Task

1. Introduction

For task 3, we are going to deal with a dataset containing the shopping behavior of 487 different customers. We have 11 statements here and the corresponding variables measuring to what extent they agree with these statements, namely: 1) organising_trip: It is very important for me to organize the shopping well. 2) knowing_buy: When I leave to go shopping, I know exactly what I am going to buy. 3) duty_responsability: By doing the shopping, I fulfill my duty and take my responsibility. 4) shopping_fun: I enjoy doing the shopping. 5) take_at_ease: I do the shopping at a leisurely pace. 6) enjoy: I enjoy the atmosphere while shopping. 7) shopping_drag: Doing the shopping is a drag. 8) minimise_shoppingtime: I try to keep the time that I spend doing the shopping to a minimum. 9) shopping_list: I usually take a list with me when I go to do the shopping. 10) shopping_with_family: I like shopping with the whole family. 11) have_stock: I like having a stock of products on hand at home.

The variables vary from 1 to 7, where 1 means totally disagree and 7 means totally agree. Our task is to cluster the customers basing on the 11 items of shopping mentioned above and try to interpret the results we will get. What's more, we are also going to test the stability of the cluster solutions deduced by different methods.

2. Methodology

We will try to do the clustering with (1) hierarchical clustering with Ward's method on squared Euclidean distances followed by k-means with the centroid of the hierarchical clustering as starting point, (2) model-based clustering with hddc(), and (3) model-based clustering using Mclust(). In order to test the stability of our cluster solutions, we are going to split our dataset randomly into a train set and a test set. The cluster methods will be firstly applied on the train set, and then on test set. After that, we will also form a cluster solution on the test set by assigning the points to the cluster centroid of the training sample that is closest. Finally, by comparing our two cluster solutions on the test set, we can evaluate the stability of our methods.

3. Result

Firstly, in order to make our cluster solution more reasonable, we are going to standardize our data first.

```
## standardize variables
shopping<-scale(shopping,center=TRUE,scale=TRUE)
```

Then, we form our train set and test set randomly for the validation work.

```
#create train and test set
set.seed(0829539)
sel<-sample(1:487,size=250,replace=FALSE)
train<-shopping[sel,]
valid<-shopping[-sel,]
```

Before applying our cluster methods on our dataset, we first try to do an exploratory factor analysis to summarize our 11 attributes, as we are going to make use of it for the interpretation of our cluster solutions. We start with a principal component analysis.

```
#compute variance accounted for by each component
kbl(round(prcomp.out$sd^2/sum(prcomp.out$sd^2),3))
```

x
0.341
0.207
0.090
0.079
0.070
0.056
0.045
0.036
0.034
0.026
0.016

As is shown here, only the first two components accounts for more than 10% of the variances, and they account for 54.8% of the variances in total. We then continue to conduct the exploratory factor analysis with 2 factors.

According to the result shown above, we can define two factors here. Enjoy: The extent to which someone enjoys shopping, summarizing the attributes "shopping_fun", "take_at_ease", "enjoy", "shopping_drag", "minimise_shoppingtime" and "shopping_with_family". Here we can see that the loadings for "shopping_drag" and "minimise_shoppingtime" is negative, and the others are positive, which is quite reasonable. Organized: The extent to which someone is organized when doing shopping, summarizing the attributes "organising_trip", "knowing_buy", "duty_responsability", "shopping_list" and "have_stock". So, we can use these two factors to classify our customers as "enjoy shopping and organized", "enjoy shopping and not organized", "dislike shopping and organized", or "dislike shopping and not organized". We denote the principal components as comp for future use, as we are going to visualize our cluster solution in the space of the first two principal components.

```
comp<-as.matrix(shopping)%*%prcomp.out$rotation
```

We also define the "clusters" function here for classifying new points to the nearest cluster centroid. We are going to use this function for the validation of our cluster methods.

