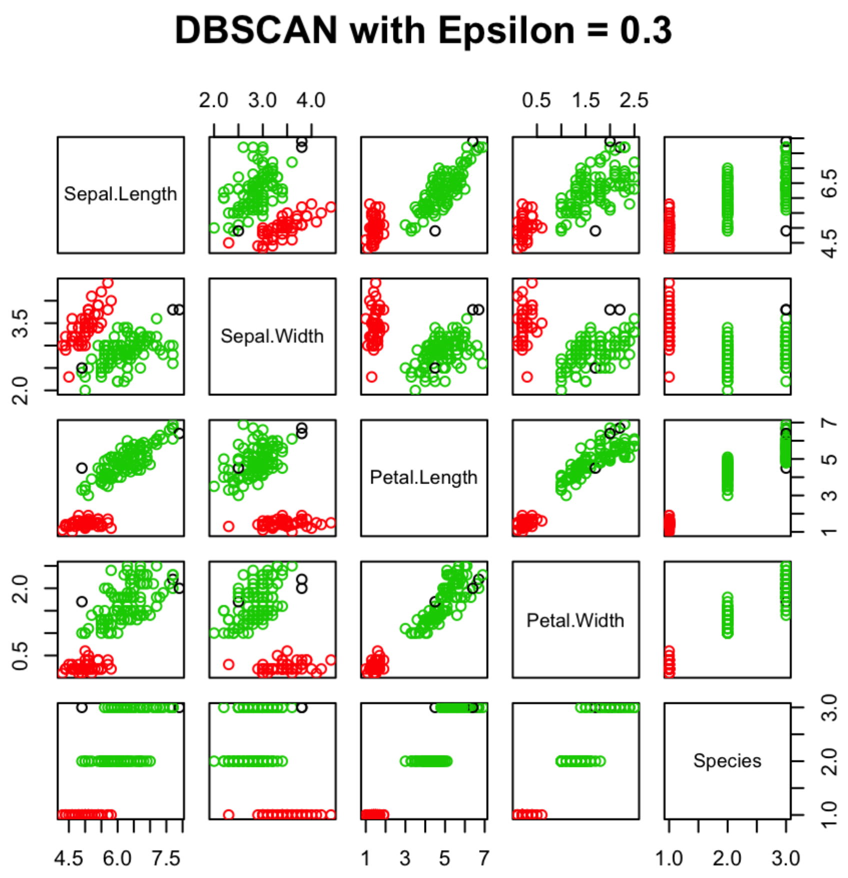


This plot is created from the built in plot function in R. As you can see from this implementation I've included the group label species. This gives you a general idea how a baseline algorithm may preform on this data.

