weightedClustSuite Instructions and Showcase

Dhanuj Gandikota

2/24/2021

Intro

Thanks for checking out my opensource package weightedClustSuite, developed by myself.

This package was created to easily implement weighted point density clustering and validation. In my package, one can easily give their datapoints weights and then see the results shown and validated with multiple unsupervised clustering algorithms. This is all done through a single clustering object making the data, ex. different clustering assignments, very easy to access.

The package operates through my implementation of manipulating the gaussian density estimates and distance matrices with the weights, prior to the bulk of each clustering algorithm. This performs more EFFICIENTLY and can allow for much LARGER weighted datasets than the brute force alternative of manually adding each points weight to the dataset.

```
#Run this command to install the package on your machine
#devtools::install_github("dhanujg/weightedClustSuite")
```

Current clustering algorithms currently implemented with weights in the package

Density Based Unsupervised Clustering

- Density Peak Clustering (DOI: 10.1126/science.1242072, 2014)
- Density-based spatial clustering of applications with noise (DOI: 10.5120/739-1038, 1996)

Partitioning Based Unsupervised Clustering

- K-Mediods Clustering (https://doi.org/10.1016/j.eswa.2008.01.039, 2008)
- Spectral Clustering (DOI: 10.5555/2980539.2980649, 2001)

Examples of Package Utilization

We will use the package on a dataset currently under research. This dataset represents a 2 species trait values and the species abundance for each species. The data is presented in 3 columns, 2 of which are the species trait values and the third are the WEIGHTS (species abundance)

Create input Data & Weight & Distance Matrices

```
# Read in the Data
Colnames = c('X','Y','Weight')
Species_Data <- read.csv(file.choose(), header = FALSE)

#Create Input Matrices for the package
Trait_Species <- as.vector(Species_Data[,1:2])
```

```
Weight_Species <- as.vector(Species_Data[,3])
Dist_Species <- dist(Species_Data[,1:2])
Trait_Species[1:5,]
```

```
##          V1         V2
## 1 0.4483389 0.06773519
## 2 0.3048936 0.08832187
## 3 0.6579542 0.24115284
## 4 0.3805193 0.21040881
## 5 0.2878266 0.39232825
```

```
Weight_Species[1:5]
```

```
## [1] 319  40 139  32  85
```

Call the package and create the overall Clustering Object

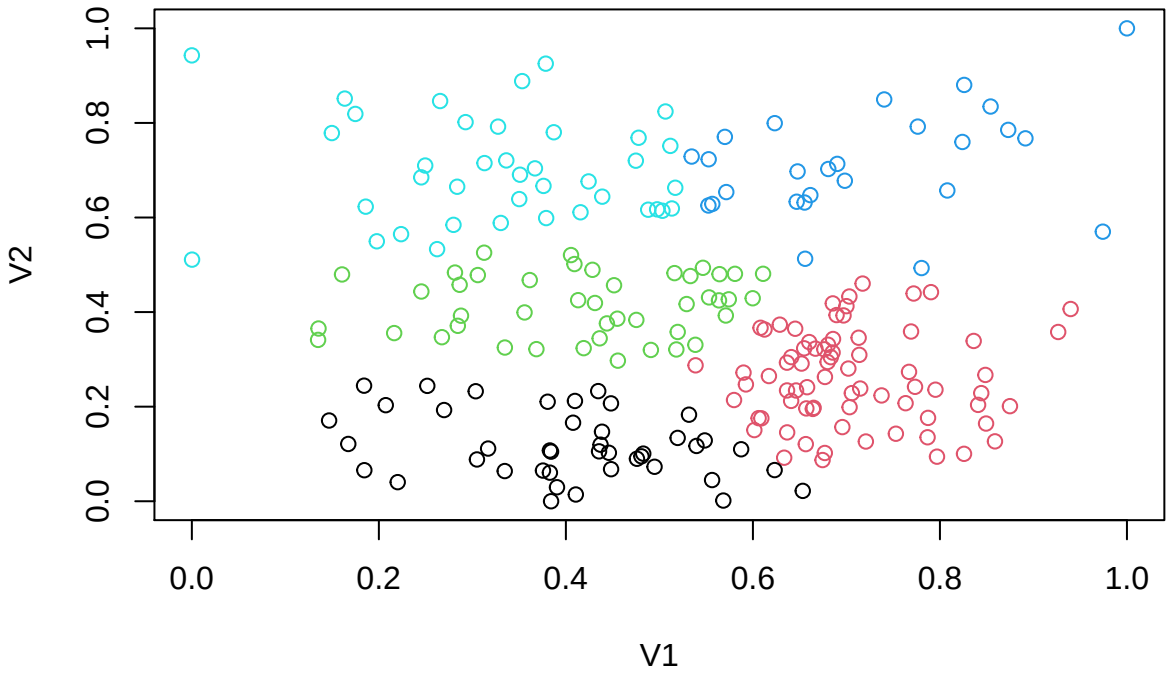
```
library(weightedClustSuite)

#feed in the input data to create the base clustering object with ClustObj function
SpeciesClust <- ClustObj(Trait_Species, Weight_Species, Dist_Species, gaussian=TRUE, verbose = TRUE)
```

```
## Calculating the weighted distance cutoff

## Distance cutoff calculated to 0.04115963

## Calculating the weighted local density for each sample based on distance cutoff

## Calculating the minimal distance of a weighted sample to another weighted sample with higher density

## Returning result...

## Warning in as.dist.default(new_dist): non-square matrix
```