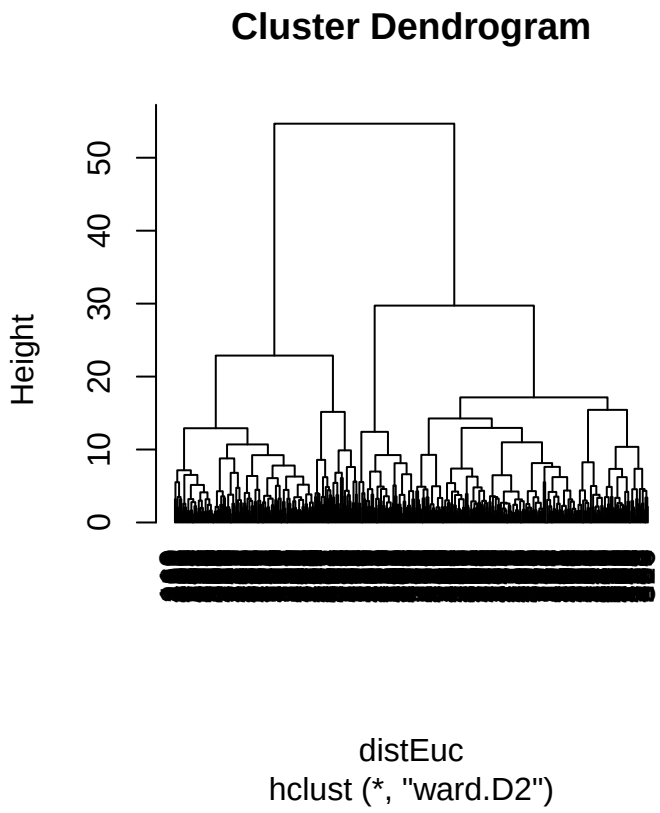
```
# function classify new points to nearest cluster centroid
clusters <- function(x, centers) {
  # compute squared euclidean distance from each sample to each cluster center
  tmp <- sapply(seq_len(nrow(x)),
                function(i) apply(centers, 1,
                                  function(v) sum((x[i, ]-v)^2)))
  max.col(-t(tmp))  # find index of min distance
```

Now we are going to apply our three different cluster methods on our dataset separately, and then interpret the results. For each method, we are going to compute the solution with 1 up to 6 solutions and choose the most reasonable solution.

I) Hierarchical clustering with Ward's method on squared Euclidean distances followed by k-means with the centroid of the hierarchical clustering as starting point

Firstly, we apply Hierarchical clustering with Ward's method on squared Euclidean distances and draw the dendrogram to observe the structure.

```
# compute Euclidean distances
distEuc<-dist(shopping, method = "euclidean", diag = FALSE, upper = FALSE)

##  hierarchical clustering method of Ward on squared Euclidean distance
hiclust_ward<- hclust(distEuc, "ward.D2")

## plot dendrogram
par(pty="s")
plot(hiclust_ward,hang=-1)
```



We can see from the dendrogram that cutting the tree at the height around 20, which means selecting 4 clusters, is quite reasonable. Then we continue with k-means method using the centroid of the hierarchical clustering as starting point, and compute solutions with 1 up to 6 clusters.

```
# classification of students for solutions with 1-6 clusters
clustvar<-cutree(hiclust_ward, k=1:6)
```

a) 1 cluster

```
# k-means with 1 clusters using centroid of Ward as starting point
nclust<-1
stat<-describeBy(shopping,clustvar[,nclust],mat=TRUE)
hcenter<-matrix(stat[,5],nrow=nclust)
rownames(hcenter)<-paste("c_",rep(1:nclust),sep="")
colnames(hcenter)<-c(colnames(shopping))
kmean1<-kmeans(shopping,centers=hcenter,iter.max=200)

round(kmean1$centers,2)
```

```
  organising_trip knowing_buy duty_responsability shopping_fun take_at_ease
1               0           0                   0            0            0
  enjoy shopping_drag minimise_shoppingtime shopping_list shopping_with_family
1     0             0                    0             0                    0
  have_stock
1          0
```

b) 2 cluster

```
# k-means with 2 clusters using centroid of Ward as starting point
nclust<-2
stat<-describeBy(shopping,clustvar[,nclust],mat=TRUE)
hcenter<-matrix(stat[,5],nrow=nclust)
rownames(hcenter)<-paste("c_",rep(1:nclust),sep="")
colnames(hcenter)<-c(colnames(shopping))
kmean2<-kmeans(shopping,centers=hcenter,iter.max=200)

round(kmean2$centers,2)
```

```
  organising_trip knowing_buy duty_responsability shopping_fun take_at_ease
1           -0.07       -0.05                0.06         0.65         0.56
2            0.11        0.08               -0.10        -1.04        -0.90
  enjoy shopping_drag minimise_shoppingtime shopping_list shopping_with_family
1  0.61         -0.66                -0.54         -0.04                 0.28
2 -0.97          1.06                 0.87          0.07                -0.44
  have_stock
1       0.04
2      -0.06
```