DBSCAN

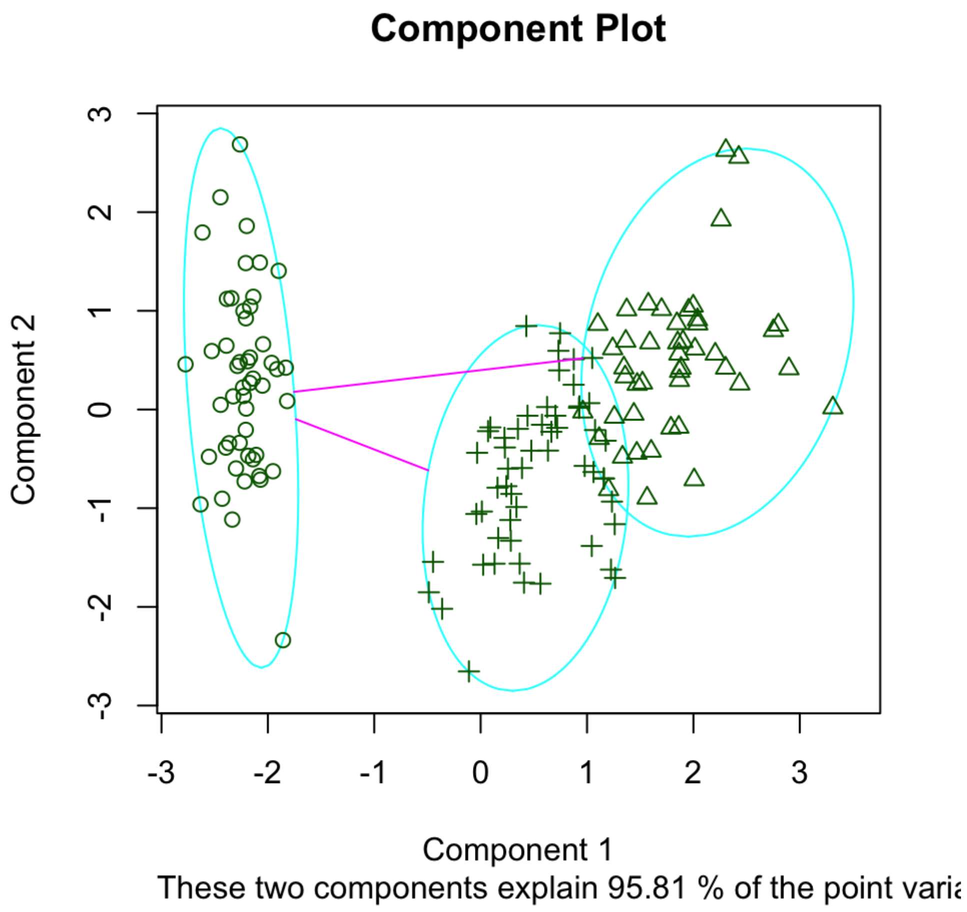
An interesting clustering algorithm I came across is the DBSCAN algorithm. It stands for density-based spatial clustering of applications with noise. It is from the dbscan library[4] and the function is named the same. This function is a density-based clustering algorithm. This function takes the data, and an epsilon value which determines the size of the neighborhood, and then calculates clusters accordingly. It was originally proposed in 1996 by Ester et al[5]. The original motivation for this algorithm was an algorithm which could create clusters of arbitrary shapes, run efficiently, and could provide information with little domain information from the user.

It accomplishes this in a process different than mclust. First the algorithm finds the neighborhood for each point as defined by the epsilon value. From there if finds all of the points which are connected to each other and assigns each group a center point. If the points are too far away from a group, it can either classify it as another cluster if the cluster contains enough points, or it will classify it as noise. This can be useful because the shape of the cluster can be an arbitrary shape. This could perform better than algorithms which find points which are the center of shapes if the data doesn't have a defined shape such as a circle.

