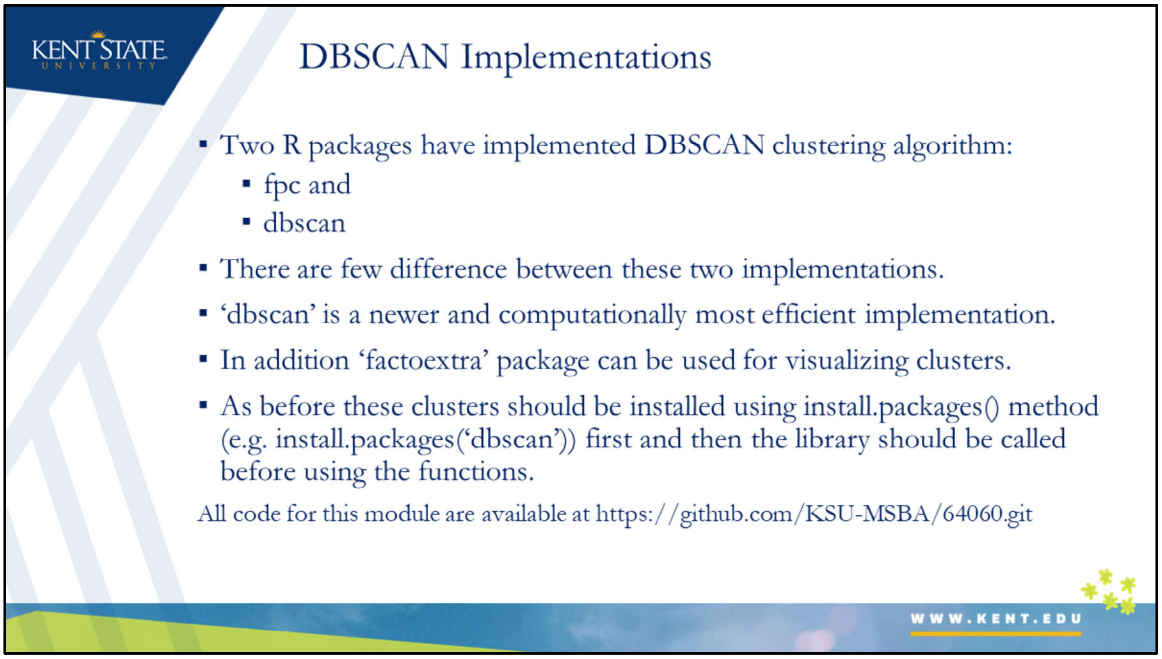
Here we discuss the DBSCAN Implementation in R.



R contains implementations of DBSCAN in the packages dbscan and fpc. Both packages support arbitrary distance functions via distance matrices. The package fpc does not have index support (and thus has quadratic runtime and memory complexity) and is rather slow due to the R interpreter. The package dbscan provides a fast C++ implementation using k-d trees (for Euclidean distance only) and also includes implementations of DBSCAN*, HDBSCAN*, OPTICS, OPTICSXi, and other related methods. For the most part, we will use dbscan. Remember to install the package before first use.

KENT STATE UNIVERSITY

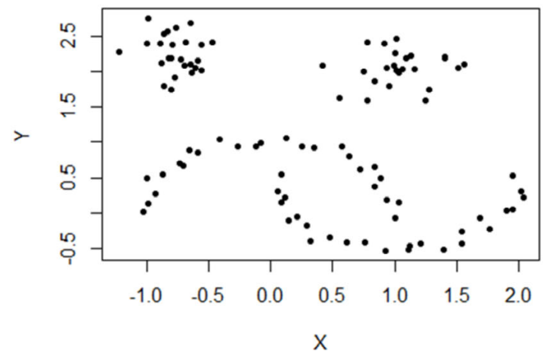
Example I: DBSCAN Package

```r
library("dbscan")

library('factoextra')

library('fpc')

#moons dataset Contains 100 2-D points, half of which are contained in two
# moons or "blobs" (25 points each blob), and the other half in asymmetric
# facing crescent shapes.
data("moons")
plot(moons, pch=20) # plot original data points
```

WWW.KENT.EDU

We now apply dbscan package to a sample dataset. Please see the github account for the code.

Example I: “moons” Dataset



WWW.KENT.EDU

This data contains nonlinearity as well differing densities.

