tutorial_archetypes_prototypes_SiQD_ensembles.r

michael

Sat Oct 29 21:38:31 2016

```
#R script to identify and analyse archetypes and prototypes
#in SiQD ensemble with Boltzmann distribution
#Clear any variable from the workspace
rm(list = setdiff(ls(), lsf.str()))
options(warn=-1)
#Load/install all the required libraries
for (mypkg in c("archetypes","gplots","scatterplot3d","stringr",
                "RColorBrewer","calibrate","rgl")){
  if (!is.element(mypkg, installed.packages()[,1])){
    install.packages(mypkg)
  }
  lapply(mypkg, require, character.only=TRUE)
}
```

```
## Loading required package: archetypes

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: nnls

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

## Loading required package: scatterplot3d

## Loading required package: stringr

## Loading required package: RColorBrewer

## Loading required package: calibrate

## Loading required package: MASS

## Loading required package: rgl
```

```r
#Set random seed for reproducibility
set.seed(1234)

#Function to find the closest structures to the cluster centroids
#or archetypes parameters
getSimilarToVector <- function(archetype,examples_coord,number){
  diff <- matrix(nrow=nrow(archetype),ncol=nrow(examples_coord))
  arch_close <- matrix(nrow=nrow(archetype),ncol=number)
  for (i in 1:nrow(archetype)){
    for (j in 1:nrow(examples_coord)){
      diff[i,j] <- sqrt(sum((examples_coord[j,]- archetype[i,])^2))