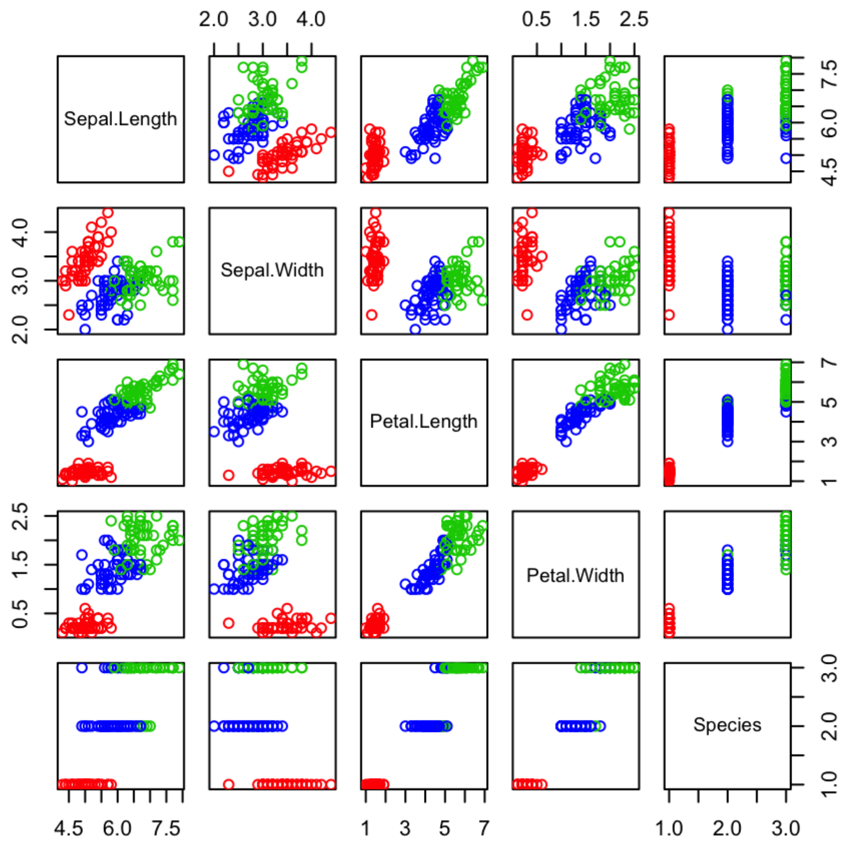


FANNY

This last clustering algorithm I explored was the FANNY algorithm, which stands for fuzzy analysis clustering. This function is from the cluster package is named fanny, which takes data and the number of clusters as inputs[6]. This method is a form of K-means clustering. It was originally created by Dunn in 1973[7]. This algorithm process works as follows. The first part of this algorithm is a number of clusters must be defined. After that start points with random cluster values and then calculate the center of each cluster. This process is then iterated until the best group of clusters is found. In addition when plotting this function the library includes a component plot. This is another way to visualize the clustering problem, by breaking it down into the two components which describe most of variance.



Component Plot
Component 2
Component 1
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
These two components explain 95.81 % of the point varia

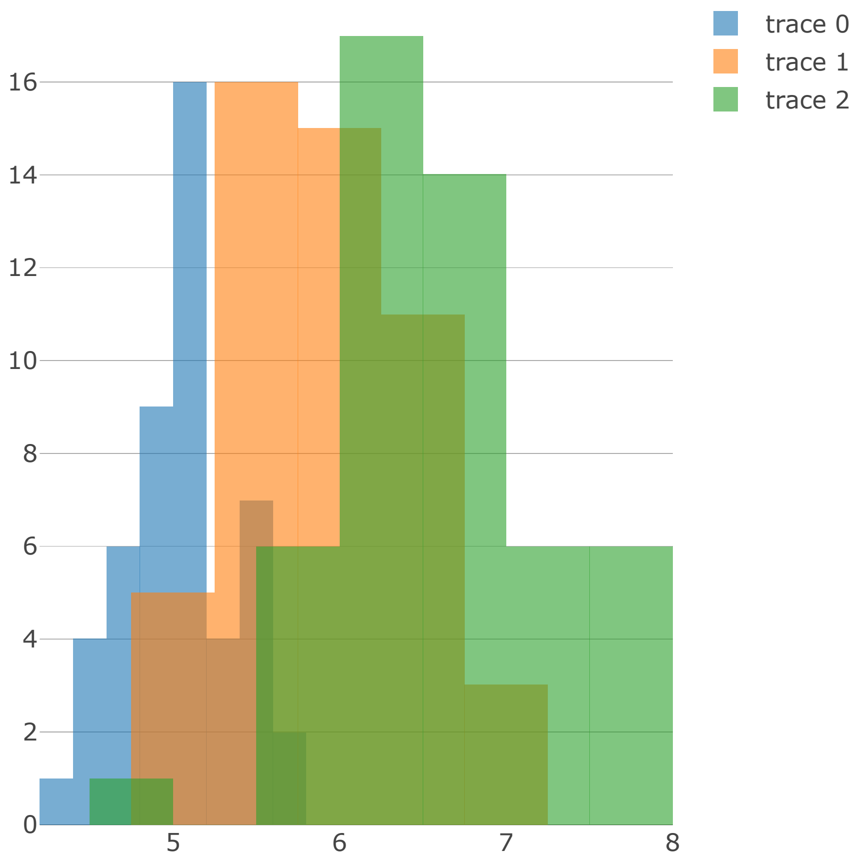


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
2.0
3.0
4.0
0.5
1.5
2.5
4.5
6.0
7.5
1
3
5
7
1.0
2.0
3.0