c) 3 cluster

```
# k-means with 3 clusters using centroid of Ward as starting point
nclust<-3
stat<-describeBy(shopping,clustvar[,nclust],mat=TRUE)
hcenter<-matrix(stat[,5],nrow=nclust)
rownames(hcenter)<-paste("c_",rep(1:nclust),sep="")
```

```
colnames(hcenter)<-c(colnames(shopping))
kmean3<-kmeans(shopping,centers=hcenter,iter.max=200)

round(kmean3$centers,2)
```

```
  organising_trip knowing_buy duty_responsability shopping_fun take_at_ease
1            0.36        0.40                0.28         0.69         0.56
2           -1.29       -1.26               -0.53         0.45         0.55
3            0.19        0.12               -0.08        -1.05        -0.94
  enjoy shopping_drag minimise_shoppingtime shopping_list shopping_with_family
1  0.68         -0.65                -0.43          0.33                 0.41
2  0.30         -0.60                -0.73         -1.08                -0.09
3 -0.97          1.07                 0.87          0.11                -0.45
  have_stock
1       0.17
2      -0.33
3      -0.05
```

d) 4 cluster

```
# k-means with 4 clusters using centroid of Ward as starting point
nclust<-4
stat<-describeBy(shopping,clustvar[,nclust],mat=TRUE)
hcenter<-matrix(stat[,5],nrow=nclust)
rownames(hcenter)<-paste("c_",rep(1:nclust),sep="")
colnames(hcenter)<-c(colnames(shopping))
kmean4<-kmeans(shopping,centers=hcenter,iter.max=200)

round(kmean4$centers,2)
```

```
  organising_trip knowing_buy duty_responsability shopping_fun take_at_ease
1            0.37        0.42                0.29         0.69         0.57
2           -1.16       -1.20               -0.51         0.55         0.52
3            0.45        0.36                0.04        -1.03        -0.96
4           -1.00       -0.84               -0.56        -1.07        -0.67
  enjoy shopping_drag minimise_shoppingtime shopping_list shopping_with_family
1  0.69         -0.65                -0.41          0.34                 0.44
2  0.39         -0.69                -0.87         -1.00                -0.14
3 -0.97          1.07                 0.87          0.44                -0.40
4 -0.98          1.02                 0.86         -1.15                -0.57
  have_stock
1       0.18
2      -0.31
3       0.24
4      -1.05
```

e) 5 cluster

```
# k-means with 5 clusters using centroid of Ward as starting point
nclust<-5
stat<-describeBy(shopping,clustvar[,nclust],mat=TRUE)
hcenter<-matrix(stat[,5],nrow=nclust)
rownames(hcenter)<-paste("c_",rep(1:nclust),sep="")
colnames(hcenter)<-c(colnames(shopping))
kmean5<-kmeans(shopping,centers=hcenter,iter.max=200)

round(kmean5$centers,2)
```

```
  organising_trip knowing_buy duty_responsability shopping_fun take_at_ease
1            0.52        0.45                0.80         0.86         0.66
2            0.20        0.26               -0.30         0.51         0.44
3           -1.33       -1.25               -0.46         0.55         0.59
4            0.45        0.36                0.04        -1.03        -0.96
5           -1.00       -0.84               -0.56        -1.07        -0.67
  enjoy shopping_drag minimise_shoppingtime shopping_list shopping_with_family
1  0.87         -0.65                -0.45          0.28                 0.99
2  0.44         -0.63                -0.47          0.41                -0.15
3  0.48         -0.73                -0.79         -1.20                -0.13
4 -0.97          1.07                 0.87          0.44                -0.40
5 -0.98          1.02                 0.86         -1.15                -0.57
  have_stock
1       0.47
2      -0.18
3      -0.25
4       0.24
5      -1.05
```

f) 6 cluster

```
# k-means with 6 clusters using centroid of Ward as starting point
nclust<-6
stat<-describeBy(shopping,clustvar[,nclust],mat=TRUE)
hcenter<-matrix(stat[,5],nrow=nclust)
rownames(hcenter)<-paste("c_",rep(1:nclust),sep="")
colnames(hcenter)<-c(colnames(shopping))
kmean6<-kmeans(shopping,centers=hcenter,iter.max=200)

round(kmean6$centers,2)
```

```
  organising_trip knowing_buy duty_responsability shopping_fun take_at_ease
1            0.54        0.41                0.83         0.83         0.64
2            0.27        0.33                0.68         0.82         0.83
3            0.08        0.17               -0.49         0.49         0.33
4           -1.49       -1.40               -0.73         0.47         0.52
5            0.46        0.36                0.03        -1.04        -0.97
6           -1.00       -0.84               -0.56        -1.07        -0.67
  enjoy shopping_drag minimise_shoppingtime shopping_list shopping_with_family
1  0.88         -0.59                -0.42          0.82                 0.87
2  0.81         -0.81                -0.44         -1.17                 0.44
```

```
3  0.37         -0.61                -0.53          0.61                -0.02
4  0.37         -0.67                -0.81         -1.15                -0.22
5 -0.97          1.08                 0.87          0.45                -0.42
6 -0.98          1.02                 0.86         -1.15                -0.57
  have_stock
1       0.42
2       0.12
3      -0.06
4      -0.40
5       0.23
6      -1.05
```