    }
    arch_close[i,] <- order(diff[i,])[1:number]
  }
  return(arch_close)
}

#Define plots margin parameters
par(mar =c(4, 4, 1, 1))

#Load the dataset from a comma separated text file
folder <- "/home/michael/Dropbox/Nanoinformatics_dev/Papers/silicon_aa/"
data_file <- paste(folder,"SiQDs_Boltzmann.csv",sep="")
datasource <- read.delim(data_file, header = TRUE, sep = ",")

#Define coloring scheme for plots from a column in the dataset,
#ie. nanoparticule shape
color_ID <- which(names(datasource)=="shape.index..for.convenience.")

#Define the column features to be used in the analysis
features <- seq(4,11)

#Define the structures to use in the analysis (all in this case)
data_select_ID <-seq(nrow(datasource))

#Select the data to run the analysis
data <- datasource[data_select_ID, features]

#Scale the data to zero mean and unit standard deviation
data_scaled <- scale(data)

#Perform PCA analysis
pca <- prcomp(data,scale = T)

##Compute the explained variance for different number of principal components
pc <- cumsum(pca$sdev^2 / sum(pca$sdev^2))

#Search archetypes ranging from 1 to 15 archetypes
result_archetype <- stepArchetypes(data_scaled, k=1:15, nrep=5)
```

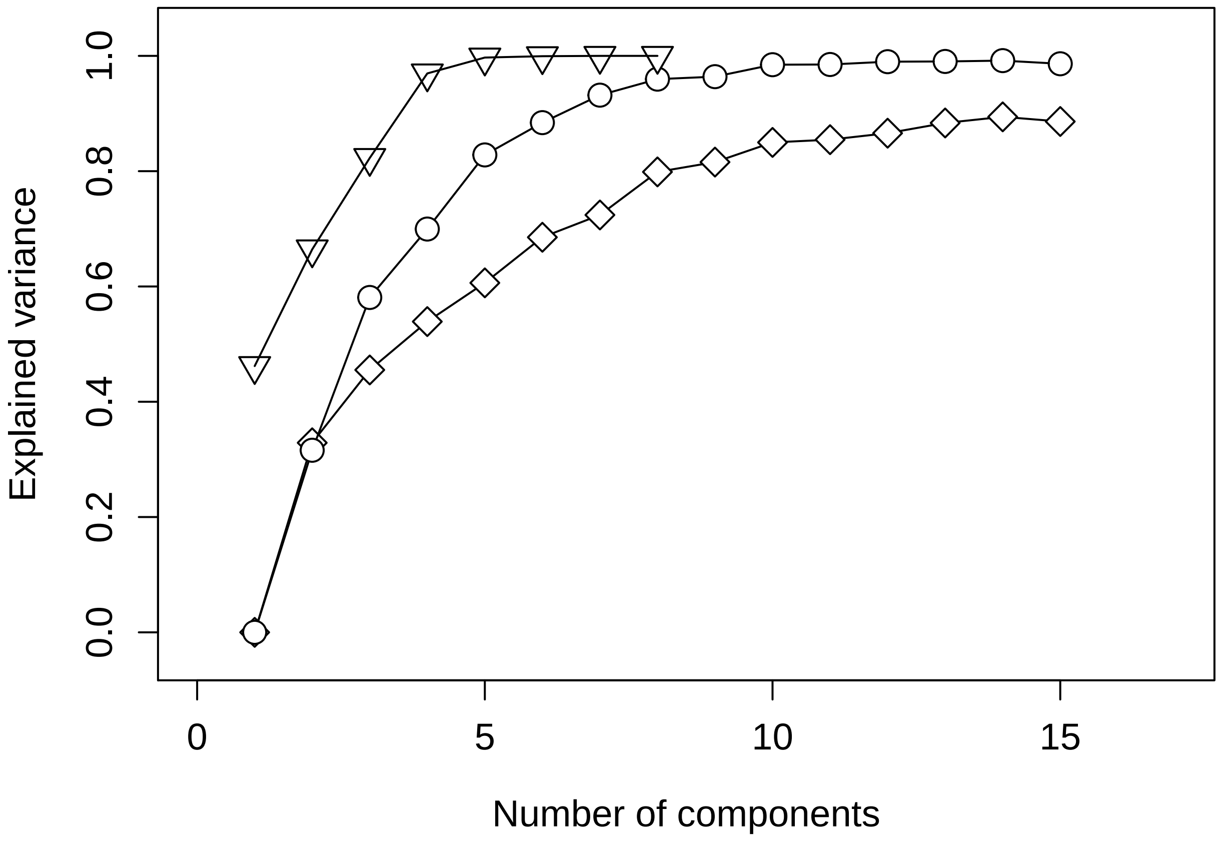
```
##
## *** k=1, rep=1:
##
```

```