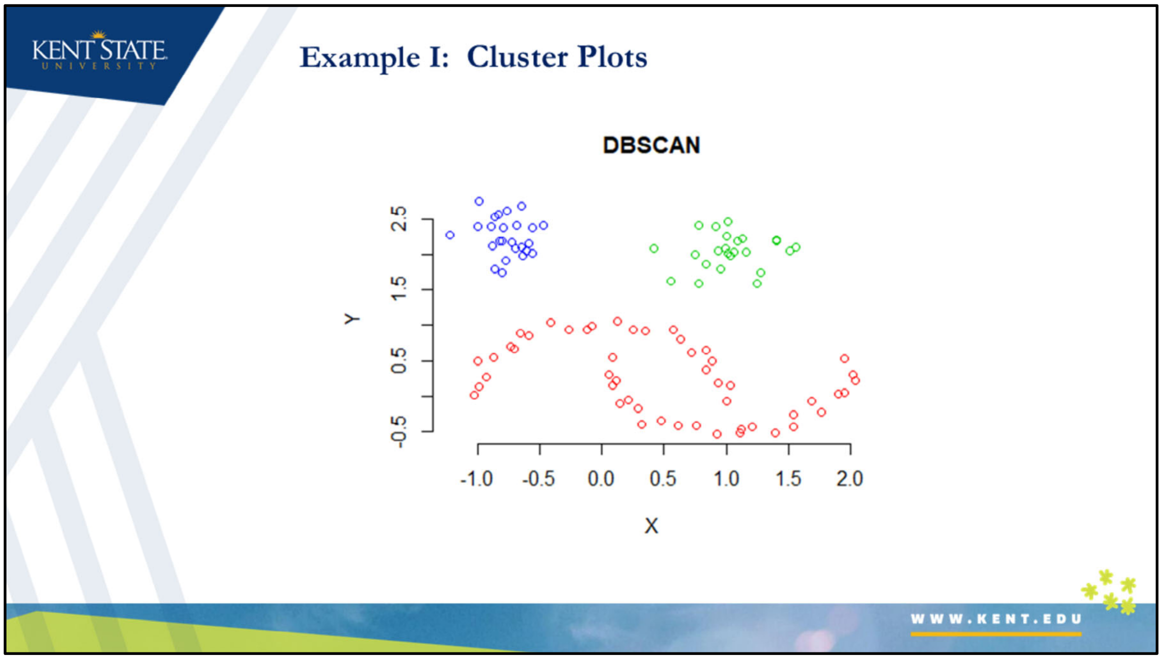
Example I: DBSCAN Implementation

```
db <- dbscan::dbscan(moons, eps = 0.5, minPts = 5) #perform clustering

print(db) #print cluster details

## DBSCAN clustering for 100 objects.
## Parameters: eps = 0.5, minPts = 5
## The clustering contains 3 cluster(s) and 0 noise points.
##
##  1  2  3
## 50 25 25
##
## Available fields: cluster, eps, minPts

plot(db, moons, main = "DBSCAN", frame = FALSE) #plot cluster details
```

Clusters' Details

WWW.KENT.EDU

The output from dbscan provides cluster details, which we can visualize using the plot command.



Clearly, dbscan has done well in identifying the clusters.

Example II: fpc Package

```
library("dbscan")
library('factoextra')
library('fpc')

df <- multishapes[, 1:2]

set.seed(123)
db <- fpc::dbscan(df, eps = 0.15, MinPts = 5) # DBSCAN using fpc package

print(db) # show clusters' details
```

Noise points

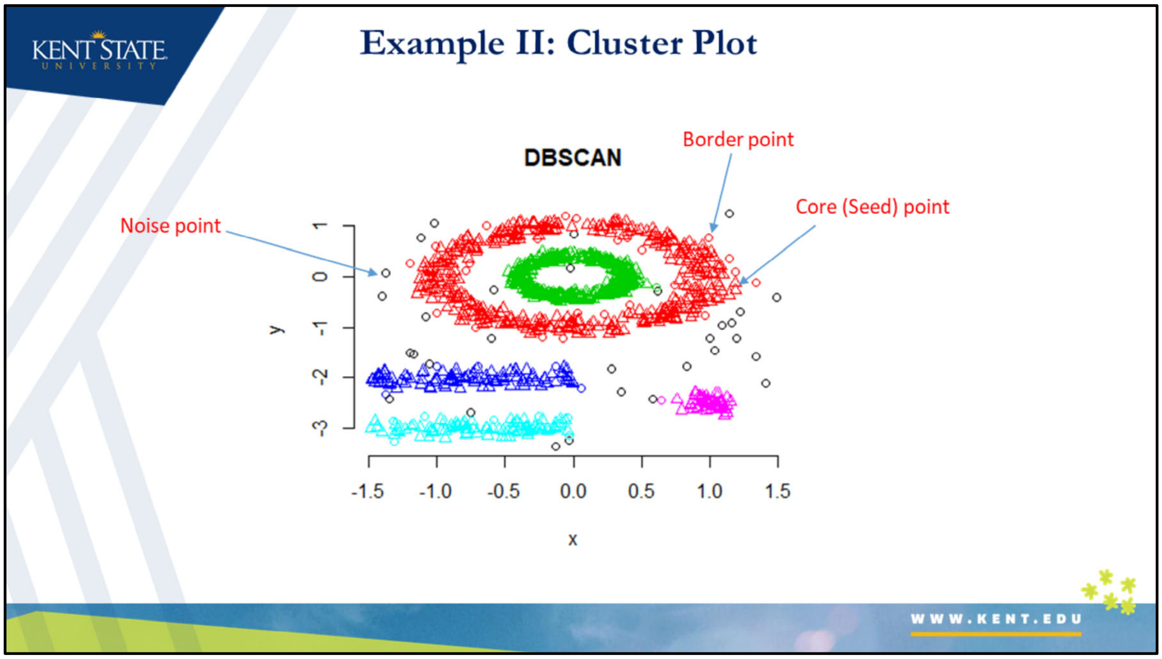
```
## dbscan Pts=1100 MinPts=5 eps=0.15
##         0   1   2   3  4  5
## border 31  24   1   5  7  1
## seed    0 386 404  99 92 50
## total  31 410 405 104 99 51
```

