Home Work 2

Adi Falach and Assaf Lovton

5/18/2021

Question 1:

1.2: The research contains data about Gene expression changes in human melanoma cell lines compared to primary melanocytes. Gene expression changes in 3 human melanoma cell lines were compared to freshly isolated normal primary melanocytes. 1.3:

```
coldata<-read.csv("coldata.csv",
                  header = T,
                  row.names=1,
                  stringsAsFactors=T)
coldata
```

```
##                     cell condition
## GSM2344965          FM_1    normal
## GSM2344966          FM_2    normal
## GSM2344967          FM_3    normal
## GSM2344968  SK_MEL_28_1    cancer
## GSM2344969  SK_MEL_28_2    cancer
## GSM2344970  SK_MEL_28_3    cancer
## GSM2344971 SK_MEL_147_1    cancer
## GSM2344972 SK_MEL_147_2    cancer
## GSM2344973 SK_MEL_147_3    cancer
## GSM2344974    UACC_62_1    cancer
## GSM2344975    UACC_62_2    cancer
## GSM2344976    UACC_62_3    cancer
```

1.4:

```
cts_data <- as.matrix(read.csv("GSE88741-GPL16791_series_matrix.txt.gz",skip
= 72,
                               header=T,sep="\t",row.names="ID_REF"))
cts<- cts_data[,c("GSM2344965","GSM2344966",
                  "GSM2344968","GSM2344969",
                  "GSM2344971","GSM2344972",
                  "GSM2344974","GSM2344975")] # choosing subsets
cts<-cts[complete.cases(cts), ]               #clean lines with NA
levels(coldata$condition)   #as we can see control group is second we need to
switch
```

```
## [1] "cancer" "normal"
```

```
levels(coldata$condition) <- c("cancer","normal")
coldata$condition <- relevel(coldata$condition, "normal")
levels(coldata$condition)   #check levels changed correctly
```

```
## [1] "normal" "cancer"
```

```
coldata_subset<-rbind(head(coldata,2), coldata[4:5,], coldata[7:8,],
coldata[10:11,])
library("DESeq2")
dds <- DESeqDataSetFromMatrix(countData = cts,
                              colData = coldata_subset,
                              design = ~ condition)
dds
```

```
## class: DESeqDataSet
## dim: 35238 8
## metadata(1): version
## assays(1): counts
## rownames(35238): A1BG A1CF ... BP-21890.2 YR211F11.2
## rowData names(0):
## colnames(8): GSM2344965 GSM2344966 ... GSM2344974 GSM2344975
## colData names(2): cell condition
```

```
head(assay(dds))
```

```
##         GSM2344965 GSM2344966 GSM2344968 GSM2344969 GSM2344971 GSM2344972
## A1BG           401        320        331        364        224        248
## A1CF             1          1          0          0          0          2
## A2M          23279      47606       2652       2707          4          9
## A2ML1            6          8          1          7          3          0
## A2MP1           21          7          0          6          0          0
## A3GALT2          0          0          1          1          0          0
##         GSM2344974 GSM2344975
## A1BG           756        391
## A1CF             1          0
## A2M          26725      10394
## A2ML1            9          4
## A2MP1           13          3
## A3GALT2          1          6
```

1.5:

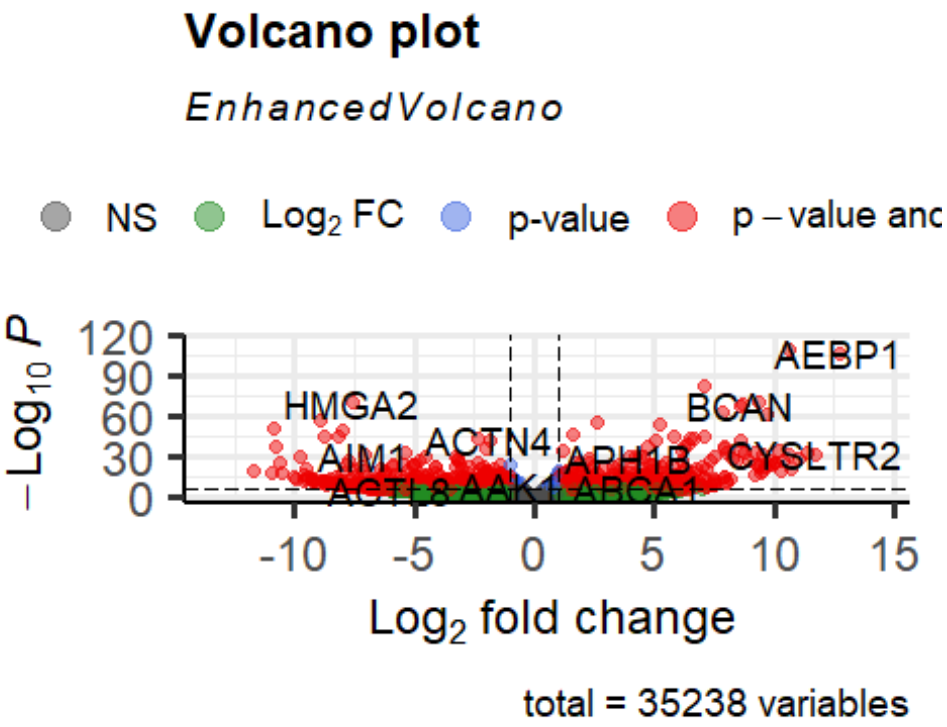
```
library("EnhancedVolcano")
```

```
## Loading required package: ggplot2
```

```
## Loading required package: ggrepel
```

```
## Registered S3 methods overwritten by 'ggalt':
##   method                  from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
```

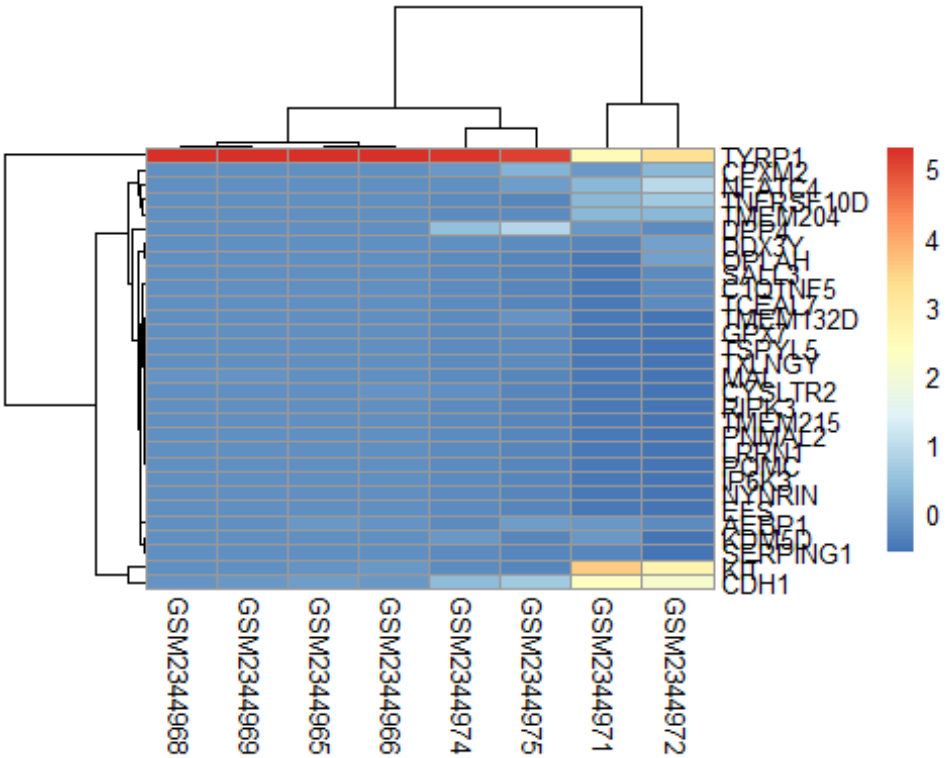
```
##   grobWidth.absoluteGrob  ggplot2
##   grobX.absoluteGrob      ggplot2
##   grobY.absoluteGrob      ggplot2

dds <- DESeq(dds, betaPrior=FALSE)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

res <- results(dds, contrast = c('condition','normal','cancer'))
res <- lfcShrink(dds,type = 'normal', contrast =
c('condition','normal','cancer'),res=res)

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than
type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2
vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895

EnhancedVolcano(res,
    lab = rownames(res),
    x = 'log2FoldChange',
    y = 'pvalue')

## Warning: Ignoring unknown parameters: xlim, ylim
```