## *** k=1, rep=2:
##
## *** k=1, rep=3:
##
## *** k=1, rep=4:
##
## *** k=1, rep=5:
##
## *** k=2, rep=1:
##
## *** k=2, rep=2:
##
## *** k=2, rep=3:
##
## *** k=2, rep=4:
##
## *** k=2, rep=5:
##
## *** k=3, rep=1:
##
## *** k=3, rep=2:
##
## *** k=3, rep=3:
##
## *** k=3, rep=4:
##
## *** k=3, rep=5:
##
## *** k=4, rep=1:
##
## *** k=4, rep=2:
##
## *** k=4, rep=3:
##
## *** k=4, rep=4:
##
## *** k=4, rep=5:
##
## *** k=5, rep=1:
##
## *** k=5, rep=2:
##
## *** k=5, rep=3:
##
## *** k=5, rep=4:
##
## *** k=5, rep=5:
##
## *** k=6, rep=1:
##
## *** k=6, rep=2:
##
## *** k=6, rep=3:
##
```

```
## *** k=6, rep=4:
##
## *** k=6, rep=5:
##
## *** k=7, rep=1:
##
## *** k=7, rep=2:
##
## *** k=7, rep=3:
##
## *** k=7, rep=4:
##
## *** k=7, rep=5:
##
## *** k=8, rep=1:
##
## *** k=8, rep=2:
##
## *** k=8, rep=3:
##
## *** k=8, rep=4:
##
## *** k=8, rep=5:
##
## *** k=9, rep=1:
##
## *** k=9, rep=2:
##
## *** k=9, rep=3:
##
## *** k=9, rep=4:
##
## *** k=9, rep=5:
##
## *** k=10, rep=1:
##
## *** k=10, rep=2:
##
## *** k=10, rep=3:
##
## *** k=10, rep=4:
##
## *** k=10, rep=5:
##
## *** k=11, rep=1:
##
## *** k=11, rep=2:
##
## *** k=11, rep=3:
##
## *** k=11, rep=4:
##
## *** k=11, rep=5:
##
```

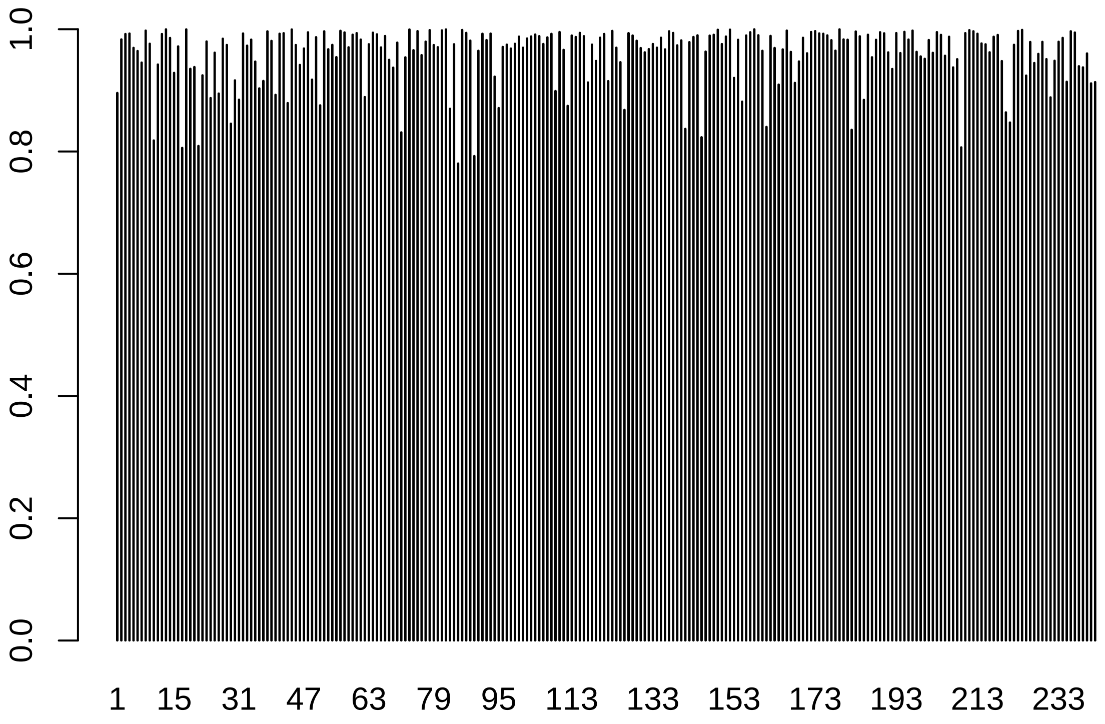
```
## *** k=12, rep=1:
##
## *** k=12, rep=2:
##
## *** k=12, rep=3:
##
## *** k=12, rep=4:
##
## *** k=12, rep=5:
##
## *** k=13, rep=1:
##
## *** k=13, rep=2:
##
## *** k=13, rep=3:
##
## *** k=13, rep=4:
##
## *** k=13, rep=5:
##
## *** k=14, rep=1:
##
## *** k=14, rep=2:
##
## *** k=14, rep=3:
##
## *** k=14, rep=4:
##
## *** k=14, rep=5:
##
## *** k=15, rep=1:
##
## *** k=15, rep=2:
##
## *** k=15, rep=3:
##
## *** k=15, rep=4:
##
## *** k=15, rep=5:
```