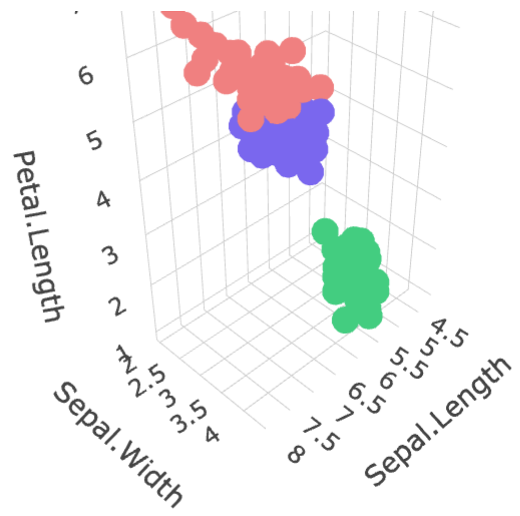
Plotly

Now to move on to some visualization techniques which may be useful for clustering. Some visualization has already been shown but the rest will center around the library of Plotly. The function plot_ly from the plotly library can enable two, three, and four dimensional plots. In addition when viewed as an html format, the graphs are interactive and allow for the manipulation of the view, and selection of variables. Plotly supports almost all general plot types, and is one of the best plotting libraries in R.

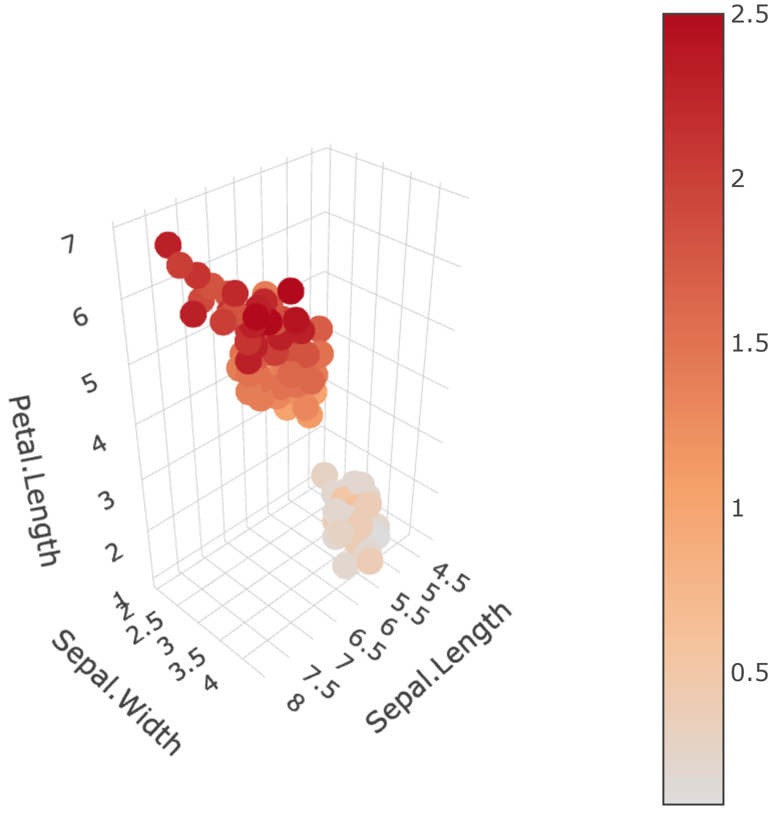
A plot in plotly which could be valuable for clustering, would be to compare cluster histograms for specific variables. As shown below plotly includes a function to create overlapping histograms.



The next two graph shows similar information but differ by three and four dimensions in a way. Technically they are both four dimensional but the first shows 3 dimensions of data and then the classification of the points, albeit the true classification. The second graph shows all four dimensions of the iris data with the fourth being petal width, which is a scale of color, while the other three dimensions are shown in 3d space. In addition the graphs allow for their manipulation of view which can be helpful in finding a specific point in a cluster which may be hidden in just a 2 dimensional view.