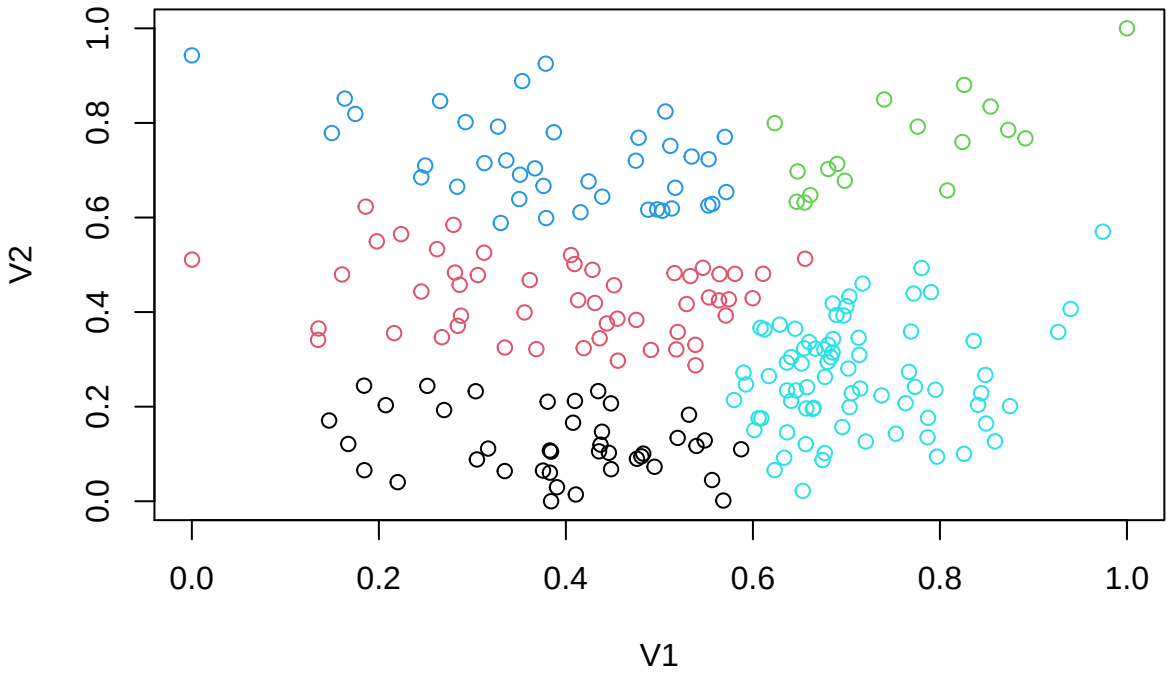
Quick Run of Clustering Algorithms

We can now update the clustering object (SpeciesClust) with data from a clustering algorithm, here we will run K mediods and Spectral to get a quick visualization of the data.

```
#SpeciesClust <- findDensityPeakClusters(SpeciesClust, rho=4.01, delta=0.21)
SpeciesClust <- findKMediodsClusters(SpeciesClust, neighbors = 5 )
```

