**PART A:Perceptron**

When the binary perceptron is used to classify the new iris dataset where I have considered the Petal and sepal legth of the species Setosa and Versicolor. On passing this as input to the perceptron with the learning rate 1, the weight, bis and update returned are:  
$weight

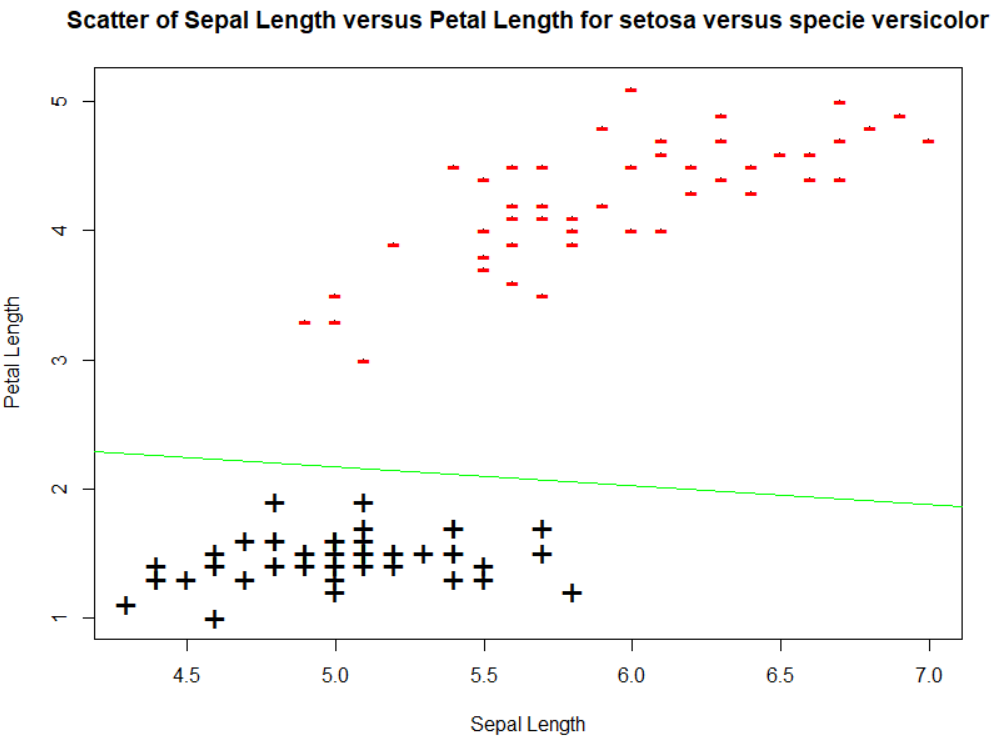
[1] -0.1419446 -0.9898746

$bias

[1] 2.855304

$updates

[1] 964



**Fig. showing Scatter Plot of Sepal Length versus Petal Length for setosa versus specie versicolor**   
  
When the data is randomized, I have radomized the dataset rowwise, and is passed to the perceptron the output returned remains unchanged:

$weight

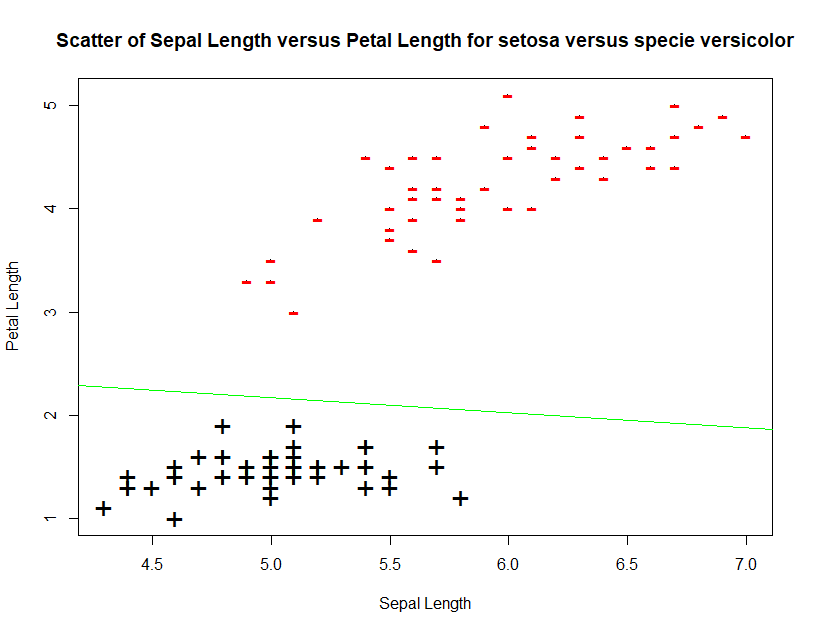
[1] -0.1419446 -0.9898746

$bias

[1] 2.855304

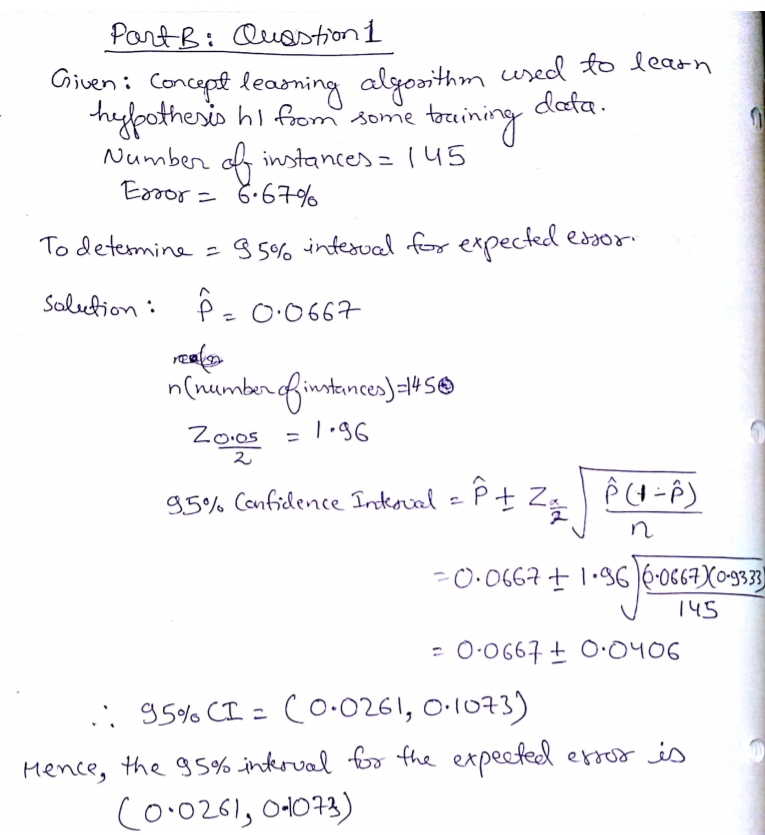
$updates  
[1] 964

The scatter plot also remains unchnaged over here.

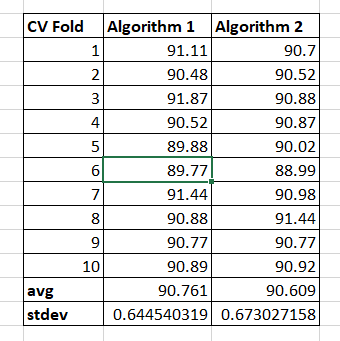


Learning rate is used to control how much we can change the bias determine how much we can control change in weight and bias. Weights and bias are updated to get a smaller error. In my case even with different learning rates the weigh,bias and updates remain the same. I tried different values of learning rates ranging form 0.1 to 5 but for all these cases the output i.e weight,bias and update remained the same i.e weight being -0.1419446, -0.9898746, bias being 2.855304 and update’s value is 964.

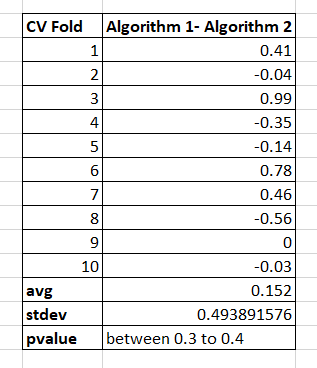
Yes, my algorithm always converges.