1.6:

```
library("DESeq2")
library ("pheatmap")
library ("RColorBrewer")
nrow(dds)

## [1] 35238

dds <- dds[rowSums(counts(dds)) > 1,]
#remove empty lines
nrow(dds)

## [1] 24554

#normalize the data
rld<-rlog(dds, blind = FALSE)
head(assay(rld),3)

##      GSM2344965 GSM2344966 GSM2344968 GSM2344969 GSM2344971 GSM2344972
## A1BG  8.7373929   8.466012   8.484732   8.462814   7.807147  7.9658105
## A1CF -0.6926806  -0.693842  -1.016845  -1.035046  -1.037047 -0.5155002
## A2M  14.1438026  15.040088  11.455398  11.344552   5.455962  5.8204231
##      GSM2344974 GSM2344975
## A1BG  9.2824674   8.473682
## A1CF -0.7377403  -1.036638
## A2M  14.0946214  12.896629

#sampleDists<-dist(t(assay(rld)))
#sampleDists
```

```
dds_analysis <- DESeq(dds)

## using pre-existing size factors

## estimating dispersions

## found already estimated dispersions, replacing these

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

#alpha for fdr<0.05 and ifc for 8X
top30res.05andlfc3 <-results(dds_analysis, alpha = 0.05, lfcThreshold = 3)
table(top30res.05andlfc3$padj < 0.05)

##
## FALSE  TRUE
## 16151   282

#filter the top 30
top30res.05andlfc3<-
head(top30res.05andlfc3[order(top30res.05andlfc3$log2FoldChange),], 30)

final_data<- cts[rownames(top30res.05andlfc3),]
scaled_data <- scale(final_data)
pheatmap(scaled_data)
```




```
downregulated_30<-rownames(head(top30res.05andlfc3[
order(top30res.05andlfc3$log2FoldChange),], 30))
upregulated_30<-rownames(head(top30res.05andlfc3[
order(top30res.05andlfc3$log2FoldChange,decreasing = TRUE), ], 30))
print("here are the top 30 most downregulated")
```

```
## [1] "here are the top 30 most downregulated"
```

```
downregulated_30
```

```
##  [1] "AEBP1"     "CYSLTR2"   "NYNRIN"    "EFS"       "IP6K3"     "TSPYL5"
##  [7] "KIT"       "C1QTNF5"   "RIPK3"     "TMEM215"   "TCEAL7"    "KDM5D"
## [13] "CDH1"      "TYRP1"     "PNMAL2"    "TMEM132D"  "SERPING1"
"TNFRSF10D"
## [19] "DPP4"      "LRRN1"     "GPX7"      "SALL3"     "TXLNGY"    "DDX3Y"
## [25] "OPLAH"     "CPXM2"     "MAL"       "NFATC4"    "TMEM204"   "POMC"
```

```
print("here are the top 30 most upregulated")
```

```
## [1] "here are the top 30 most upregulated"
```

```
upregulated_30
```

```
##  [1] "POMC"      "TMEM204"   "NFATC4"    "MAL"       "CPXM2"     "OPLAH"
##  [7] "DDX3Y"     "TXLNGY"    "SALL3"     "GPX7"      "LRRN1"     "DPP4"
## [13] "TNFRSF10D" "SERPING1"  "TMEM132D"  "PNMAL2"    "TYRP1"     "CDH1"
## [19] "KDM5D"     "TCEAL7"    "TMEM215"   "RIPK3"     "C1QTNF5"   "KIT"
## [25] "TSPYL5"    "IP6K3"     "EFS"       "NYNRIN"    "CYSLTR2"   "AEBP1"
```

We will review the ANO9 gene as one of the top 30 most downregulated The protein encoded by this gene is a member of the TMEM16 (anoctamin) family of proteins, some of which form integral membrane calcium-activated chloride channels. The function of the encoded protein has yet to be elucidated, although it may have channel-forming abilities and also may have phospholipid scramblase activity. This gene has been observed to be upregulated in stage II and III colorectal cancers.

We will also review the SALL3 gene as one of the top 30 most upregulated This gene encodes a sal-like C2H2-type zinc-finger protein, and belongs to a family of evolutionarily conserved genes found in species as diverse as Drosophila, C. elegans, and vertebrates. Mutations in some of these genes are associated with congenital disorders in human, suggesting their importance in embryonic development. This protein binds to DNA methyltransferase 3 alpha (DNMT3A), and reduces DNMT3A-mediated CpG island methylation. It is suggested that silencing of this gene, resulting in acceleration of DNA methylation, may have a role in oncogenesis.

Question 2:

2.1: An exon is any part of a gene that will encode a part of the final mature RNA produced by that gene after introns have been removed by RNA splicing and refers to both the DNA sequence within a gene and to the corresponding sequence in RNA transcripts. when we mapping reads we want to analyze gene expression so we will be interested in mapping to exons that represent the genes.

2.2: Sequencing reads that map to intronic intervals may originate from several different sources such as overlapping genes. These confound measurements of the magnitude of true IR (intron retention). It is therefore crucial to correctly define the intronic intervals that will be used to measure IR. Second cause could be that the introns weren't spliced yet, meaning the RNA-seq is mRNA which contains introns.

2.3.1 Aliases for MARCH7 Gene:

- Membrane Associated Ring-CH-Type Finger 7
- Axotrophin
- MARCH-VII
- RNF177
- Membrane-Associated Ring Finger (C3HC4) 7, E3 Ubiquitin Protein Ligase
- RING-Type E3 Ubiquitin Transferase MARCHF7
- Membrane-Associated RING Finger Protein 7
- Membrane-Associated RING-CH Protein VII
- E3 Ubiquitin-Protein Ligase MARCHF7
- RING Finger Protein 177

• MARCHF7

• AXOT

• Membrane-Associated Ring Finger (C3HC4) 7

• RING-Type E3 Ubiquitin Transferase MARCH7

• E3 Ubiquitin-Protein Ligase MARCH7

• Membrane Associated Ring Finger 7

• EC 2.3.2.27

• EC 6.3.2

• MARCH7

• AXO

The databases these gene names were extracted from: Hugo Gene Numenclature Committee, NCBI Entrez Gene, Ensembl, UniprotKb/Swiss-Prot, DME.

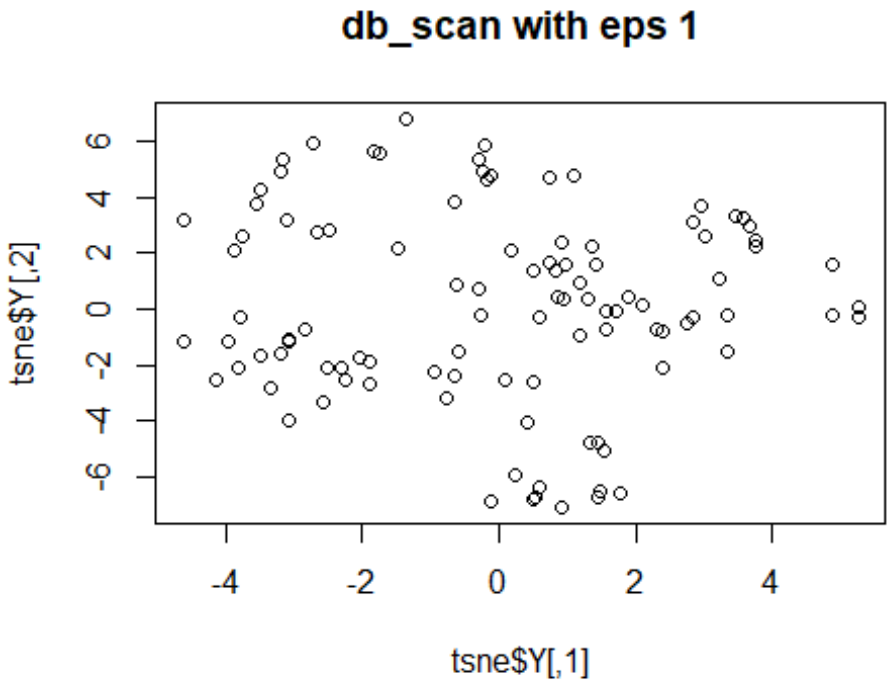
2.3.2: This gene has several names. if we will paste it manually into Excel sheet, a problem can arise when we accidentally use in the same sheet with several different names. This problem can cause problems in filtering and decoding the information. In addition, the name of the gene can sometimes be similar to date format or floating point, therefore the excel may convert it to a form that its opinion.

Question 3:

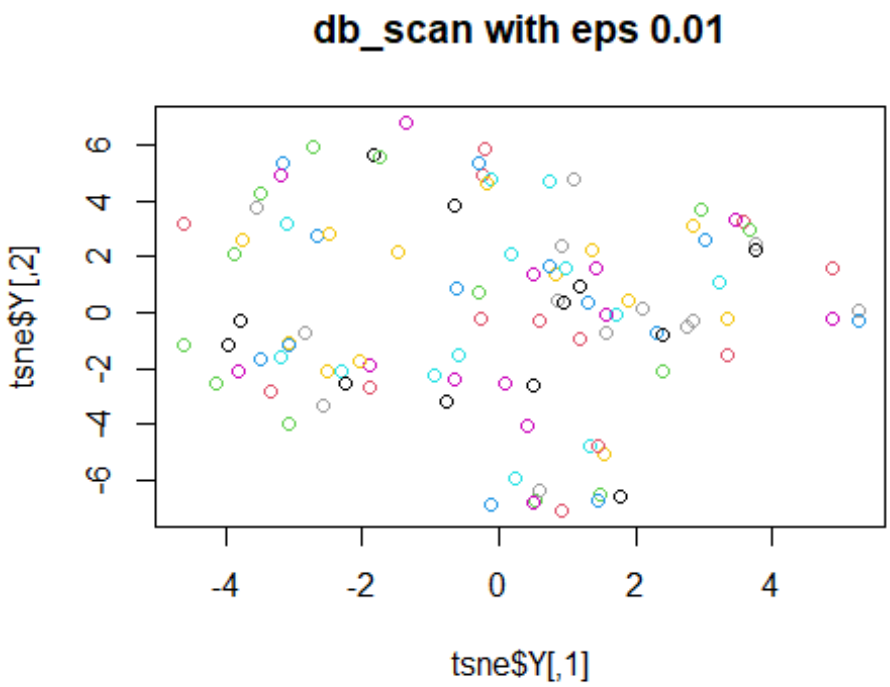
3.2: Playing with functions parameters: db_scan: 1.db_scan eps-size (radius) of the epsilon neighborhood. As we can see the results for sizes-0.01 and 0.001 produced the same results while eps = 0.1 produced a lower number of groups as predicated since the radius of is much bigger and can contain a larger numbers of elements in each group. for eps = 1 the radius was too big since all the elements finished in same group. Since the eps = 0.1 appeared to produce the best results we have decided to try changing the minPts value we ran 2 test one with minPts = 1 and one with minPts = 5. 2.minPts responsible for the number of minimum points required in the eps neighborhood for core points. as we can see when we asked for minPts = 5 all the clusters with less than 5 elements where removed. We will use this kind of method when we have a lot of small clusters that we don't care about and we want to fucous on the main ones.

```
#running db scan
# eps-size (radius) of the epsilon neighborhood. Can be omitted if x is a
frNN object.
# minPts number of minimum points required in the eps neighborhood for core
points
#(including the point itself).
db_scan_eps1 <- dbscan(dist(t(scores)), eps = 1, minPts = 1)
db_scan_eps0.1andminPts1 <- dbscan(dist(t(scores)), eps = 0.1, minPts = 1)
```