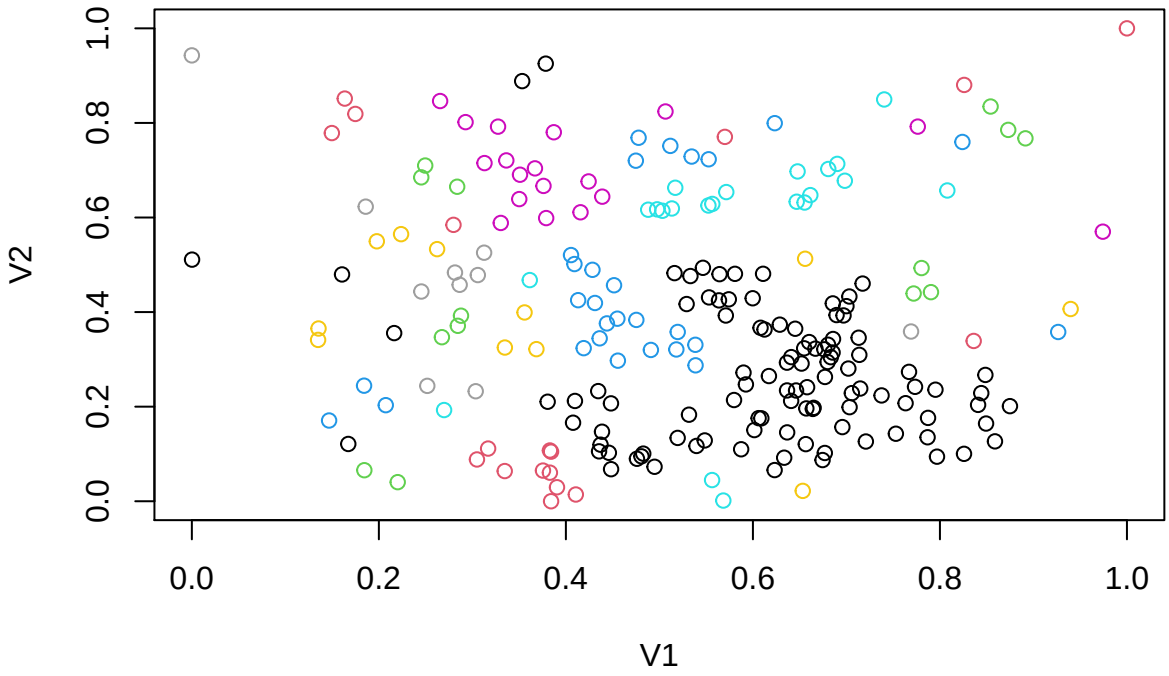



```
SpeciesClust <- findSpectralClusters(SpeciesClust, neighbors = 5 )
```



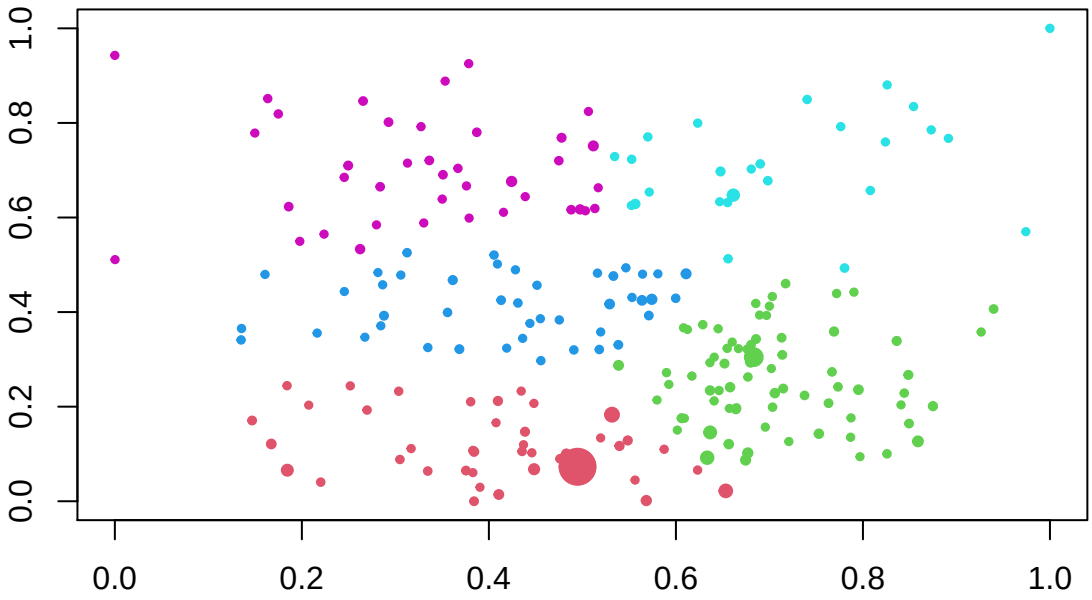

```
##           [,1]        [,2]        [,3]         [,4]        [,5]
## 1 -0.03482799 -0.11168252 -0.03700830  0.039882040 -0.06681531
## 2  0.09676751  0.01141596 -0.04053977  0.002355249 -0.06681531
## 3  0.03804633 -0.07600336  0.17006778 -0.119432347 -0.06681531
## 4 -0.05845349  0.05044484 -0.05329874 -0.099540353 -0.06681531
## 5 -0.02639332  0.04208394  0.03652498  0.055193015 -0.06681531
```

```
SpeciesClust <- findDBSCANClusters(SpeciesClust, eps = 0.05)
```