We also list the sizes and the proportions of explained variance for all the solutions here.

```
##number of observations per cluster
kmean1$size
```

```
[1] 487
```

```
kmean2$size
```

```
[1] 300 187
```

```
kmean3$size
```

```
[1] 219  87 181
```

```
kmean4$size
```

```
[1] 214  86 143  44
```

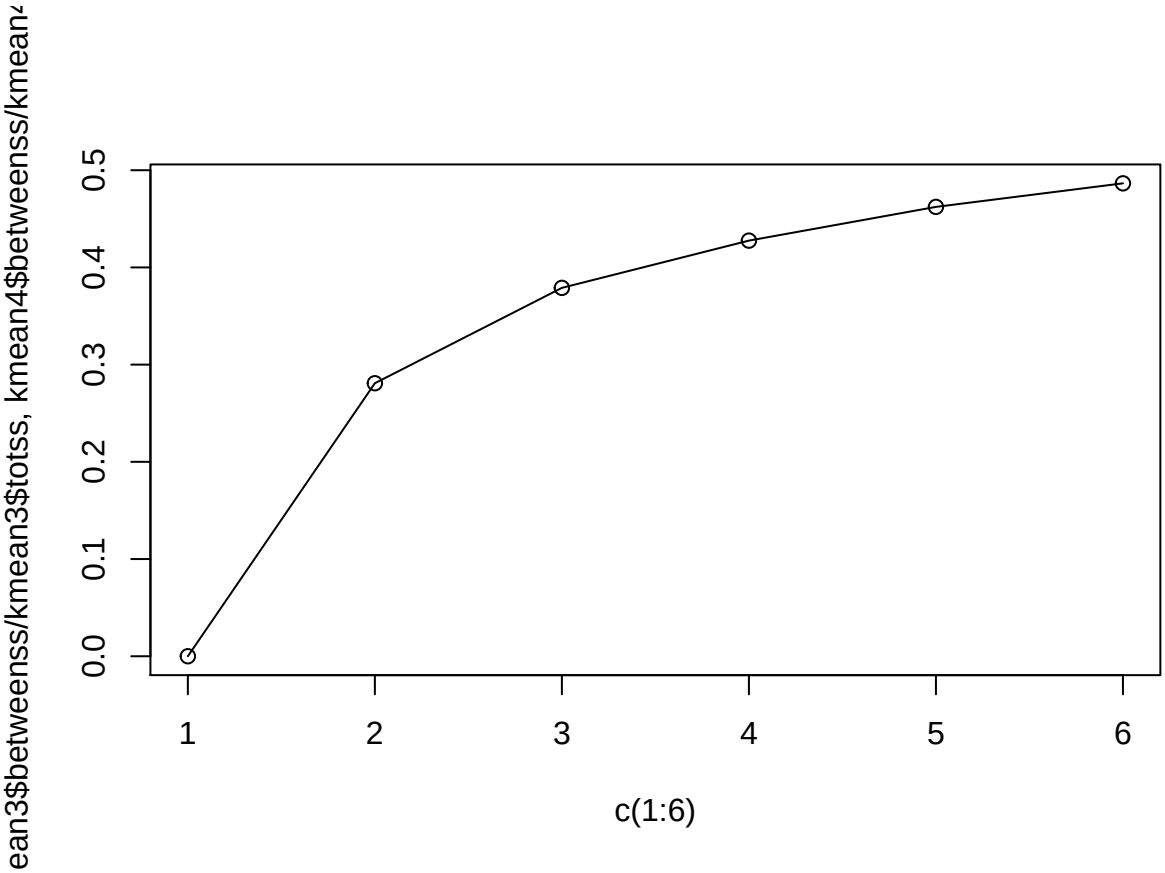
```
kmean5$size
```

```
[1] 110 115  75 143  44
```

```
kmean6$size
```