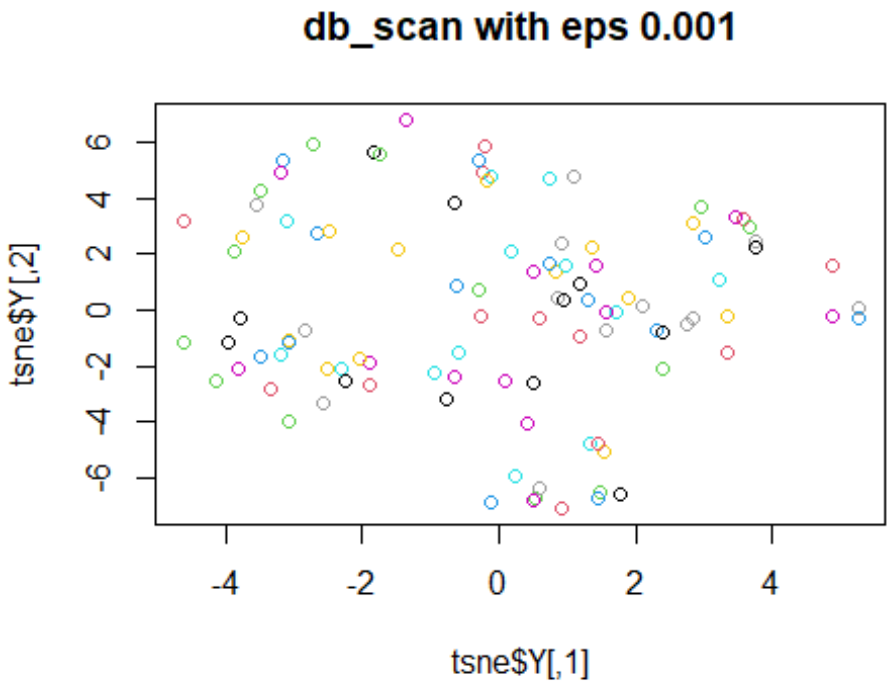
```r
db_scan_eps0.01 <- dbscan(dist(t(scores)), eps = 0.01, minPts = 1)
db_scan_eps0.001 <- dbscan(dist(t(scores)), eps = 0.001, minPts = 1)
db_scan_eps0.1andminPts5 <- dbscan(dist(t(scores)), eps = 0.1, minPts = 5)

#drawing dc_scan using Rtsne with different parameters:
plot(tsne$Y, col = db_scan_eps1$cluster, main="db_scan with eps 1")
```



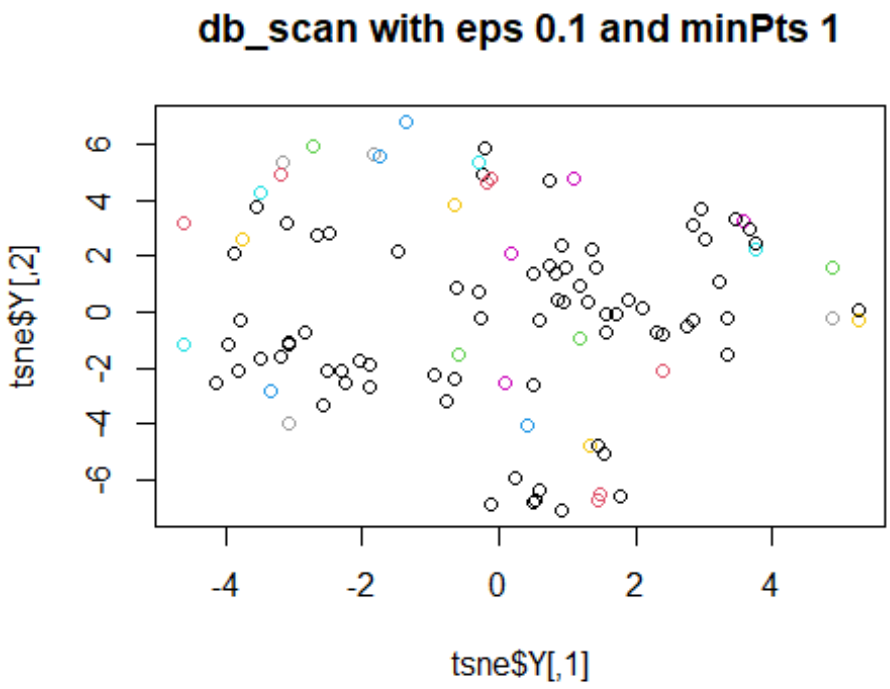

```r
plot(tsne$Y, col = db_scan_eps0.01$cluster, main="db_scan with eps 0.01")
```



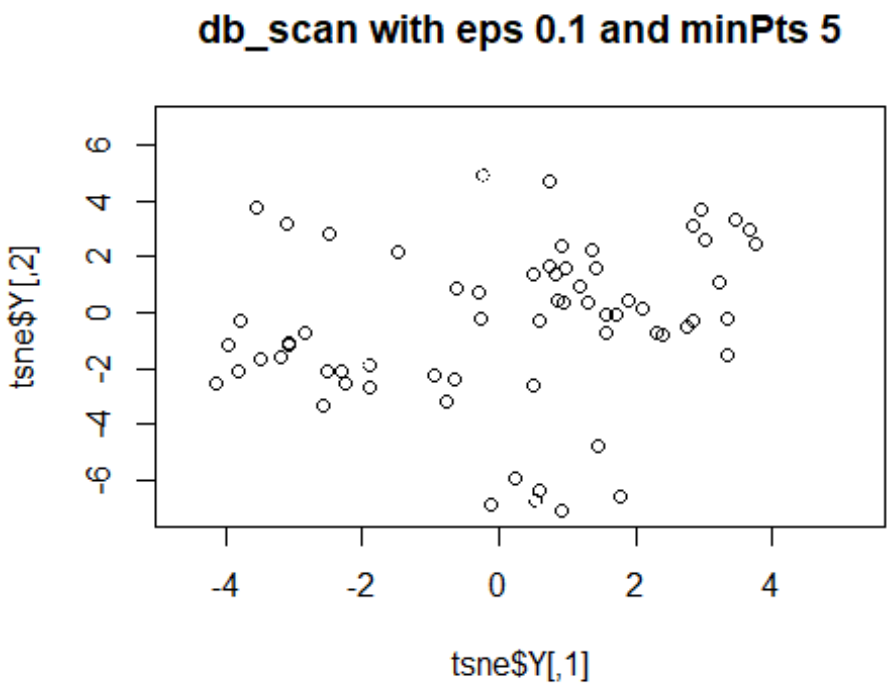

```
plot(tsne$Y, col = db_scan_eps0.001$cluster, main="db_scan with eps 0.001")
```



```
plot(tsne$Y, col = db_scan_eps0.1andminPts1$cluster, main="db_scan with eps
0.1 and minPts 1")
```




```
plot(tsne$Y, col = db_scan_eps0.1andminPts5$cluster, main="db_scan with eps
0.1 and minPts 5" )
```



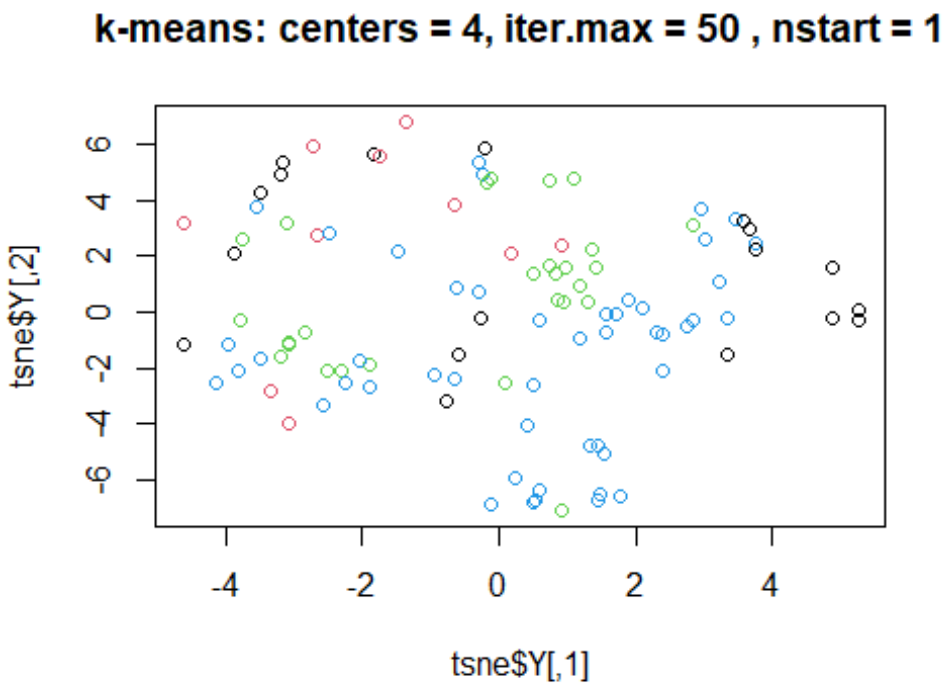
k-means:

1.iter.max is the number of times the algorithm is run before results are returned. As we cam see the larger the number of iterations we provide the separation is better, with 10 iterations the clusters were very mixed while after 50 itrations we got a better separation.