Petal.Length
Sepal.Width
Sepal.Length



Petal.Length
Sepal.Width
Sepal.Length
2.5
2
1.5
1
0.5

Analysis

Normally for an analysis you would try to compare models and their predictive power based on a test set. For clustering which is typically an unsupervised method I will instead focus on discussing the pros and cons of the methods shown, and for the visualizations I will also describe their use cases and where they might fall short.

Mclust and plot function.

A vast majority of observations get classified correctly with a few misclassified. It appears to do well for clusters which have a defined distribution (are generally ovular in shape). But it may not perform as well when given more abstract shapes which aren't all defined around a point. This could definitely change though depending on the number of clusters selected. In addition you can see the strengths and weaknesses of this plot. It does well by displaying all of the two way cluster interactions. For example you can see what clusters look like with petal length and sepal width. This gives a solid overview of all of the interactions. Where this falls apart is if you want to see three or four or n dimensional interactions as the built in R function just doesn't do that.

DBSCAN

When I came across the DBSCAN algorithm it stood out to me. While the Mclust and FANNY algorithms appear to work quite well with the Iris data when given the number of clusters DBSCAN doesn't do quite as well. I feel like this data highlights a possible weakness of DBSCAN which is a strength for different clustering problems. The neighborhood value appears to be quite useful for cases where more centroid or hierarchical methods may fail. It would probably perform better on data with a more abstract shape versus this data which appears to cluster in more circular/oval shapes. In addition it does seem to be quite fast, which given the large amount of data in this digital age would be useful in processing greater amounts of data.

FANNY

The FANNY algorithm appears to have performed quite well and similar to the Mclust algorithm. Even though they approach the problem in different ways they both appear quite similar in their results on the Iris dataset. As described before I think these two algorithms perform similarly on this kind of data relative to the DBSCAN algorithm. The other benefit of this function and library was they had specific plotting functions built into the library.

Plotly

This was the first time I have used plotly and I'm glad I did. What I found out about plotly is that the library can create just about any plot one might think of, and at the same time making it interactive. Unless an algorithm comes with a specific way to be viewed, then the default choice in my opinion should be plotly. Ggplot2 is also commonly used, but plotly seems to have more functionality. Specifically the three and four dimensional plots are quite useful for clustering.

Summary

In conclusion while the holy grail for data analysis would be one algorithm and visualization which works for all possible cases, that just isn't feasible. Clustering is a difficult task due to its typically unsupervised nature. Methods can be tested like I did on a supervised dataset like the Iris dataset, but when doing clustering without labels or knowledge of the number of groups the task becomes quite hard. That's why visualization and a variety of clustering techniques are needed. Throughout my analysis I found algorithms such as Mclust and FANNY which performed quite well, while DBSCAN did not. But all of these could do better or worse depending on how you tune the parameters. In order to tune properly visualization is needed which can give hints at how to select and then tune a certain algorithm. Plotly appears to be a solid choice for visualization in the R ecosystem. There isn't one solution to all data analysis problems, but R has a rich ecosystem for solutions to these problems, some of which I have shown off in this analysis.

Footnotes

1. datasets library (https://www.rdocumentation.org/packages/datasets/versions/3.6.2)↩
2. Iris Dataset (https://en.wikipedia.org/wiki/Iris_flower_data_set)↩
3. Mclust Function (https://www.rdocumentation.org/packages/mclust/versions/5.4.6/topics/Mclust)↩
4. DBSCAN Library (https://www.rdocumentation.org/packages/dbscan/versions/1.1-5/topics/dbscan)↩
5. DBSCAN Paper (https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf)↩
6. Fanny Function (https://www.rdocumentation.org/packages/cluster/versions/2.1.0/topics/fanny)↩
7. Dunn's Paper (https://www.tandfonline.com/doi/abs/10.1080/01969727308546046)↩