```r
#Select the optimum number of principal components, cluster and archetypes
#Compute the explained variance for different number of archetypes
EV = array()
for (i in 1:15){
  aa_temp <-bestModel(result_archetype[[i]])
  XCs <- aa_temp$alphas %*% aa_temp$archetypes
  EV[i] = mean((rowSums(data_scaled**2) - rowSums((data_scaled-XCs)**2))/
                 rowSums(data_scaled**2))
}

#Compute the explained variance for different number of clusters
clusters_ss <- array()
for (i in 1:15){
  clusters_tmp <- kmeans(data_scaled,i)
  clusters_ss[i] <- clusters_tmp$betweenss/clusters_tmp$totss
}

#Plot the amount of explain variance vs. number of components
plot(clusters_ss,ylim=c(-0.040,1.04),xlim=c(0,17),cex = 0.8,cex.axis = 1.2,
     fg = "black", col = "white", xlab="Number of components",
     ylab="Explained variance", cex.lab = 1.2,pin=c(7,5))
lines(clusters_ss,col="black")
points(clusters_ss,pch = 23,cex = 1.6 , fg = "black",bg = "white")
lines(EV,col="black")
points(EV,pch=21,cex = 1.6 , fg = "black",bg = "white")
points(pc,pch=25,cex = 1.6 , fg = "black",bg = "white")
lines(pc,col="black")
```

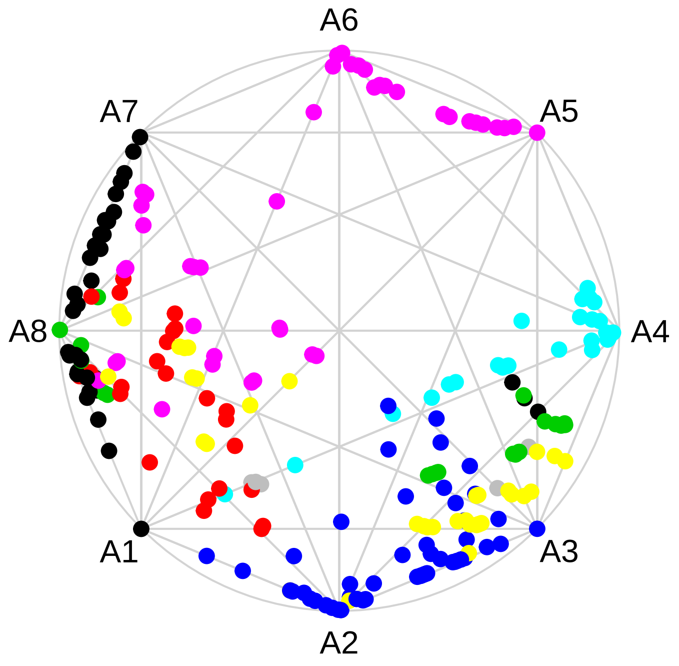


```
#Set the optimum number of clusters and archetypes according to the
#elbow criteria in the plots
number_cluster <- 8
number_archetype <- 8

#Compute the archetype model
aa <- bestModel(result_archetype[[number_archetype]])