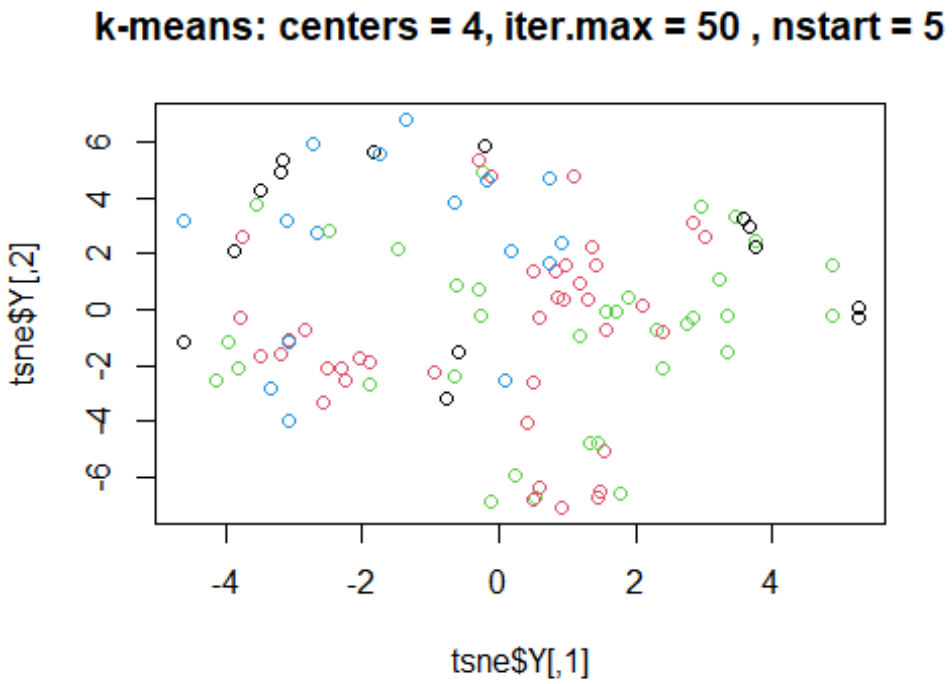
2.nstart represents the number of random data sets used to run the algorithm. This parameter is responsible for the number of times the algorithm will run, it chooses the best result out of them, so for nstart = 1 the algorithm will run once while with 40 it will run 40 times and returns the minimal (best) solution. As predictaed the results were better for a larger nstart.

```
#running kmeans with different numbers of centers
#iter.max is the number of times the algorithm is run before results are
returned.
#nstart represents the number of random data sets used to run the algorithm
k_means_with_4clu_iter50_1start<-kmeans(dist(t(scores)), centers = 4,
iter.max = 50,nstart = 1 )
k_means_with_4clu_iter50_5start<-kmeans(dist(t(scores)), centers = 4,
iter.max = 50,nstart = 5 )
k_means_with_4clu_iter50_10start<-kmeans(dist(t(scores)), centers = 4,
iter.max = 50,nstart = 10 )
k_means_with_4clu_iter50_20start<-kmeans(dist(t(scores)), centers = 4,
iter.max = 50,nstart = 20 )

k_means_with_4clu_iter10_10start<-kmeans(dist(t(scores)), centers = 4,
```

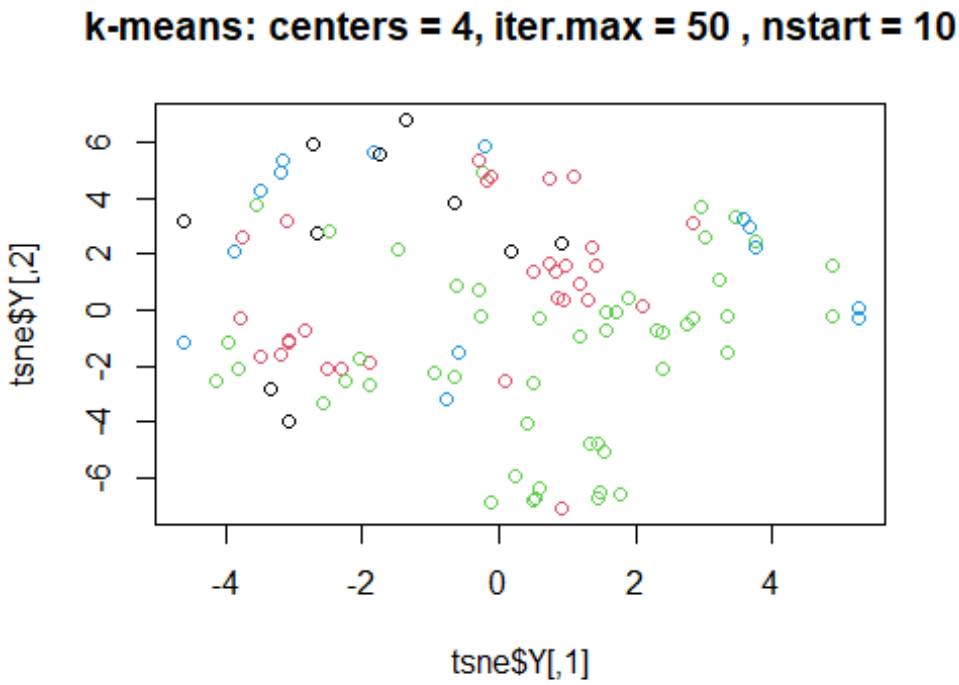
```
iter.max = 10,nstart = 10 )
k_means_with_4clu_iter20_10start<-kmeans(dist(t(scores)), centers = 4,
iter.max = 20,nstart = 10 )
k_means_with_4clu_iter30_10start<-kmeans(dist(t(scores)), centers = 4,
iter.max = 30,nstart = 10 )
k_means_with_4clu_iter40_10start<-kmeans(dist(t(scores)), centers = 4,
iter.max = 40,nstart = 10 )

#drawing kmeans using Rtsn
plot(tsne$Y, col = k_means_with_4clu_iter50_1start$cluster, main="k-means:
centers = 4, iter.max = 50 , nstart = 1")
```



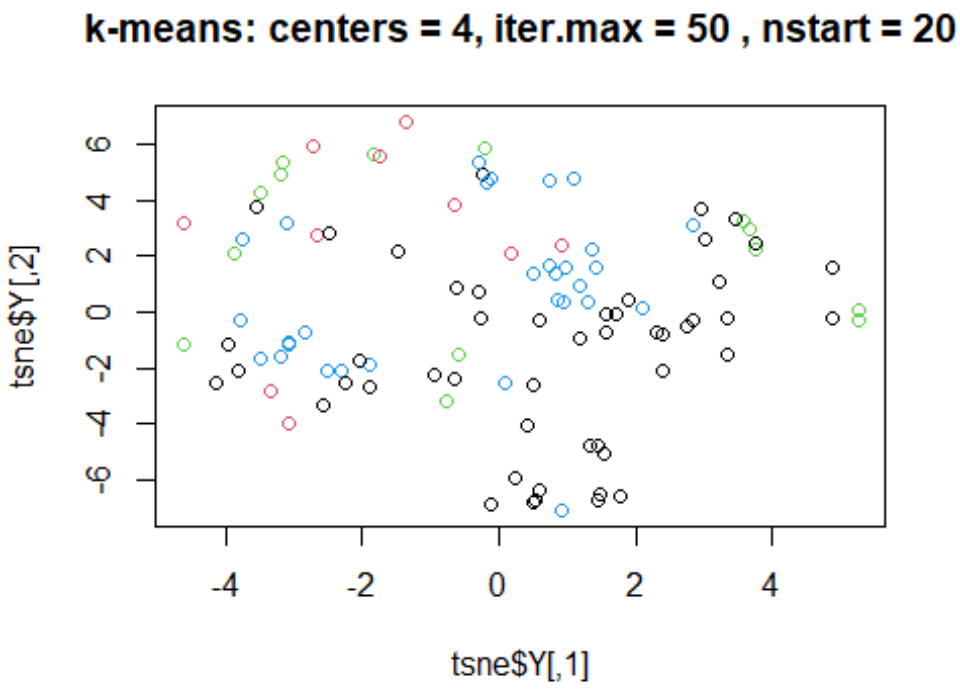

```
plot(tsne$Y, col = k_means_with_4clu_iter50_5start$cluster, main="k-means:
centers = 4, iter.max = 50 , nstart = 5")
```