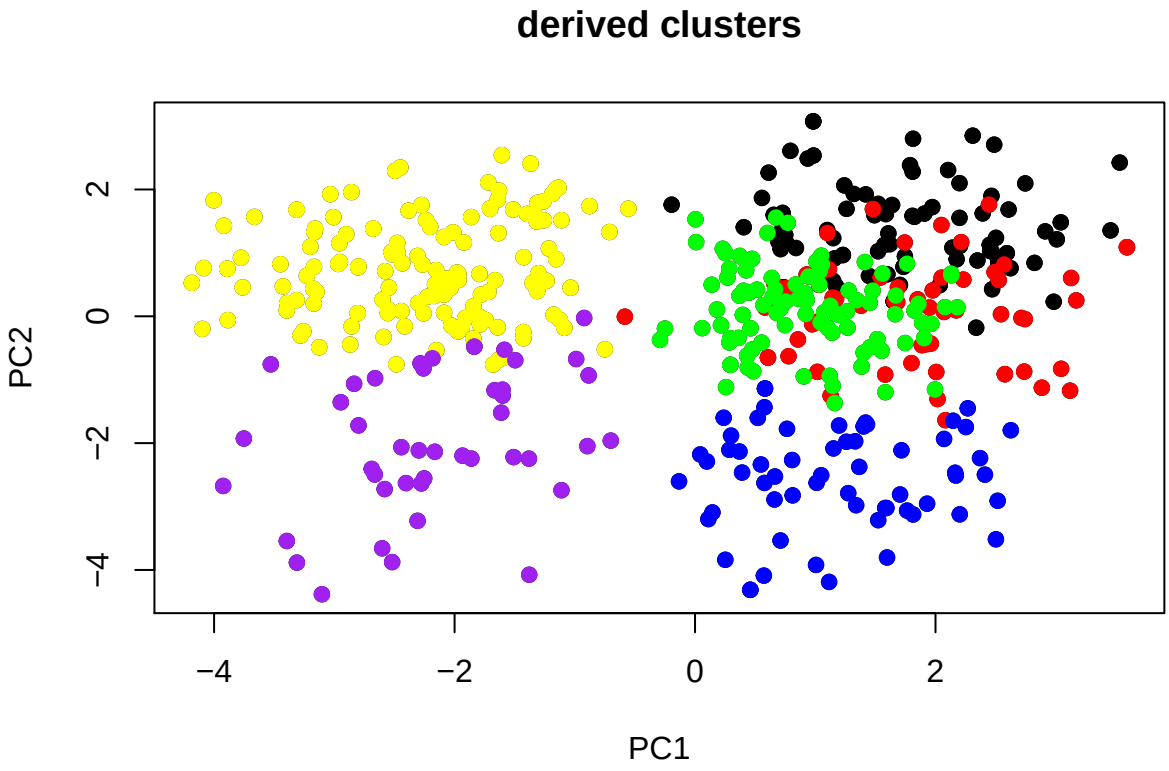
```
[1]  84  61  95  61 142  44
```

```
## proportion of explained variance
plot(c(1:6),c(kmean1$betweenss/kmean1$totss,kmean2$betweenss/kmean2$totss,
              kmean3$betweenss/kmean3$totss,kmean4$betweenss/kmean4$totss,
              kmean5$betweenss/kmean5$totss,kmean6$betweenss/kmean6$totss))
lines(c(1:6),c(kmean1$betweenss/kmean1$totss,kmean2$betweenss/kmean2$totss,
               kmean3$betweenss/kmean3$totss,kmean4$betweenss/kmean4$totss,
               kmean5$betweenss/kmean5$totss,kmean6$betweenss/kmean6$totss))
```