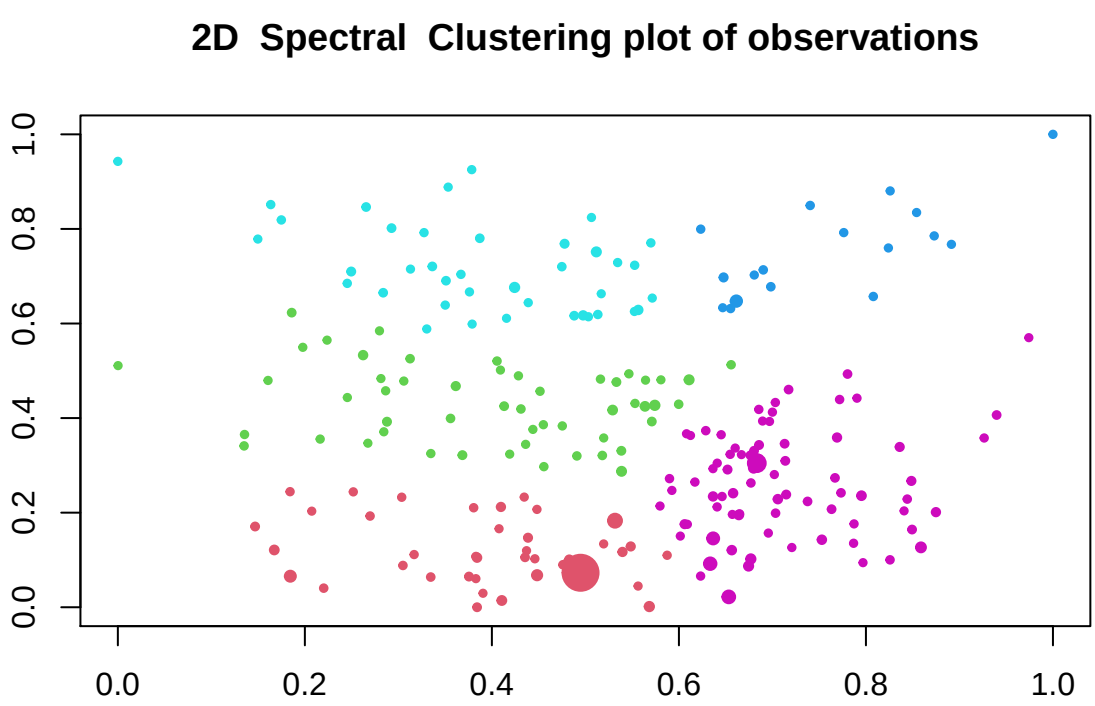


We can also create plots of each of the clustering assignments WITH the datapoint sizes changing corresponding to the weights

```
KMediods_2D <- plot2DCluster(SpeciesClust, SpeciesClust$clustersKmediods, type = 'Kmediods')
```




```
Spectral_2D <- plot2DCluster(SpeciesClust, SpeciesClust$clustersSpectral, type = 'Spectral')
```



clustering assignments are also stored in the class object and can be viewed in examples like

```
SpeciesClust$clustersKmediods[1:10]
```

```
##  [1] 1 1 2 1 3 1 2 3 4 2
```