Core (Seed) points for each cluster

```
# Plot DBSCAN results
plot(db, df, main = "DBSCAN", frame = FALSE)
```

WWW.KENT.EDU

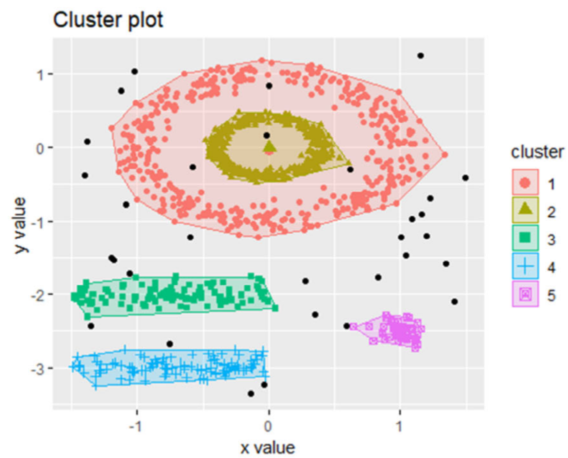
Now we apply the fpc package to a sample problem. The output identifies 31 border points. The values in the row seed indicate core points. A graphical depiction of this is shown in the next slide.



Noise, or outlier points are also identified as black circles in the graph.

Example II: Using fviz_cluster()

- It's also possible to draw the plot above using the function **fviz_cluster()** [factoextra package].
- `fviz_cluster(db, df, stand = FALSE, frame = FALSE, geom = "point")`

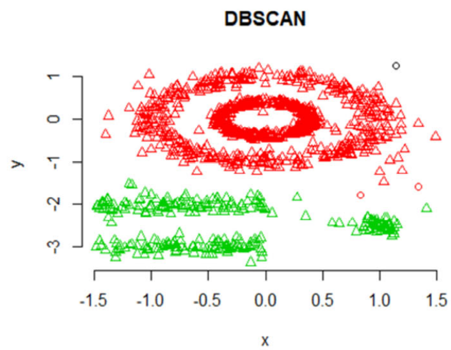


WWW.KENT.EDU

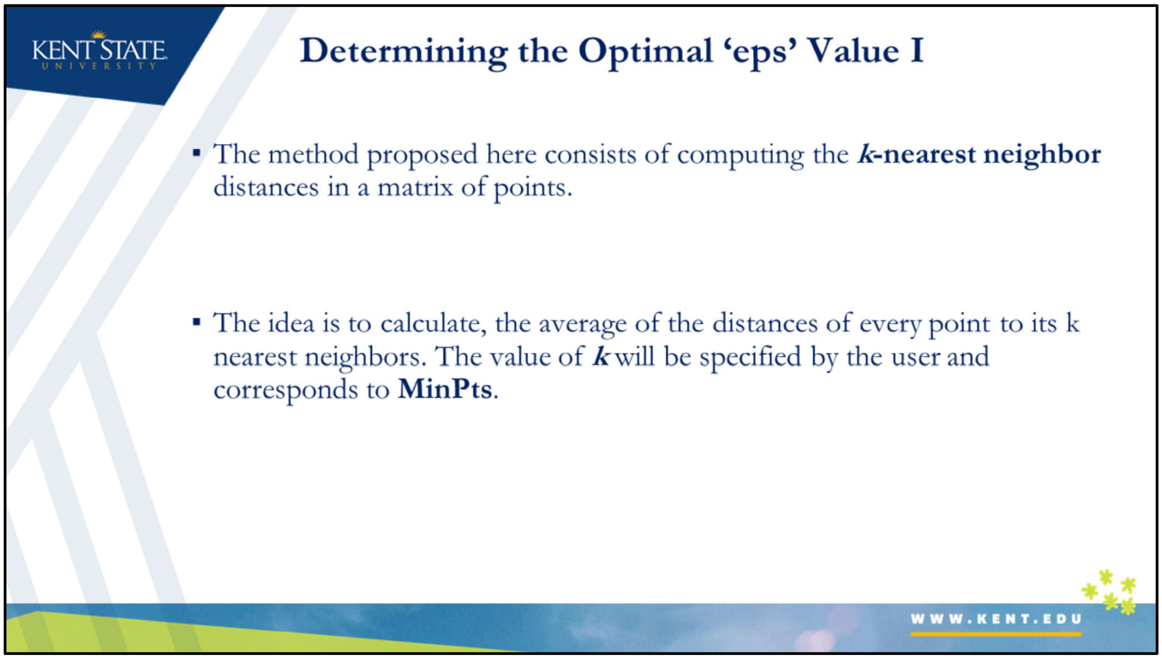
An alternative way to depict the previous graph.