**PART B Question1:**

**Part B Question 2:**We start by assuming that both the algorithms Algorithm 1 and Algorithm 2 are equally accurate(Null Hypothesis). Further we calculate the average and standard deviation for both the algorithms as shown below:

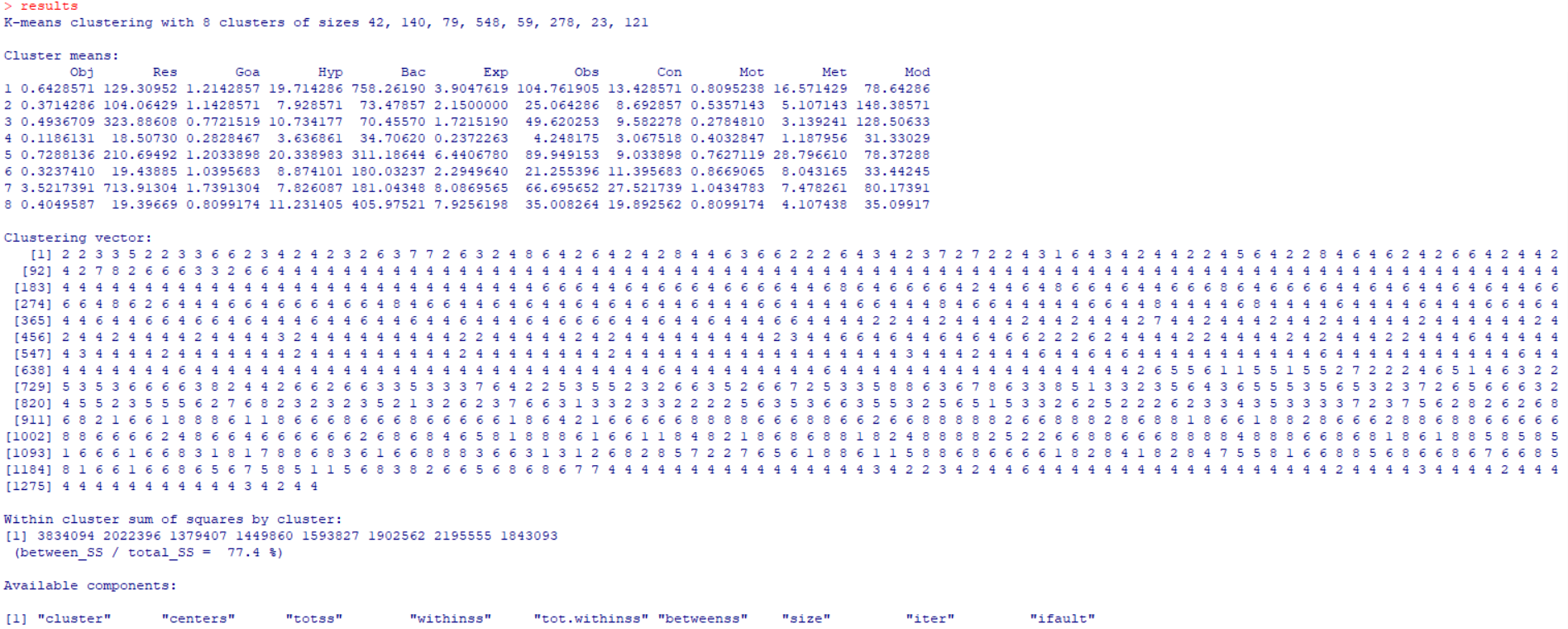


From the average we can see that the Average of Algorithm1 is better than the average of algorithm2.

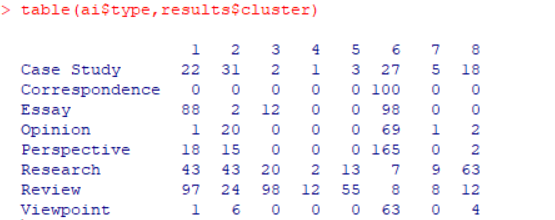
Further, we do a t-test which is a used to test whether the mean of two groups is same or different. hypothesis test and compute the value of t which is found out to be 0.9732. By referring to the t table the value of p is found out to be within the range 0.30 to 0.40 as shown below.   


Hence we can say with confidence ranging between 30% to 40% that algorithm 1 is better than algorithm 2.