Detailed Clustering and Validation : Example using 2014 Density Peak Clustering

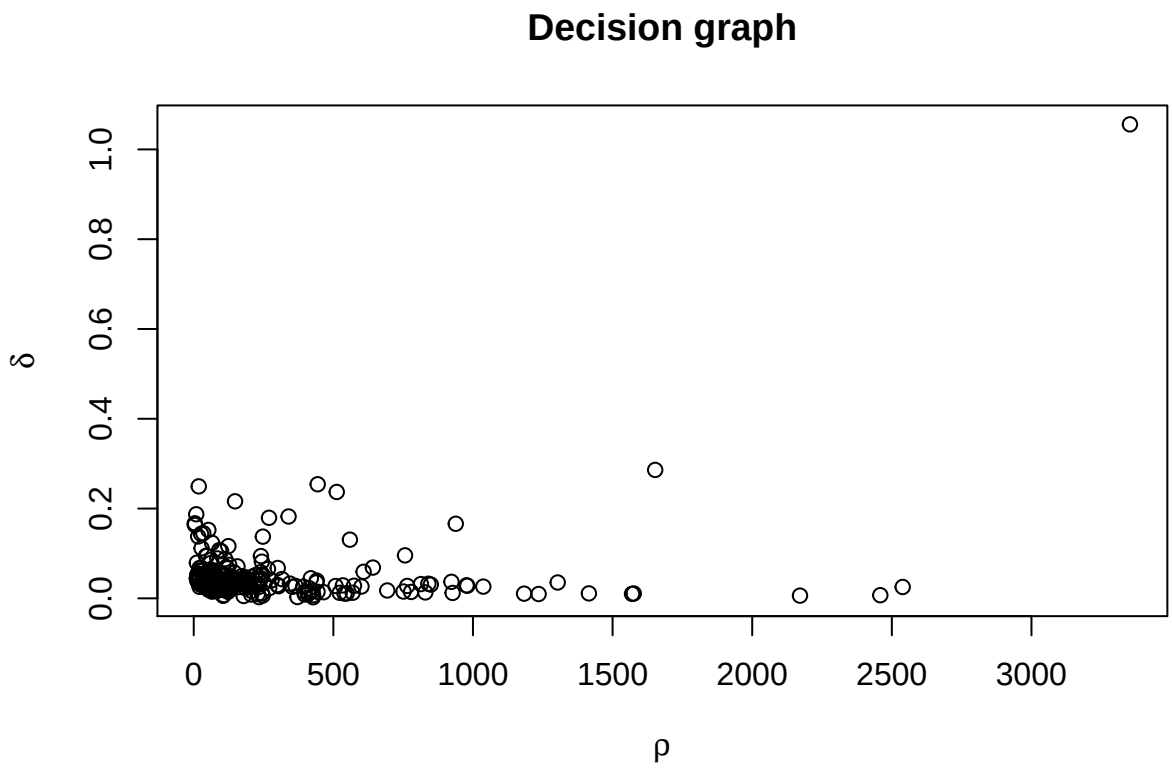
#Here I will go through the full training and implementation of performing Weighted Density Peak Cluster by Rodriguez et al. 2014.

Create a Validation chart to test various rho/delta combinations and obtain ideal classification score

```
SpeciesChart <- findDensityPeakCluster_validationChart(SpeciesClust, DBCV = FALSE, status = FALSE)
View(SpeciesChart[1:5,])
```

We can also visualize the density peaks by rho/delta values in this plot

```
plotDensityPeak(SpeciesClust)
```