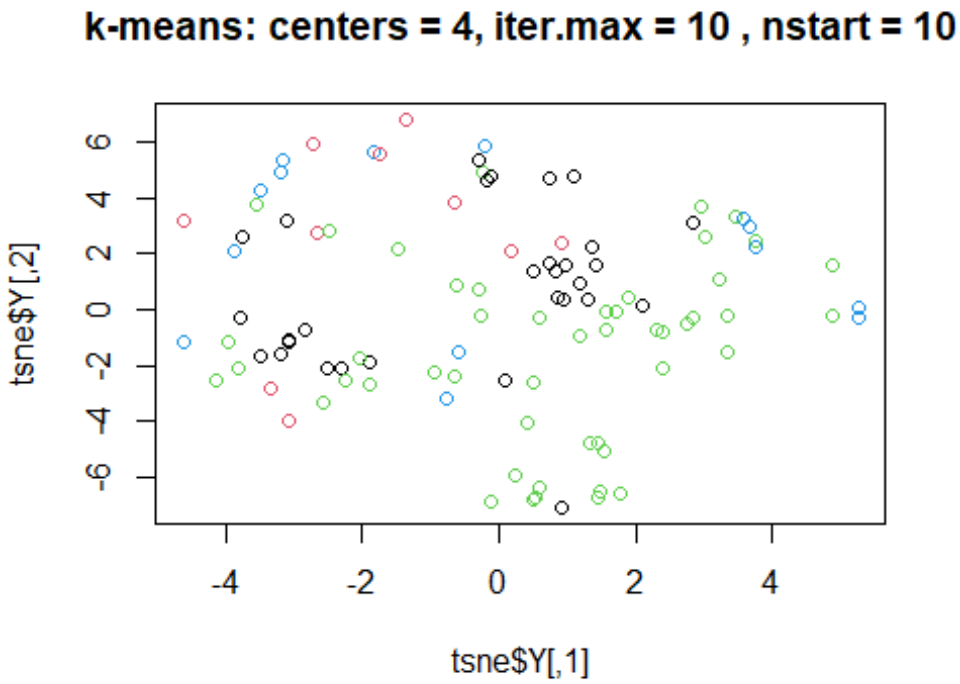

```
plot(tsne$Y, col = k_means_with_4clu_iter50_10start$cluster, main="k-means:
centers = 4, iter.max = 50 , nstart = 10")
```



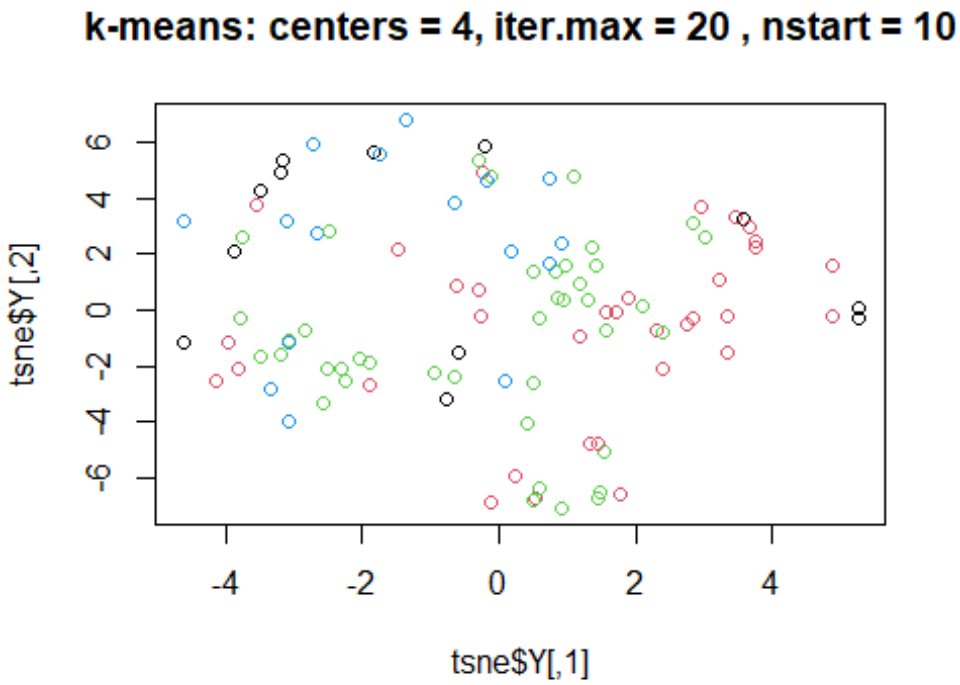
```
plot(tsne$Y, col = k_means_with_4clu_iter50_20start$cluster, main="k-means:
centers = 4, iter.max = 50 , nstart = 20")
```



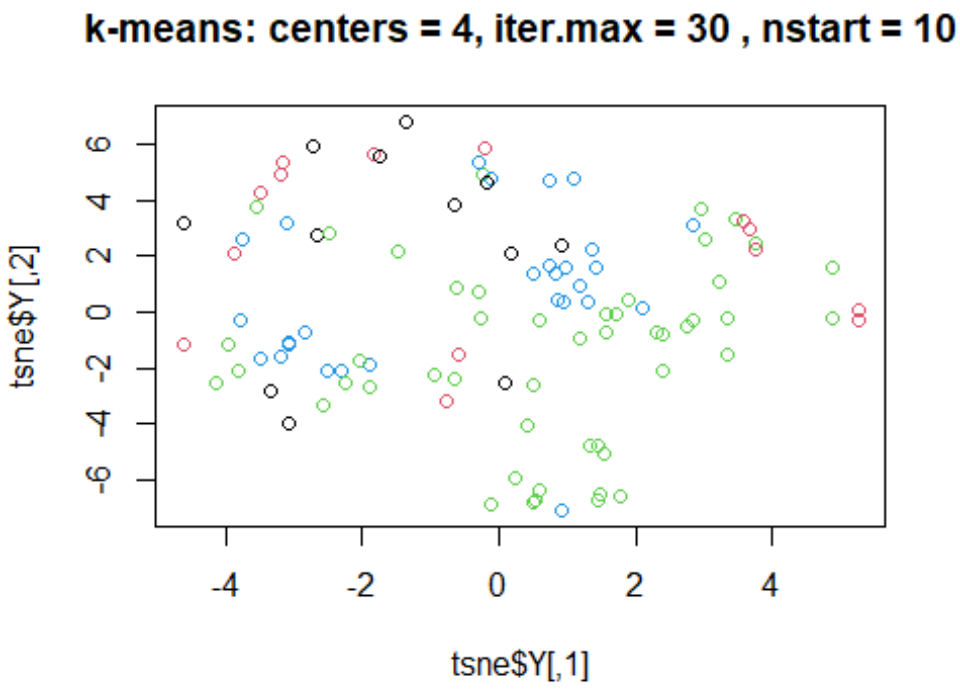

```
plot(tsne$Y, col = k_means_with_4clu_iter10_10start$cluster, main="k-means:
centers = 4, iter.max = 10 , nstart = 10")
```



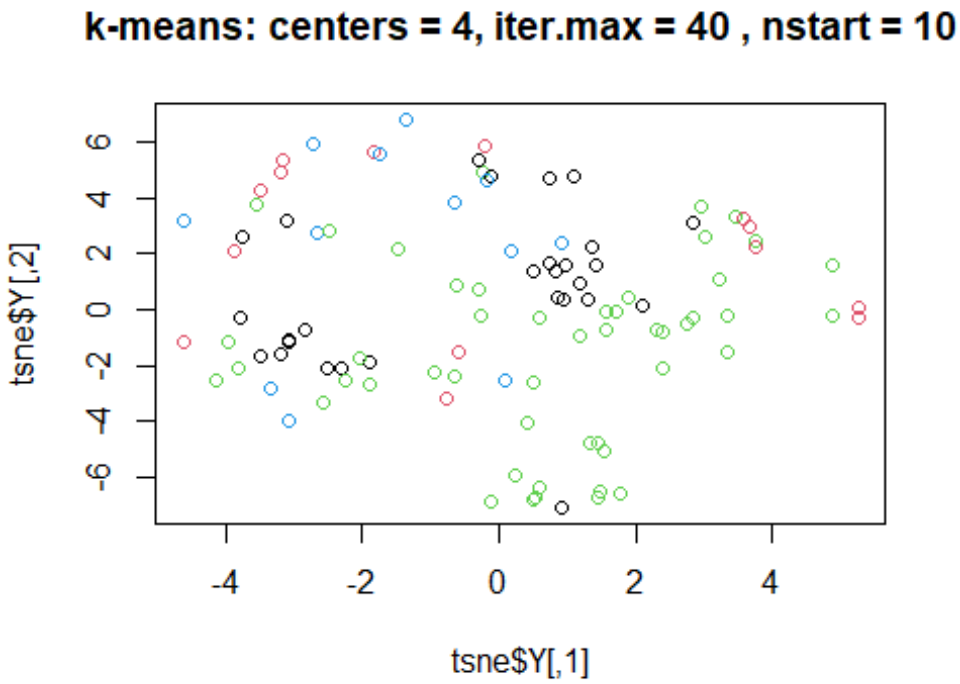

```
plot(tsne$Y, col = k_means_with_4clu_iter20_10start$cluster, main="k-means:
centers = 4, iter.max = 20 , nstart = 10")
```



```
plot(tsne$Y, col = k_means_with_4clu_iter30_10start$cluster, main="k-means:
centers = 4, iter.max = 30 , nstart = 10")
```