#Plot the dataset as simplex plots of the archetypes
pointcolor <- datasource[,color_ID] #Coloring the points according a column value

#Compute the sample explained variance
```

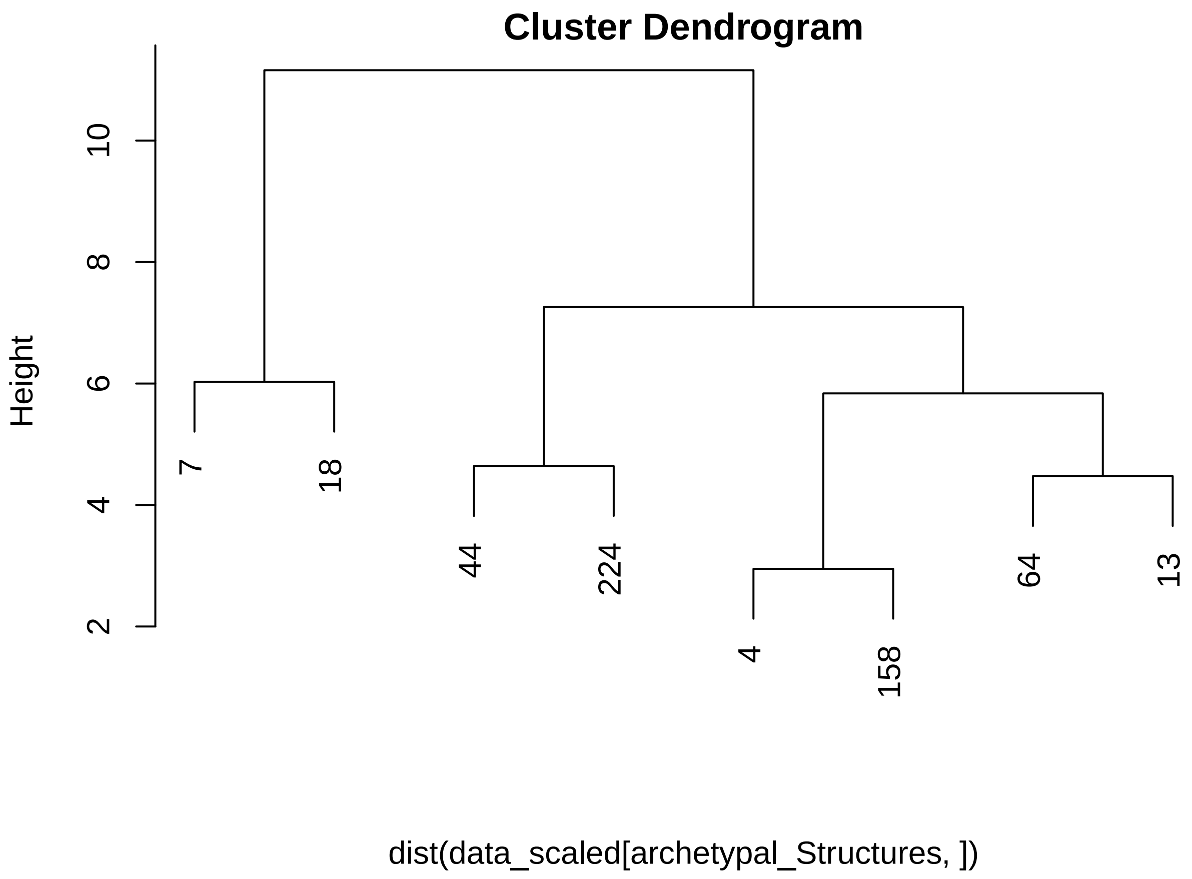
```
XCs <- aa$alphas %*% aa$archetypes
ESV = (rowSums(data_scaled**2) - rowSums((data_scaled-XCs)**2))/rowSums(data_scaled**2)
barplot(ESV,ylim=c(0,1.1),space=10)
```



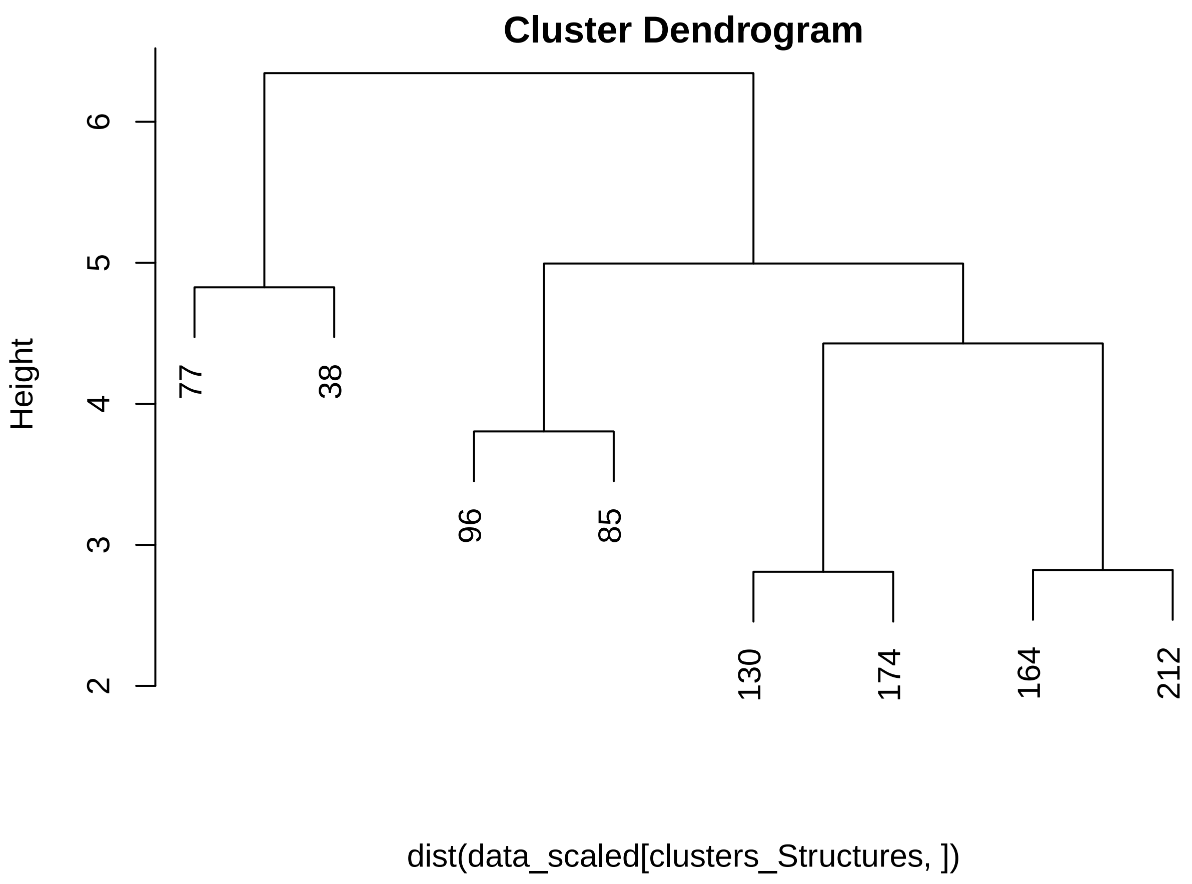
```
#Compute the projection of the data in a simpleplot
simplexplot(aa,points_col=pointcolor)
```



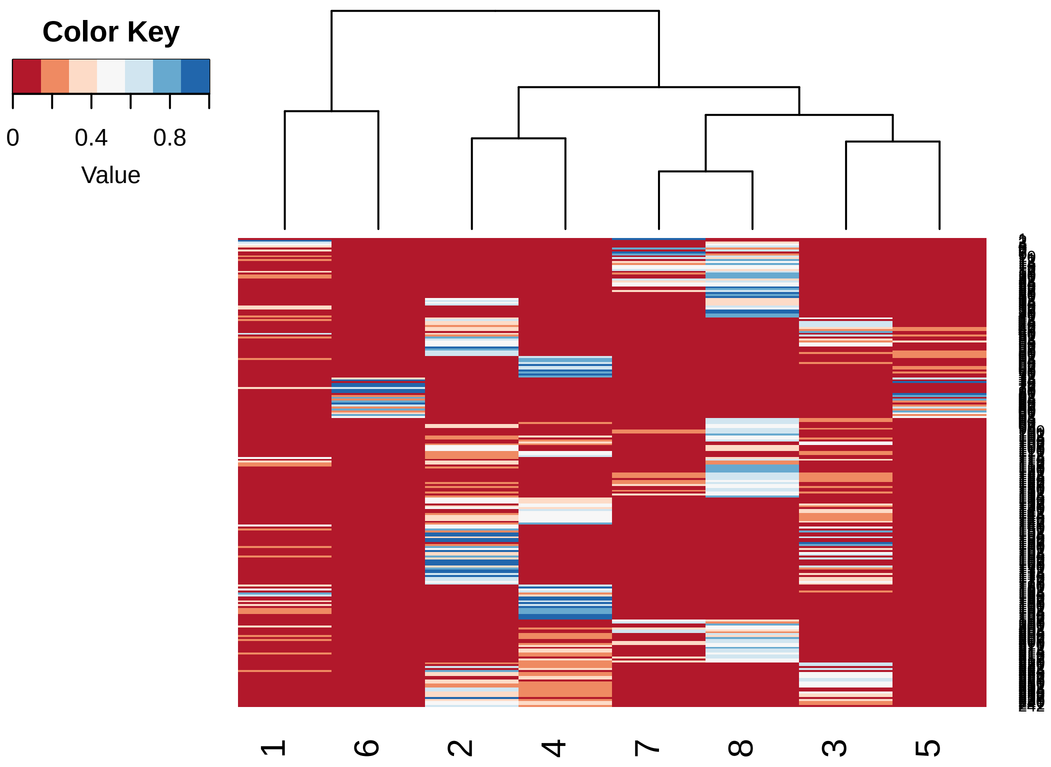
```r
sp <- simplex_projection(aa$archetypes)

#Find the closest structure to the simplex nodes (archetypes)
archetypal_Structures <- getSimilarToVector(sp,aa$alphas%*%sp,1)

#Compute the clusters
clusters <- kmeans(data_scaled,number_cluster)

#Find the closest structure to the cluster centroids (prototypes)
clusters_Structures <- getSimilarToVector(clusters$centers,data_scaled,1)

#Build a dendrogram to compare the archetypes
hc_structure <- hclust(dist(data_scaled[archetypal_Structures,]))
plot(hc_structure)
```



```r
#Build a dendrogram to compare the prototypes
hc_structure_clusters <- hclust(dist(data_scaled[clusters_Structures,]))
plot(hc_structure_clusters)
```



```
#Build a heapmat of the crontribution of the achetypes to each sample
#Define plots margin parameters.
#In case of "Error in plot.new() : figure margins too large" resize the plot windows