Example II: Sensitivity to ‘eps’ Values

```r
db <- fpc::dbscan(df, eps = 0.5, MinPts = 5)
plot(db, df, main = "DBSCAN", frame = FALSE)
```



How do you choose eps? dbscan is sensitive to the choice of eps. Note here that there are fewer outliers and border points, in addition to fewer clusters.



The method proposed here uses the concept of k-nearest neighbor distances. We calculate the average distance of every point to its k nearest neighbors. The value of k is chosen by the modeler and corresponds to MinPts. We will then plot these distances against the sampled points, and then identify the value at which there is a sharp change in values. This is similar to identifying the elbow when determining k in k-means.

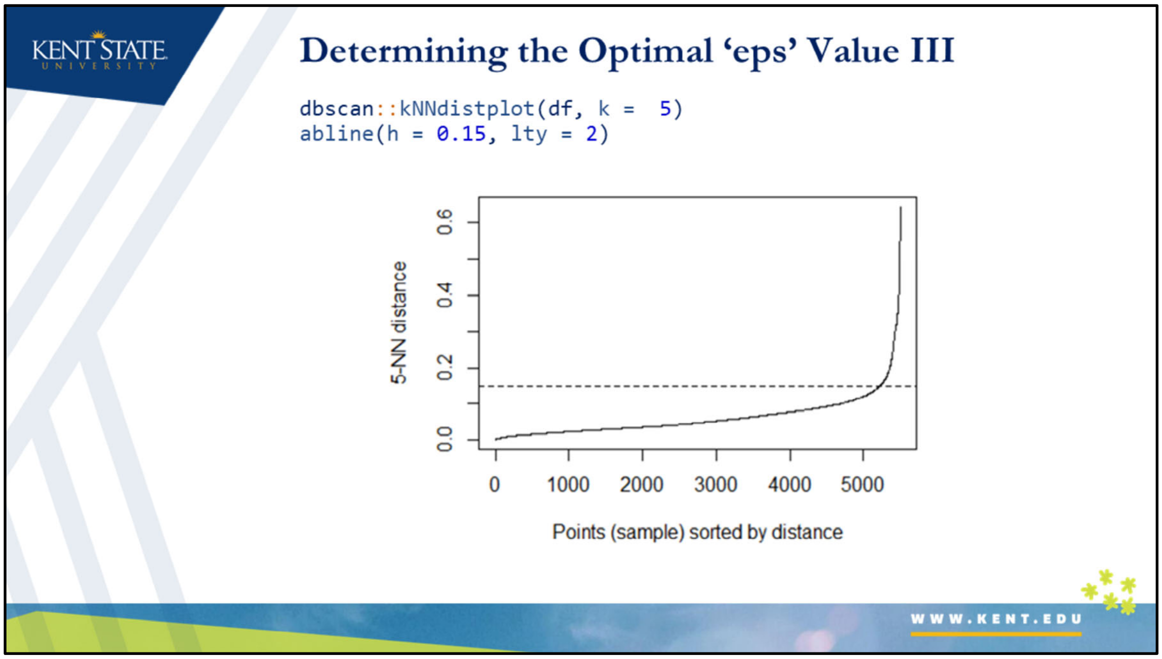
KENT STATE UNIVERSITY

Determining the Optimal 'eps' Value II

- Next, these ***k***-distances are plotted in an ascending order. The aim is to determine the "knee", which corresponds to the optimal eps parameter.
- A knee corresponds to a threshold where a sharp change occurs along the k-distance curve.
- The function **kNNdistplot()** [in dbscan package] can be used to draw the k-distance plot:

WWW.KENT.EDU

The plot of distance versus sampled points can be plotted using the kNNdistplot function, and is shown in the next slide.



The optimal eps value according the plot is approximately 0.15.

This concludes our discussion on dbscan.