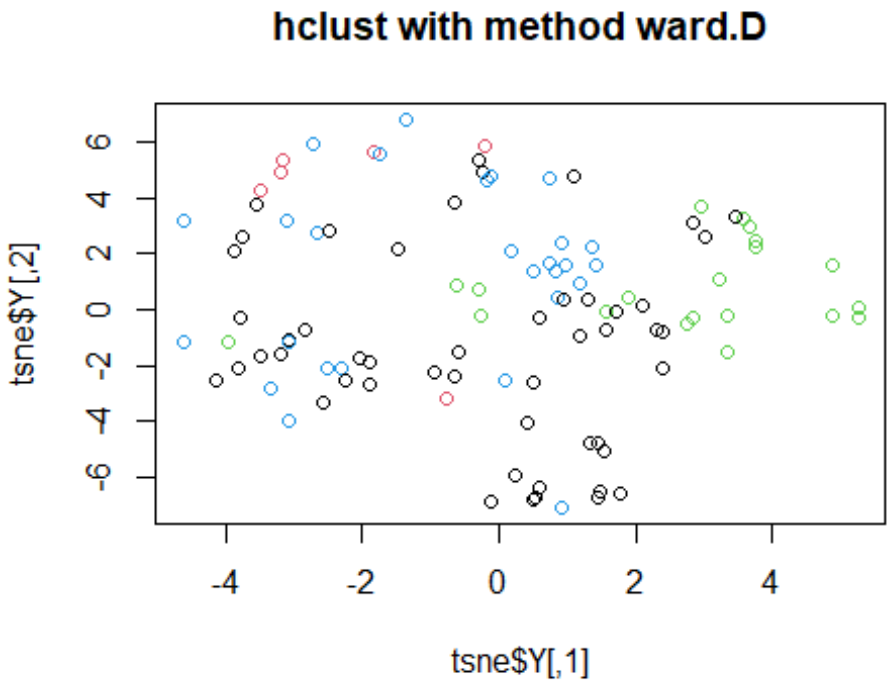

```
plot(tsne$Y, col = k_means_with_4clu_iter40_10start$cluster, main="k-means:
centers = 4, iter.max = 40 , nstart = 10")
```



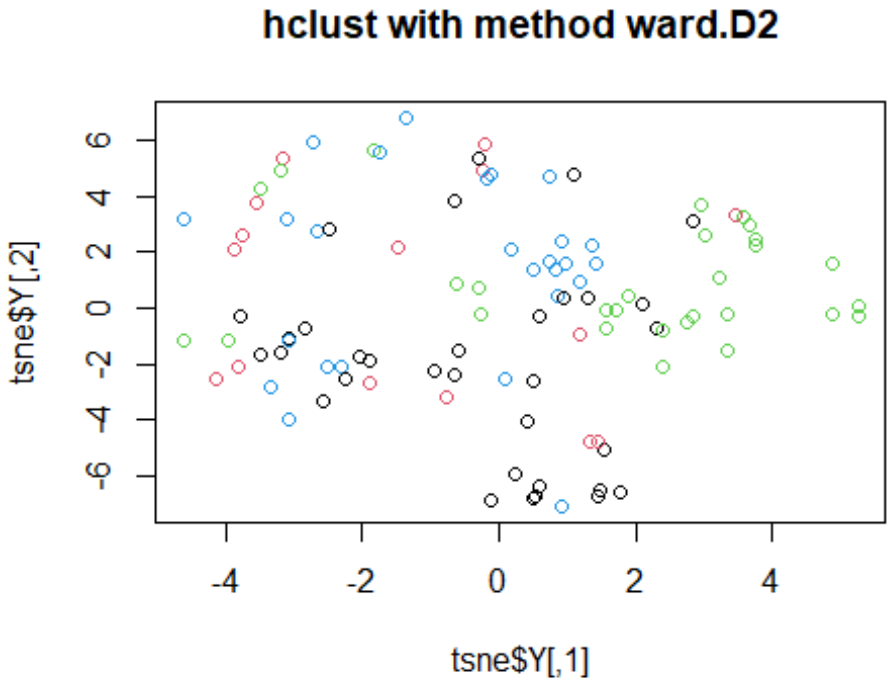
hclust:

method: the agglomeration method to be used we tried many methods- wardD and wardD2 supported resonable results while the single and centroid provided the same results, with one big cluster and three small ones with only one element in each cluster. Average and median also produced simmilar results with one big cluster and the rest of them were fairly small clusters.

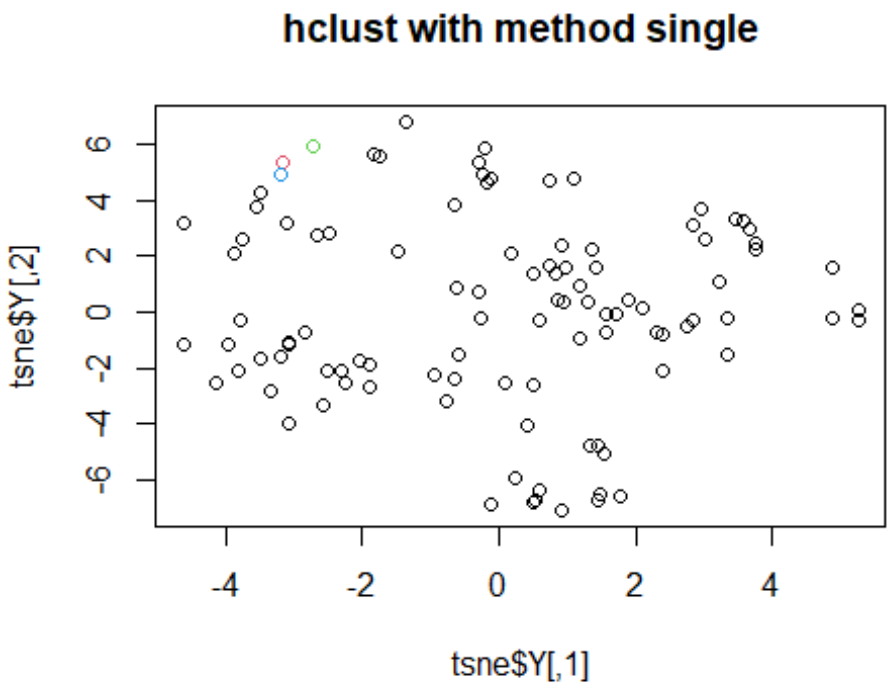
```
#running hclust
h_clust_wardD<-hclust(dist(t(scores)), method = "ward.D", members = NULL)
h_clust_wardD2<-hclust(dist(t(scores)), method = "ward.D2", members = NULL)
h_clust_single<-hclust(dist(t(scores)), method = "single", members = NULL)
h_clust_centroid<-hclust(dist(t(scores)), method = "centroid", members =
NULL)
h_clust_median<-hclust(dist(t(scores)), method = "median", members = NULL)
h_clust_average<-hclust(dist(t(scores)), method = "average", members = NULL)
#drawing hclust using Rtsne
groups<-cutree(h_clust_wardD, k=4)
plot(tsne$Y, col = groups, main="hclust with method ward.D")
```



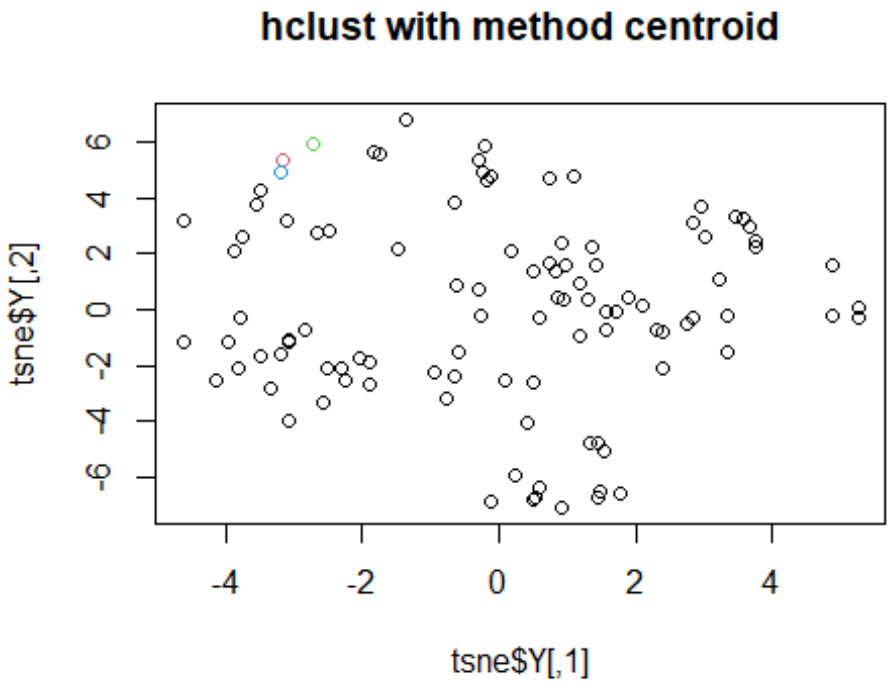

```
groups<-cutree(h_clust_wardD2, k=4)
plot(tsne$Y, col = groups, main="hclust with method ward.D2")
```



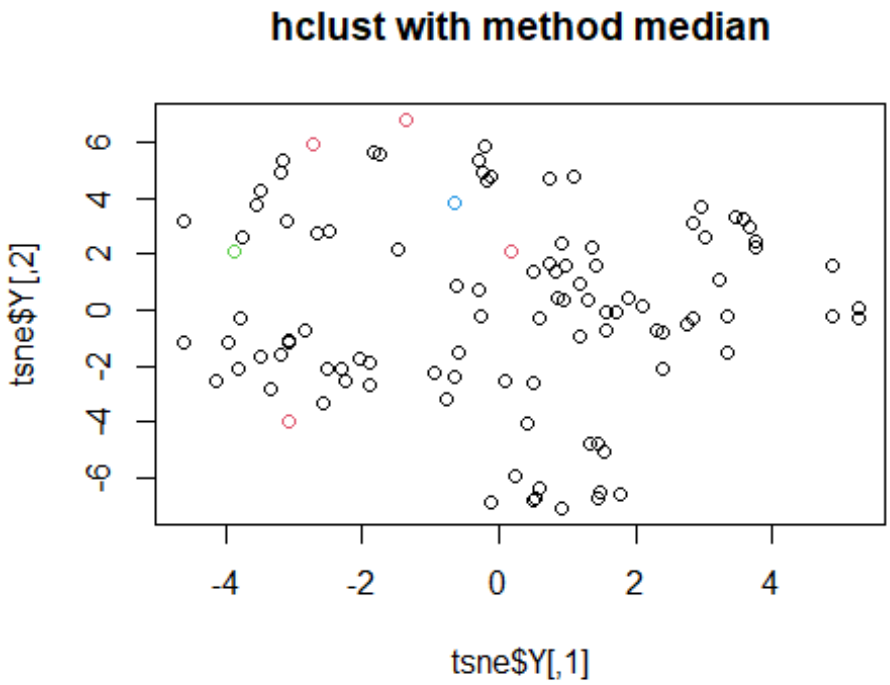
```
groups<-cutree(h_clust_single, k=4)
plot(tsne$Y, col = groups, main="hclust with method single")
```