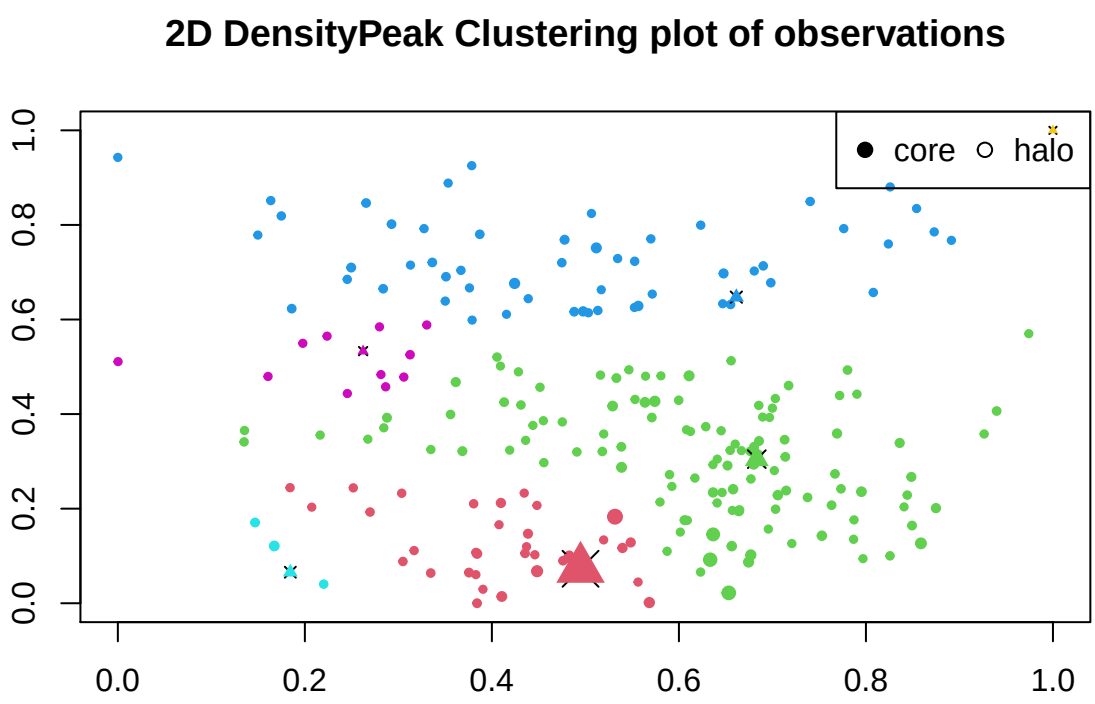


from the validation chart, we have chosen this specific rho/delta combination to give us 5 clusters with no unclassified points, we can then run the Density Peak clustering

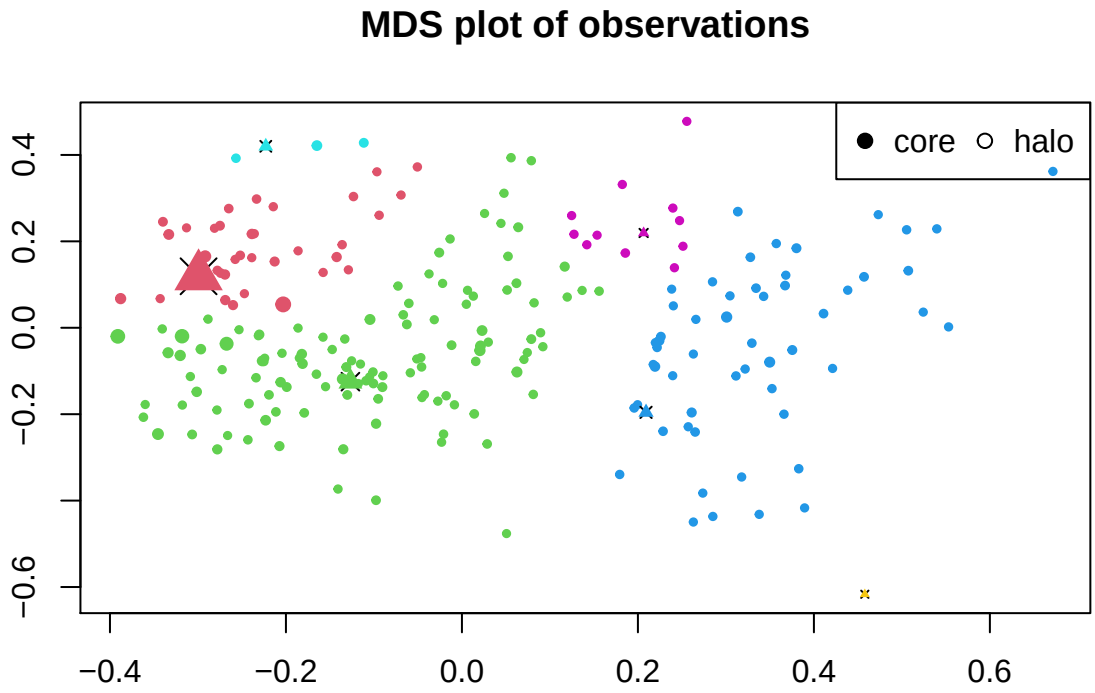
```
SpeciesClust <- findDensityPeakClusters(SpeciesClust, rho=4.01, delta=0.21, verbose = FALSE)
```

Now we can plot our clustering assignments through a 2D Graph, a MDS graph (for higher dimensions), and a TSNE Graph. The cross marks indicate the cluster centers and the size of the points correspond to the weights.