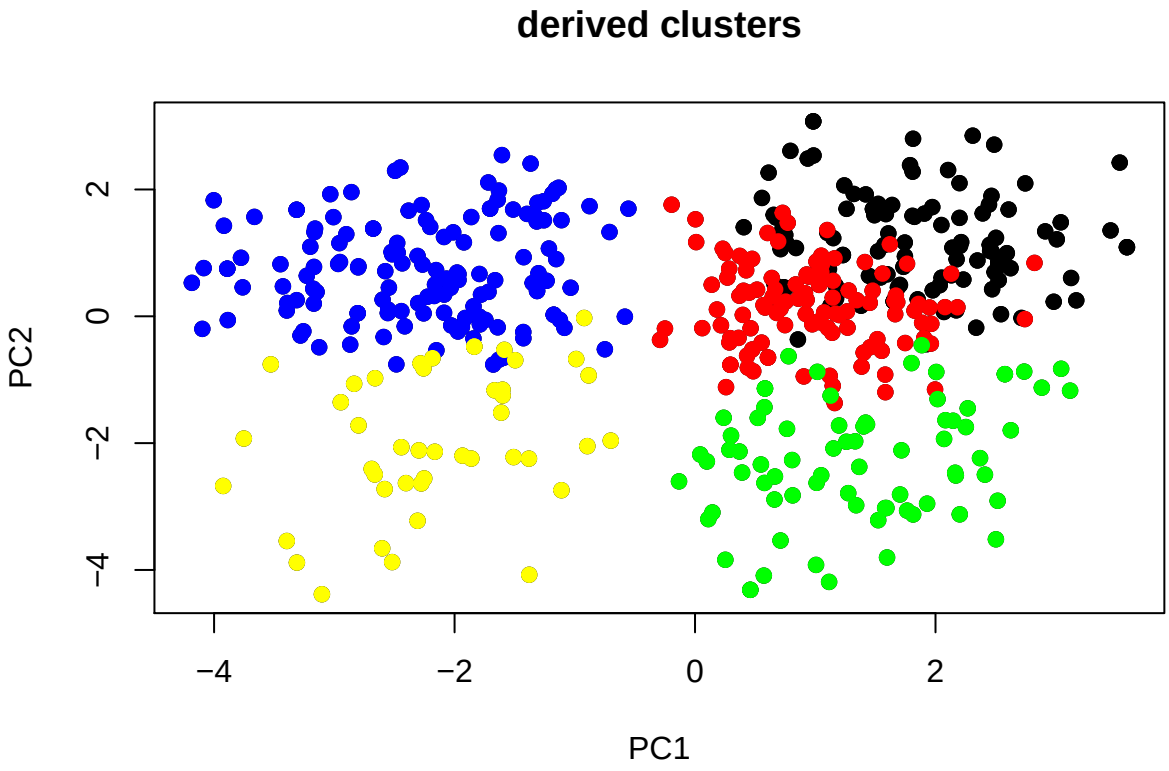
We can see here that the more clusters we select, the larger the proportion of explained variance is. However, the proportion grows lower and lower while we increase the number of clusters. When we take 6 clusters, the size of each cluster seems relatively small, but not too small, so we still decide to take 6 clusters now, in order to get maximum explained variance. Then we visualize our cluster solution in the space of the first two principal components.

```
plot(comp,main="derived clusters")
points(comp[kmean6$cluster==1,],col="black",pch=19)
points(comp[kmean6$cluster==2,],col="red",pch=19)
points(comp[kmean6$cluster==3,],col="green",pch=19)
points(comp[kmean6$cluster==4,],col="blue",pch=19)
points(comp[kmean6$cluster==5,],col="yellow",pch=19)
points(comp[kmean6$cluster==6,],col="purple",pch=19)
```



We can see that the green part and the red part seem to be totally mixed up with each other, while our principal component 1, the "enjoy" factor, still separates the clusters well into positive ones and negative ones, namely, those who enjoy shopping and those who dislike shopping. As the result corresponding to our principal component 2, the "organized" factor, seems really too hard to interpret, we decide to try to use 5 clusters, as this won't make a great decrease in the explained variance. We visualize our solution with 5 clusters as below:

```
#plot clusters extracted with HDDC in space of first two principal components
plot(comp,main="derived clusters")
points(comp[kmean5$cluster==1,],col="black",pch=19)
points(comp[kmean5$cluster==2,],col="red",pch=19)
points(comp[kmean5$cluster==3,],col="green",pch=19)
points(comp[kmean5$cluster==4,],col="blue",pch=19)
points(comp[kmean5$cluster==5,],col="yellow",pch=19)
```