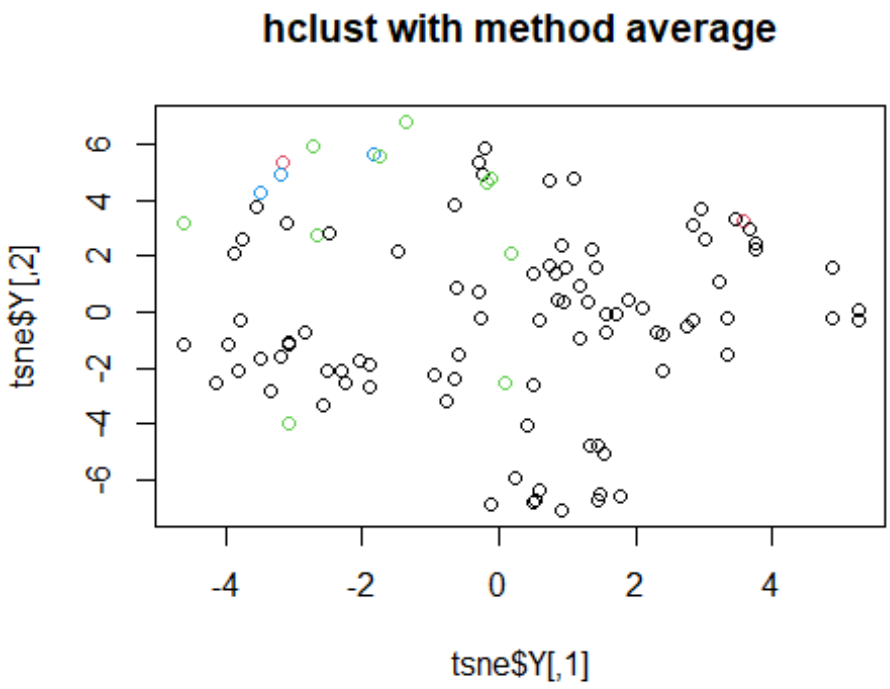

```
groups<-cutree(h_clust_centroid, k=4)
plot(tsne$Y, col = groups, main="hclust with method centroid")
```




```
groups<-cutree(h_clust_median, k=4)
plot(tsne$Y, col = groups, main="hclust with method median")
```

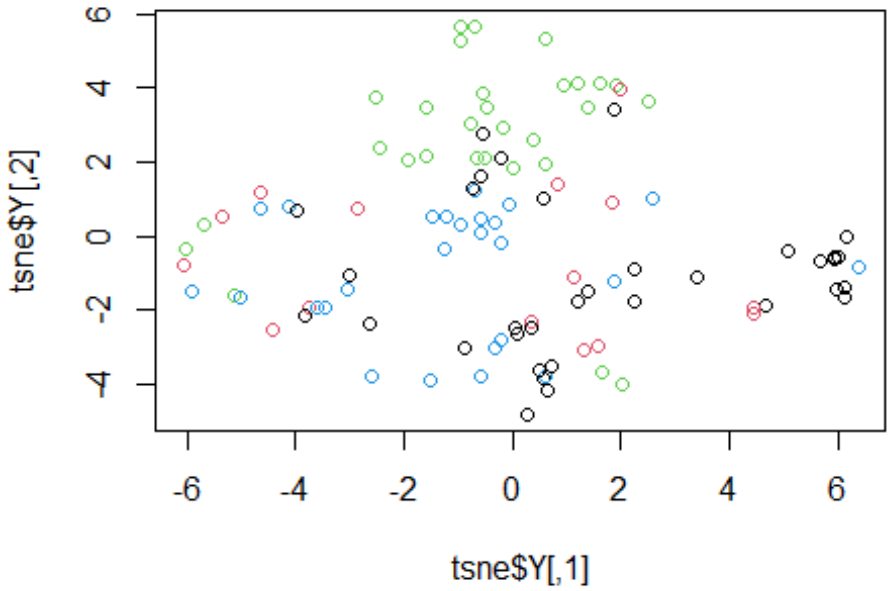


```
groups<-cutree(h_clust_average, k=4)
plot(tsne$Y, col = groups, main="hclust with method average")
```

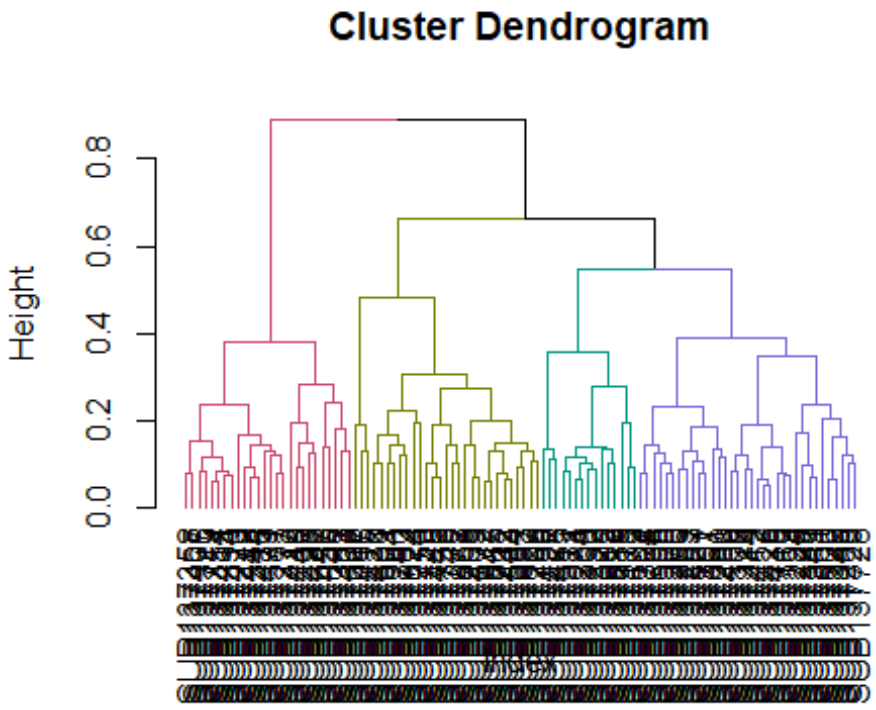


3.3: we chose the wardD2 function since it seems to provide the best results. here you can see the dendogram of the best result in our opinion:

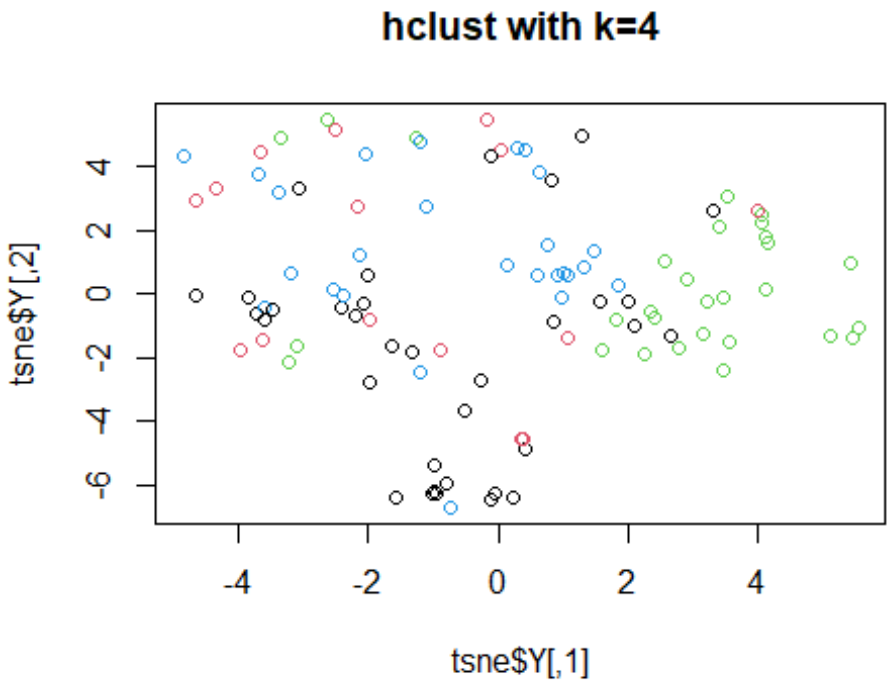
```
library(dendextend)
h_clust<-hclust(dist(t(scores)), method = "ward.D2", members = NULL)

groups<-cutree(h_clust, k=4)
tsne = Rtsne(t(sdy$expr))
plot(tsne$Y, col = groups)
```