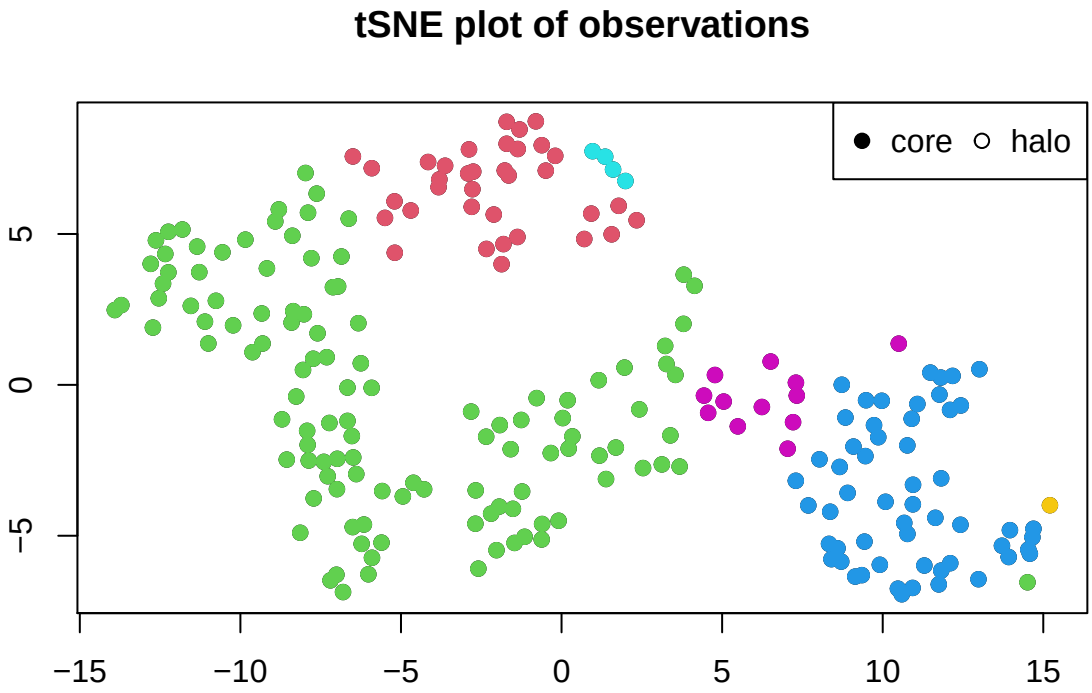
```
DensityPeak_2D = plotDensityPeak2D(SpeciesClust)
```




```
DensityPeak_MDS = plotMDS(SpeciesClust)
```



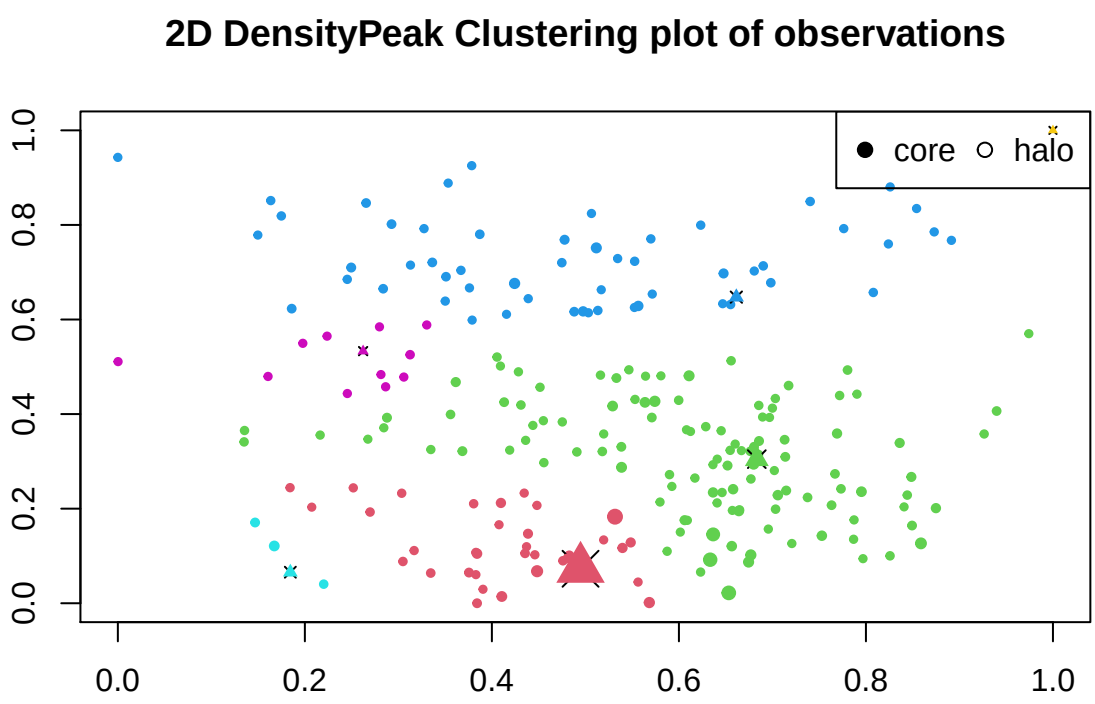

```
DensityPeak_TSNE = plotTSNE(SpeciesClust)
```