#in rstudio to be the full left panel.
par(mar =c(1, 1, 1, 1))
heatmap.2(aa$alphas[order(datasource[data_select_ID,color_ID]),],trace = "none",
          Rowv = F,density.info="none",dendrogram = "column",
          Colv=as.dendrogram(hclust(dist(data_scaled[archetypal_Structures,]))),
          col=brewer.pal(7,"RdBu"))
```



```
#Build 3D scatter plots of the Principal Component Analysis (PCA)
#Archetype structures are bigger spheres and prototypes are shadowed spheres
library(rgl)
rgl.open()
rgl.bg( sphere = FALSE, fogtype = "none", color=c("white","black"), back="lines")
for (i in 1:nrow(pca$x)) {rgl.spheres(pca$x[i,1],pca$x[i,2],pca$x[i,3],0.1,
                                     color=c(as.numeric(datasource[data_select_ID[i],
                                                                   color_ID])))}
rgl.spheres(pca$x[archetypal_Structures,1],pca$x[archetypal_Structures,2],
            pca$x[archetypal_Structures,3],0.5,
            color=c(as.numeric(datasource[data_select_ID[archetypal_Structures],
                                                                      color_ID])))
rgl.spheres(pca$x[clusters_Structures,1],pca$x[clusters_Structures,2],
            pca$x[clusters_Structures,3],0.5,
            color=c(as.numeric(datasource[data_select_ID[clusters_Structures],
                                                    color_ID])),alpha=0.5)