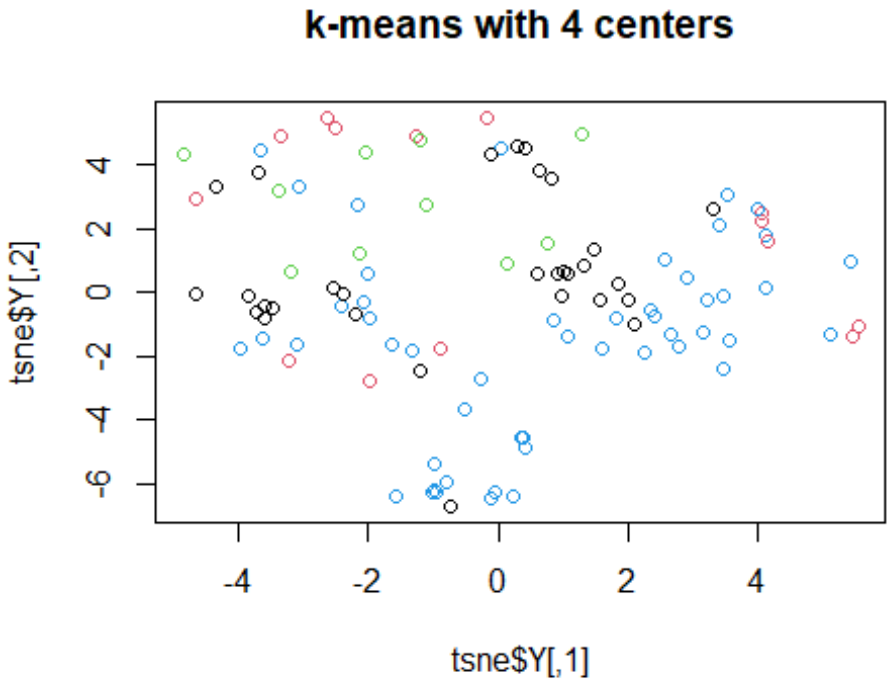

```
dend<-as.dendrogram(hclust(dist(t(scores)), method = "ward.D2", members =
NULL))
d1=color_branches(dend,k=5, col = c(3,1,1,4,1))
d2=color_branches(d1,k=4) # auto-coloring 5 clusters of branches.
plot(d2, labels = NULL, hang = 0.1, check = TRUE,
     axes = TRUE, frame.plot = FALSE, ann = TRUE,
     main = "Cluster Dendrogram",
     sub = NULL, xlab = NULL, ylab = "Height")
```



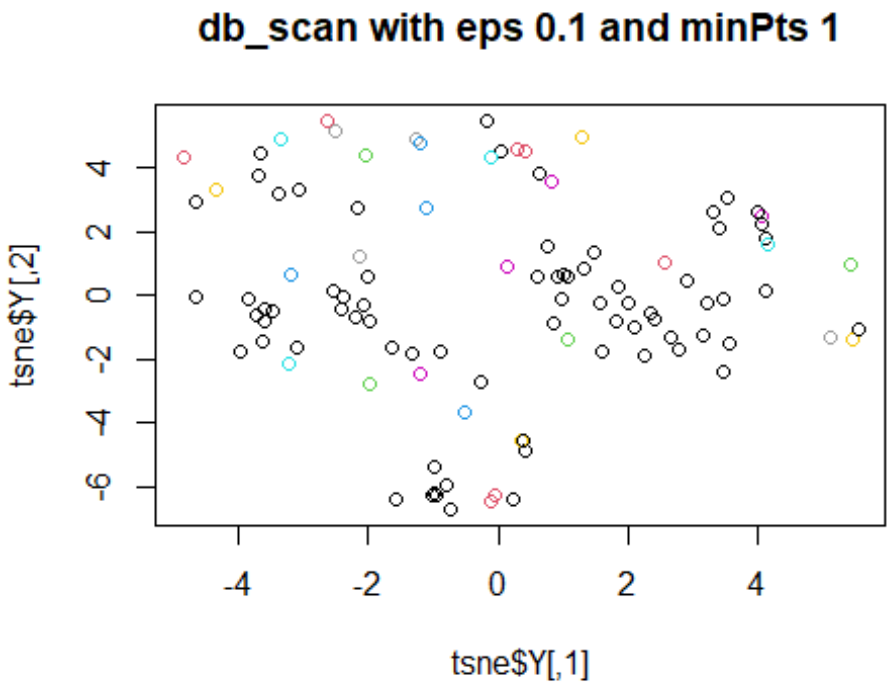

```r
sdy <- readRDS("sdy420.rds")
cell.types.use =
intersect(colnames(xCell.data$spill.array$K),rownames(sdy$fcs))
scores <- xCellAnalysis(sdy$expr, rnaseq=F, cell.types.use = cell.types.use)

#running db scan
db_scan_eps0.1andminPts1 <- dbscan(dist(t(scores)), eps = 0.1, minPts = 1)


#running hclust
h_clust<-hclust(dist(t(scores)), method = "ward.D2", members = NULL)


#running kmeans with different numbers of centers
k_means_with_4<-kmeans(dist(t(scores)), centers = 4)

tsne = Rtsne(t(sdy$expr))

#drawing hclust using Rtsne
groups<-cutree(h_clust, k=4)
plot(tsne$Y, col = groups, main="hclust with k=4")
```




```r
#drawing kmeans using Rtsne
plot(tsne$Y, col = k_means_with_4$cluster, main="k-means with 4 centers")
```



```
#drawing dc_scan using Rtsne with different parameters:
plot(tsne$Y, col = db_scan_eps0.1andminPts1$cluster, main="db_scan with eps
0.1 and minPts 1")
```



As we can see the different methods got us different results, the hclust and kmeans got different results for the same number of clusters. We believe the difference comes from the way the algorithms works, their stop condition and the iteration process.we will explain about the differences dieriving from the way that the algorithms works later on. Also we can see that the db_scan produced very bad results, with over seven clusters, there is a one main cluster while the other are very small.

- K-means is a centroid-based or partition-based clustering algorithm. This algorithm partitions all the points in the sample space into K groups of similarity.

  K Means clustering is found to work well when the structure of the clusters is hyper spherical (like circle in 2D, sphere in 3D). In addition, it is more efficient for varying densities of the data points. On the other hand, it does not work well with outliers and noisy datasets.

- Hierarchical clustering, as the name suggests is an algorithm that builds hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.

  Hierarchical clustering doesn't work as well as k means when the shape of the clusters is hyper spherical.

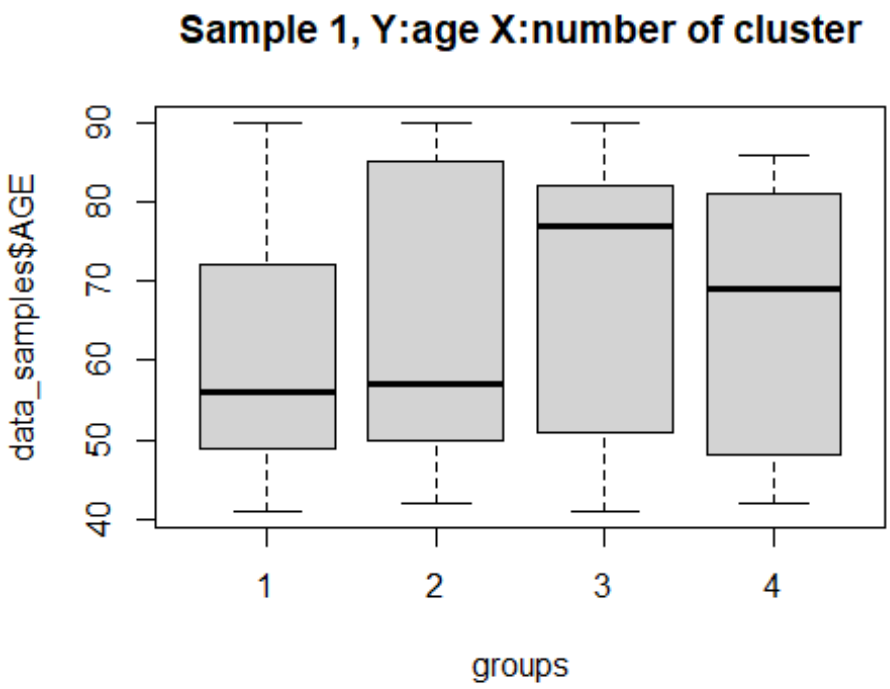
- DBScan is a density-based clustering algorithm. The key fact of this algorithm is that the neighborhood of each point in a cluster which is within a given radius must have a minimum number of points.

  This algorithm has proved extremely efficient in detecting outliers and handling noise. In addition, clusters formed are arbitrary in shape and may not have same feature size, it does not work very well for sparse datasets or for data points with varying density.

To conclude the k means and the hclust seems to be the better options in this case while the dbscan fail to give us meaningful results.

3.5: Accordingly to the results below we can see that the race and the gender were not showing a distinguishing attribute, But! we noticed that in groups 1 and 2 had a very similar age while clusters 3 and 4 average age was higher in at least 20 years in average.

```
data_samples<-read.csv("sample_info.csv",
                       header = T,
                       row.names=1,
                       stringsAsFactors=T)

boxplot(data_samples$AGE~groups, main="Sample 1, Y:age X:number of cluster")
```




```
table(data_samples$GENDER, groups)

##         groups
##           1  2  3  4
```

```
##   Female 22  9 13 15
##   Male   12  6 16 11

table(data_samples$RACE, groups)

##                                                groups
##                                                  1  2  3  4
##   Asian                                         10  1  6  2
##   Black or African American                      1  0  0  0
##   Native Hawaiian or Other Pacific Islander      1  0  0  0
##   Not Specified                                  1  1  1  1
##   Other                                          2  0  2  0
##   White                                         19 13 20 23
```