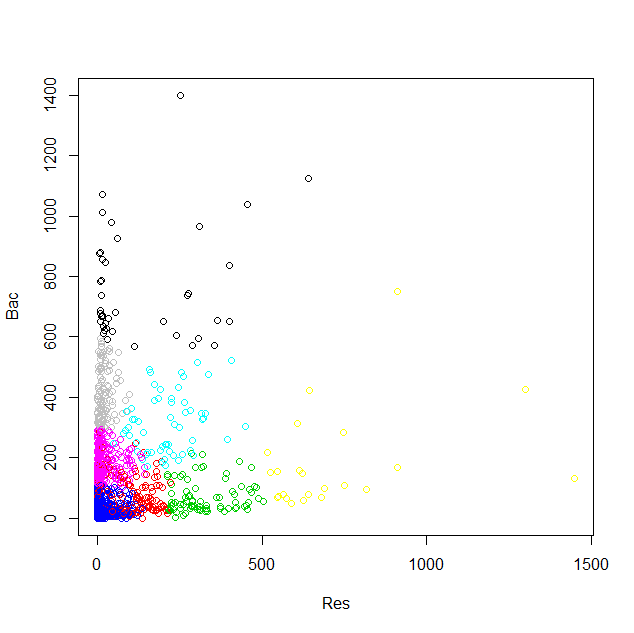
**Part B Question 3:**I have chosen three different types of clustering i.e K means clustering, DBscan and Hierarchical Clustering because all these three algorithms are one of the most popular clustering algorithms.  
  
Also,**K-Means** has the advantage that it’s pretty fast, as all we’re really doing is computing the distances between points and group centers; very few computations! It thus has a linear complexity O(n). **DBSCAN** poses some great advantages over other clustering algorithms. Firstly, it does not require a pe-set number of clusters at all. It also identifies outliers as noises unlike mean-shift which simply throws them into a cluster even if the data point is very different. Additionally, it is able to find arbitrarily sized and arbitrarily shaped clusters quite well. A particularly good use case of **hierarchical clustering** methods is when the underlying data has a hierarchical structure and you want to recover the hierarchy; other clustering algorithms can’t do this. These advantages of hierarchical clustering come at the cost of lower efficiency, as it has a time complexity of O(n³), unlike the linear complexity of K-Means and GMM.  
  
a) **K means clustering:** For implementing this type of clustering I loaded the csv file into the R and removed the column file and type from the dataset. After that the **results<- kmeans(ai,8)** line runs which, is used to run k means on the dataset to make 8 clusters, as the class variable “type” of the dataset has 8 different types. Further we check the value of the results:

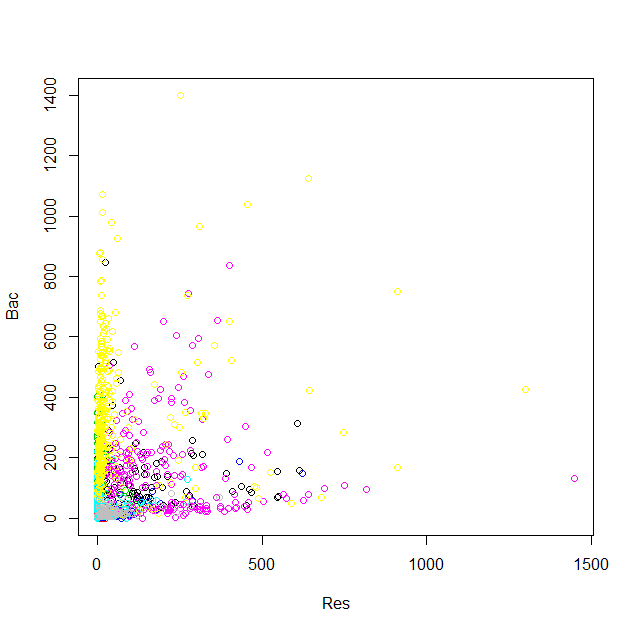


The results section shows that we have K means clustering with 8 clusters of respective sizes, as 42,140,79,548,59,278,23 and 121. Then the means of each cluster and each feature which are for Object, Result, Goal, Hypothesis, Background, Experiment, Observation, Conclusion, Method and Model for each of the eight clusters. Then the result shows clustering vector which cluster was each observation assigned to i.e. in our case the first and second observation are assigned to cluster2, the third result was assigned to cluster3 and so on.

Now, I determine how the clustering performed as compared to the real data. So we create a table and we look at the original **ai** dataset’s column type.  
**table(ai$type,results$cluster)**The above code generates the output as :  


Purpose here is to check how any of them match with the **results** clusters that we obtained here.