rgl.lines(c(min(pca$x[,1])-1,max(pca$x[,1])+1),c(max(pca$x[,2])+1,max(pca$x[,2])+1),
          c(min(pca$x[,3])-1,min(pca$x[,3])-1),color=c("black"))
rgl.lines(c(min(pca$x[,1])-1,min(pca$x[,1])-1),c(min(pca$x[,2])-1,max(pca$x[,2])+1),
          c(min(pca$x[,3])-1,min(pca$x[,3])-1),color=c("black"))
rgl.lines(c(min(pca$x[,1])-1,min(pca$x[,1])-1),c(max(pca$x[,2])+1,max(pca$x[,2])+1),
          c(min(pca$x[,3])-1,max(pca$x[,3])+1),color=c("black"))
rgl.viewpoint(0,-90,zoom=0.7)

rgl.texts(max(pca$x[,1])+1, max(pca$x[,2])+1, min(pca$x[,3])-2, c("PC1"),cex=1.5,
          color=c("black"),family=c("sans"))
rgl.texts(min(pca$x[,1])-1, min(pca$x[,2])+1, min(pca$x[,3])-2, c("PC2"),cex=1.5,
          color=c("black"),family=c("sans"))
rgl.texts(min(pca$x[,1])-1, max(pca$x[,2])+1, max(pca$x[,3])+2, c("PC3"),cex=1.5,
          color=c("black"),family=c("sans"))

#Build simplex plots of the archetypes model
#Archetype structures are bigger spheres and prototypes are shadowed spheres
angle = -(90/180)*3.14
M <- matrix(c(cos(angle),-sin(angle),sin(angle),cos(angle)),2,2)
rgl.open()
rgl.bg(sphere = FALSE, fogtype = "none", color=c("white","black"), back="lines")
for (i in 1:length(archetypal_Structures)-1){
  rgl.lines(c((sp %*% M)[i,1],(sp %*% M)[i+1,1]),c((sp %*% M)[i,2],
                                                   (sp %*% M)[i+1,2]),
            c(0,0),color=c("black")) }
rgl.lines(c((sp %*% M)[length(archetypal_Structures),1],(sp %*% M)[1,1]),