Here the borders become much clearer. We can interpret the cluster solutions as: The blue part: Positive for the "organized" factor but negative for the "enjoy" factor. The customers who dislike shopping but are organized. The yellow part: Negative for the "organized" factor and negative for the "enjoy" factor. The customers who dislike shopping and are not organized. The green part: Negative for the "organized" factor but positive for the "enjoy" factor. The customers who enjoy shopping but are not organized. The red part: Around 0 for the "organized" factor and positive for the "enjoy" factor. The customers who enjoy shopping and are moderately organized. The black part: Positive for the "organized" factor and positive for the "enjoy" factor. The customers who enjoy shopping and are organized. This interpretation seems to make sense, so we continue with the validation work. We first conduct the cluster method on our train dataset to get several clusters, and the cluster our validation dataset with the same method. After that, we cluster our validation data set again according to the clusters we've got from our train dataset. By comparing the difference between the two results we get from our validation dataset, we can measure the stability of our cluster method. The measurement of difference here is always ARI index, which has zero expected value in the case of a random partition and is bounded above by 1 in the case of perfect agreement between two partitions.

```
# cluster train data Ward + kmeans
disttrain<-dist(train, method = "euclidean", diag = FALSE, upper = FALSE)
wardtrain<- hclust(disttrain, "ward.D2")
nclust<-5
clustvar<-cutree(wardtrain, k=nclust)
stat<-describeBy(train,clustvar,mat=TRUE)
hcenter<-matrix(stat[,5],nrow=nclust)
rownames(hcenter)<-paste("c_",rep(1:nclust),sep="")
colnames(hcenter)<-c(colnames(train))
kmeantrain<-kmeans(train,centers=hcenter,iter.max=200)
```

```
  # cluster validation data Ward + kmeans
  distvalid<-dist(valid, method = "euclidean", diag = FALSE, upper = FALSE)
  wardvalid<- hclust(distvalid, "ward.D2")
  nclust<-5
  clustvar<-cutree(wardvalid, k=nclust)
  stat<-describeBy(valid,clustvar,mat=TRUE)
  hcenter<-matrix(stat[,5],nrow=nclust)
  rownames(hcenter)<-paste("c_",rep(1:nclust),sep="")
  colnames(hcenter)<-c(colnames(valid))
  kmeanvalid<-kmeans(valid,centers=hcenter,iter.max=200)


  classif1<-clusters(valid, kmeantrain[["centers"]])
  classif2<-kmeanvalid$cluster
  table(classif1,classif2)
```

```
        classif2
classif1  1  2  3  4  5
       1 12  0  1 43  0
       2  2 30  0  0  1
       3  0  0  9  0 20
       4  0  0 64  0  0
       5 40  1  0 14  0
```

```
  cl1<-as.factor(classif1)
  cl2<-factor(classif2,levels=c("4","2","5","3","1"))
  mat<-table(cl1,cl2)
  print(mat)
```

```
   cl2
cl1  4  2  5  3  1
  1 43  0  0  1 12
  2  0 30  1  0  2
  3  0  0 20  9  0
  4  0  0  0 64  0
  5 14  1  0  0 40
```

```
  percent5=sum(diag(mat))/sum(mat)
  round(percent5,2)
```

```
[1] 0.83
```

```
  rand5<-adjustedRandIndex(cl1,cl2)
  round(rand5,2)
```

```
[1] 0.65
```

We get an 83% here for the match rate, and 0.65 here for ARI index, which is acceptable. Further, we test the stability for this method with 1 up to 6 clusters, and obtain the curve for match rate as well as ARI index below:

```
#match rate ARI Index
```

We can see that both the match rate and the ARI index, which also reveal the stability of the method, decrease monotonously as the number of clusters increases, and a rapid decrease occurs at 5. So, in order to increase the stability of our method, we turn to select 4 clusters as our final decision, and this still won't cause great loss in explained variance. The visualized result for 4 clusters case is as below:

```
#4 clusters
```

We have 4 very clear clusters here: The green part: Positive for the "organized" factor but negative for the "enjoy" factor. The customers who dislike shopping but are organized. The blue part: Negative for the "organized" factor and negative for the "enjoy" factor. The customers who dislike shopping and are not organized. The red part: Negative for the "organized" factor but positive for the "enjoy" factor. The customers who enjoy shopping but are not organized. The black part: Positive for the "organized" factor and positive for the "enjoy" factor. The customers who enjoy shopping and are organized.

This is also consistent with our initial guess from the dendrogram.

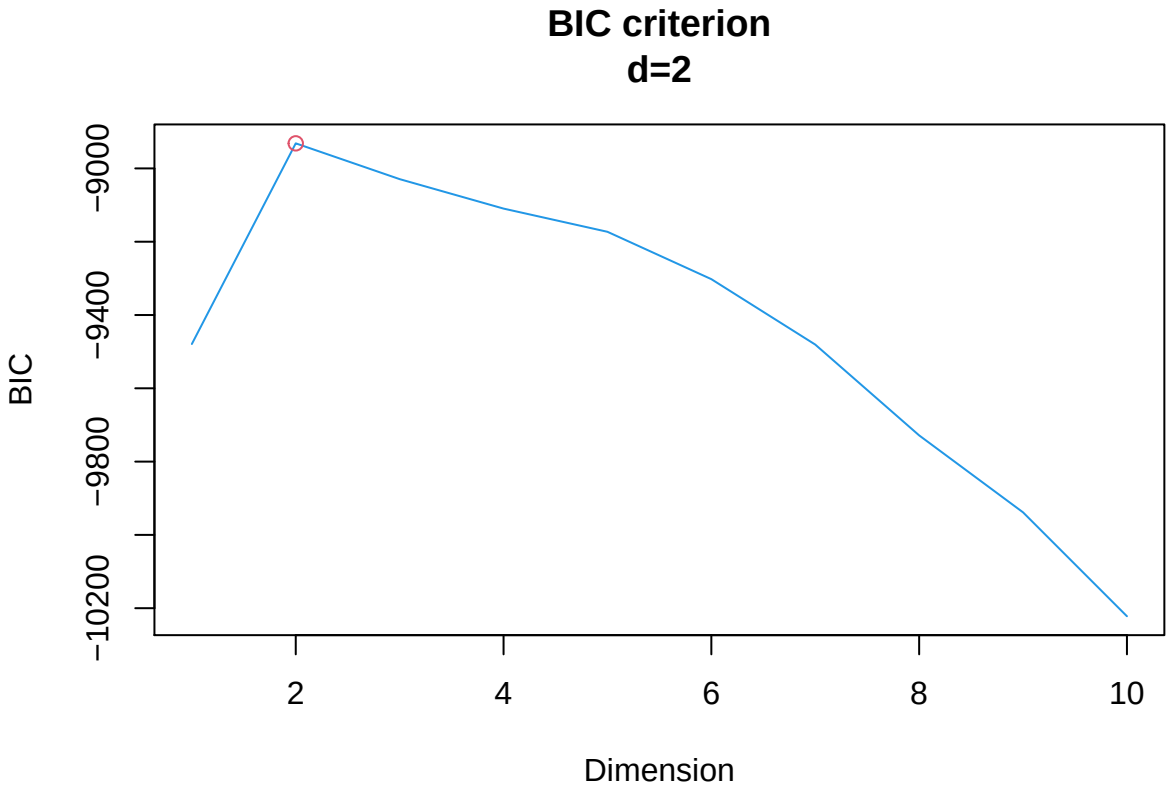
II) Model-based clustering with hddc()