In order to Plot the clusters from the result variable we run the following line: **plot(ai[c("Res","Bac")],col=results$cluster)**

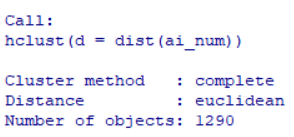
In the above plot we see that 8 different clusters generated in the result and seem to be quite good groups based on the types of the documents. In order to check how accurate, the groupings are we do the same plot again, but now based on actual class i.e. types, by executing below code: **plot(ai[c("Res","Bac")],col=ai$type)**

From the above plot it can be observed that the results didn’t change at all. So, it looks very similar to the k means clustering results group on the actual class i.e. types.

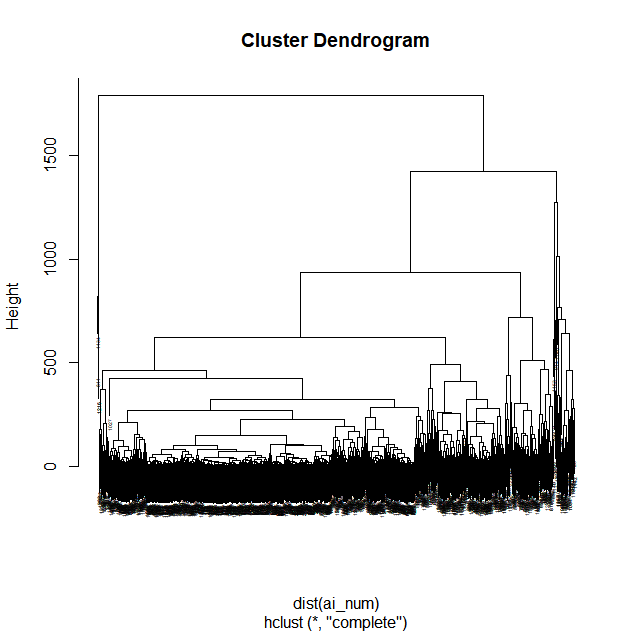
b) The second type of clustering that I have assumed over here is the **Hierarchical Agglomerative Clustering.** This algorithm tries to split the data points until it can get deviated. Unlike the K means clustering we cannot define the number of clusters here beforehand. It tries to divide the data points on the basis of distance. The following code was executed:  
**ai\_num<-ai** # assigning the ai dataset to ai\_num

**ai\_num$type<-NULL** # dropping the type column from the dataset

**ai\_num$file<-NULL** # dropping the file column from the dataset

**clust\_results<- hclust(dist(ai\_num))** # Hierarchical cluster analysis for the ai\_num dataset on the basis of distance and assigning the results to clust\_results.   
**clust\_results** # generates the output as shown below

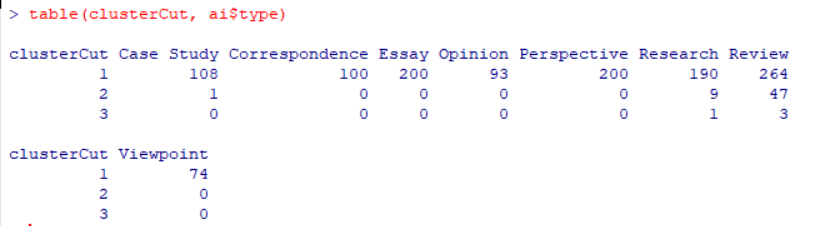
**plot(clust\_results,cex = 0.3)** # Plotting the cluster\_results generating a dendogram

The dendogram returned as an output is:  


We see from the above figure that the best choice for the total number of clusters are either 3 or 4.To do this we can cut the tree at the desired number of clusters using cutree.

**clusterCut<- cutree(cluster\_results,3)**

Then we compare the output clusters with the original document types:  
**table(clusterCut, ai$type**)



It looks like that the algorithm successfully classified the document types correspondence, essay, opinion, perspective and viewpoint into cluster 1, but had difficulty with review an research and casestudy.

c) **DBSCAN Clustering:**In order to implement the DBScan clustering the following steps were implemented:  
1. First of all the data frame as cleaned of the first and last column and then: **ai <- read.csv ("C:/Users/dkoec/downloads/ai2013\_papers (4).csv")**