          c((sp %*% M)[length(archetypal_Structures),2],(sp %*% M)[1,2]),
          c(0,0),color=c("black"))
rgl.spheres((aa$alphas%*%sp%*% M)[,1], (aa$alphas%*%sp%*% M)[,2],r = 0.3,
            rep(0,nrow(data)),color = as.numeric(datasource[data_select_ID,color_ID]))
rgl.spheres((aa$alphas%*%sp%*% M)[archetypal_Structures,1],
            (aa$alphas%*%sp%*% M)[archetypal_Structures,2],r = 0.7,rep(0,nrow(data)),
            color = as.numeric(datasource[data_select_ID,color_ID][archetypal_Structures]))
rgl.spheres((aa$alphas%*%sp%*% M)[clusters_Structures,1],
            (aa$alphas%*%sp%*% M)[clusters_Structures,2],r = 0.7,alpha=0.3,
            rep(0,nrow(data)),
            color = as.numeric(datasource[data_select_ID,color_ID][archetypal_Structures]))
rgl.spheres(seq(19,19 + length(unique(datasource[data_select_ID,color_ID]))*(-0.),-0.),
            seq(1,1 + (length(unique(datasource[data_select_ID,color_ID]))-1)*0.6,0.6),
            rep(0,length(unique(datasource[data_select_ID,color_ID]))) ,r=0.3,
            color=c(as.numeric(unique(datasource[data_select_ID,color_ID])) ))
rgl.texts(seq(22.5,22.5 + length(unique(datasource[data_select_ID,color_ID]))*(-0.),-0.),
          seq(1.05,1.05 + (length(unique(datasource[data_select_ID,color_ID]))-1)*0.6,0.6),
          rep(0,length(unique(datasource[data_select_ID,color_ID])))  ,
          unique(datasource[data_select_ID,color_ID]),cex=1.5,color=c("black"),family=c("sans"))
rgl.viewpoint(0,0,zoom=0.7)
```