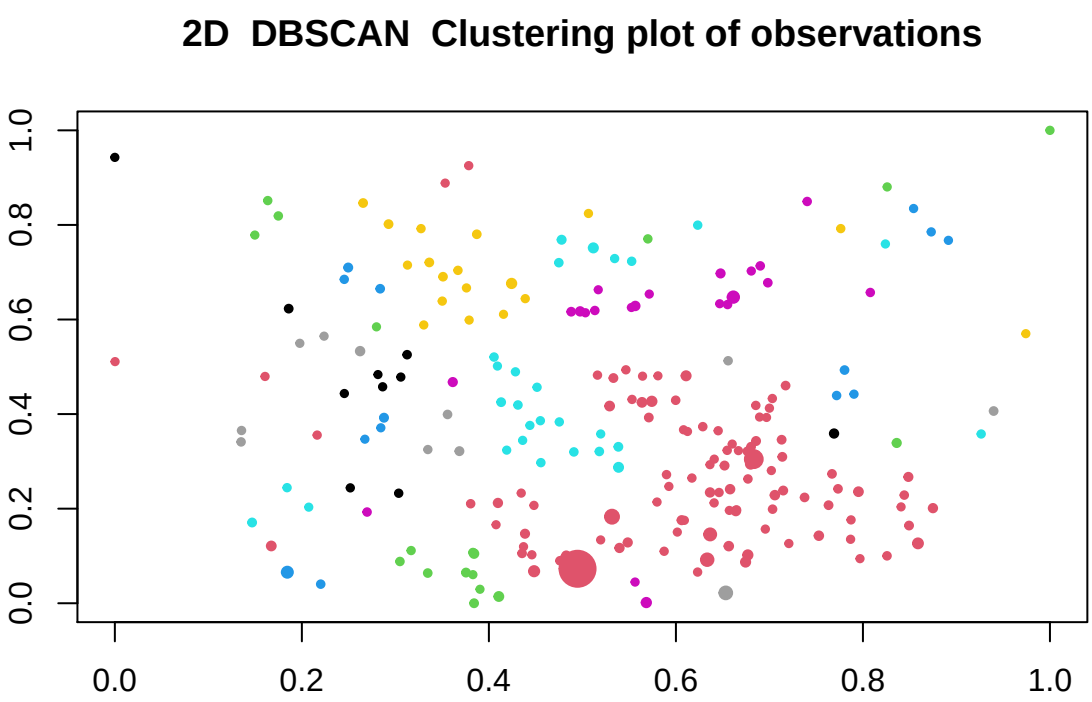
###Model Comparison

We can compare the results of our Density Peak Clustering, with DBSCAN clustering utilizing the same weighted distance cutoff for epsilon.

```
DensityPeak_2D = plotDensityPeak2D(SpeciesClust)
```




```
DBSCAN_2D = plot2DCluster(SpeciesClust, SpeciesClust$clustersDBSCAN, type = 'DBSCAN')
```



Future

There is a lot more to be added to the package and still currently being edited for implementation. Please email me at dhanujg@umich.edu if you have any questions.