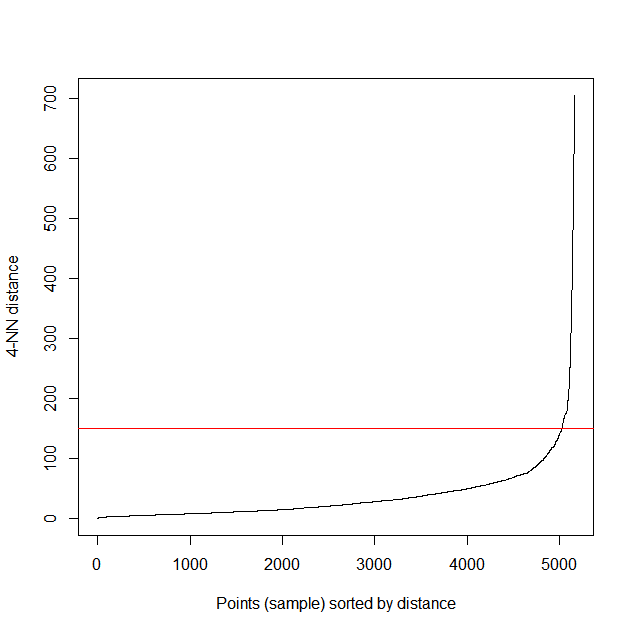
**ai$type<-NULL**

**ai$file<-NULL**

2. Then the clean dataset was converted to a matrix  
**ai\_mat<-as.matrix(ai)**

3.Then in order to determine the value of Epsilon for the dbscan I used the K nearest neighbor distances in the matrix of points

**kNNdistplot(ai\_mat, k=4)  
abline(h=150, col="red")**



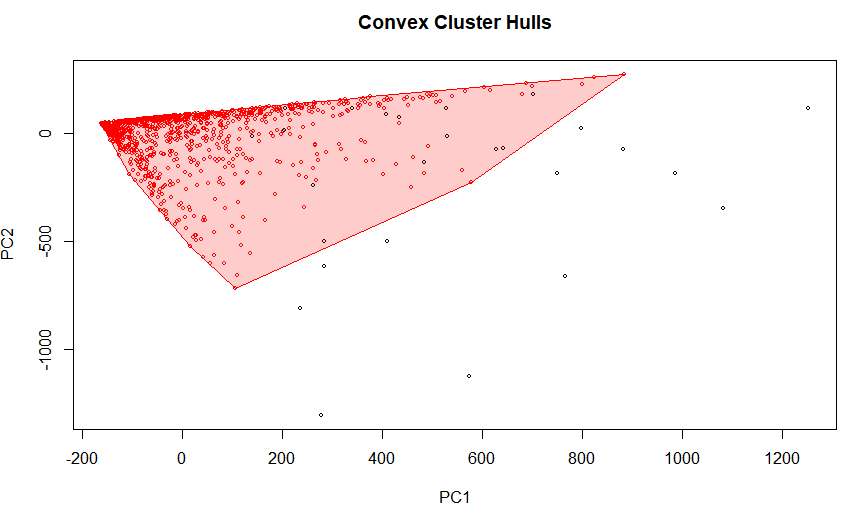
**4.**With the help of abline, it can be seen that the optimal epsilon value is around a distance of 155.

**set.seed(1234)** #The seed number you choose is the starting point used in the generation of a sequence of random numbers

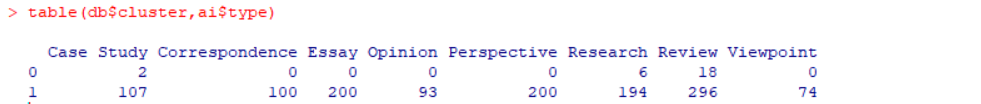
**fpc <- fpc::dbscan(ai\_mat, eps = 155, MinPts = 4)**# Compute DBSCAN using fpc package

**db <- dbscan::dbscan(ai\_mat, 155, 4)** # Compute DBSCAN using dbscan package

5. In the result of the DBScan its clearly visible that 1264 points have been assigned to the cluster1 and there are 26 noise points. Hullplot is used to show the separation among the clusters but the total number of clusters here is just 1 and the black points are the outliers.

**hullplot(ai\_mat,db$cluster)**6. In order to compare the data within each cluster we run:

ai <- read.csv("C:/Users/dkoec/downloads/ai2013\_papers (4).csv")#Load the file

table(db$cluster,ai$type)  
  
The result shows that correspondence, essay, opinion, perspective and viewpoint are all assigned to the cluster 1 whereas some of the points of case study, research and review remain unclustered and are categorized as noise.