Firstly, we let the hddc() function itself to select the optimal number of clusters.

```
#number of clusters chosen by hddc
#use BIC to select the number of dimensions
set.seed(0829539)
hddc1.out<-hddc(shopping,K=1:6,model="all")
hddc1.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: ABQD
  Posterior probabilities of groups
       1      2     3     4     5
   0.155 0.0637 0.194 0.315 0.272
      Intrinsic dimensions of the classes:
       1 2 3 4 5
  dim: 2 2 2 2 2

A: 1.09

B: 0.434
BIC:  -13426.7
```

```
plot(hddc1.out)
```



We can see that the hddc() function has chosen 5 clusters, and the optimal dimension to get the highest BIC value here is 2, which is consistent with our previous results from principal component analysis. Then we continue to get the solutions with 1 up to 6 clusters.

Then we continue to get the solutions with 1 up to 6 clusters.

a) 1 cluster

```
#fit all models with K=1
set.seed(0829539)
hddc1.out<-hddc(shopping,K=1,model="all")
hddc1.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: AKJBKQKDK
  Posterior probabilities of groups
   1
   1
      Intrinsic dimensions of the classes:
       1
  dim: 2

Class   a1   a2
    1 3.75 2.27
        1
```

```
Bk: 0.551
BIC:  -13842.64
```

b) 2 clusters

```
#fit all models with K=2
set.seed(0829539)
hddc2.out<-hddc(shopping,K=2,model="all")
hddc2.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: ABKQKD
  Posterior probabilities of groups
       1     2
   0.297 0.703
      Intrinsic dimensions of the classes:
       1 2
  dim: 2 2

A: 2.69
        1     2
Bk: 0.626 0.398
BIC:  -13630.49
```

c) 3 clusters

```
#fit all models with K=3
set.seed(0829539)
hddc3.out<-hddc(shopping,K=3,model="all")
hddc3.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: AKBKQKD
  Posterior probabilities of groups
       1     2     3
   0.159 0.449 0.392
      Intrinsic dimensions of the classes:
       1 2 3
  dim: 2 2 2
       1    2    3
Ak: 1.71 1.17 1.74
        1     2     3
Bk: 0.456 0.375 0.518
BIC:  -13521.93
```

d) 4 clusters

```
#fit all models with K=4
set.seed(0829539)
hddc4.out<-hddc(shopping,K=4,model="all")
hddc4.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: AJBQD
  Posterior probabilities of groups
      1      2     3     4
   0.463 0.0641 0.156 0.317
      Intrinsic dimensions of the classes:
       1 2 3 4
  dim: 2 2 2 2
      a1   a2
Aj: 1.26 0.98

B: 0.458
BIC:  -13429.53
```

e) 5 clusters

```
#fit all models with K=5
set.seed(0829539)
hddc5.out<-hddc(shopping,K=5,model="all")
hddc5.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: ABQD
  Posterior probabilities of groups
      1      2     3     4     5
   0.155 0.0639 0.244 0.222 0.315
      Intrinsic dimensions of the classes:
       1 2 3 4 5
  dim: 2 2 2 2 2

A: 1.09

B: 0.435
BIC:  -13426.96
```

f) 6 clusters

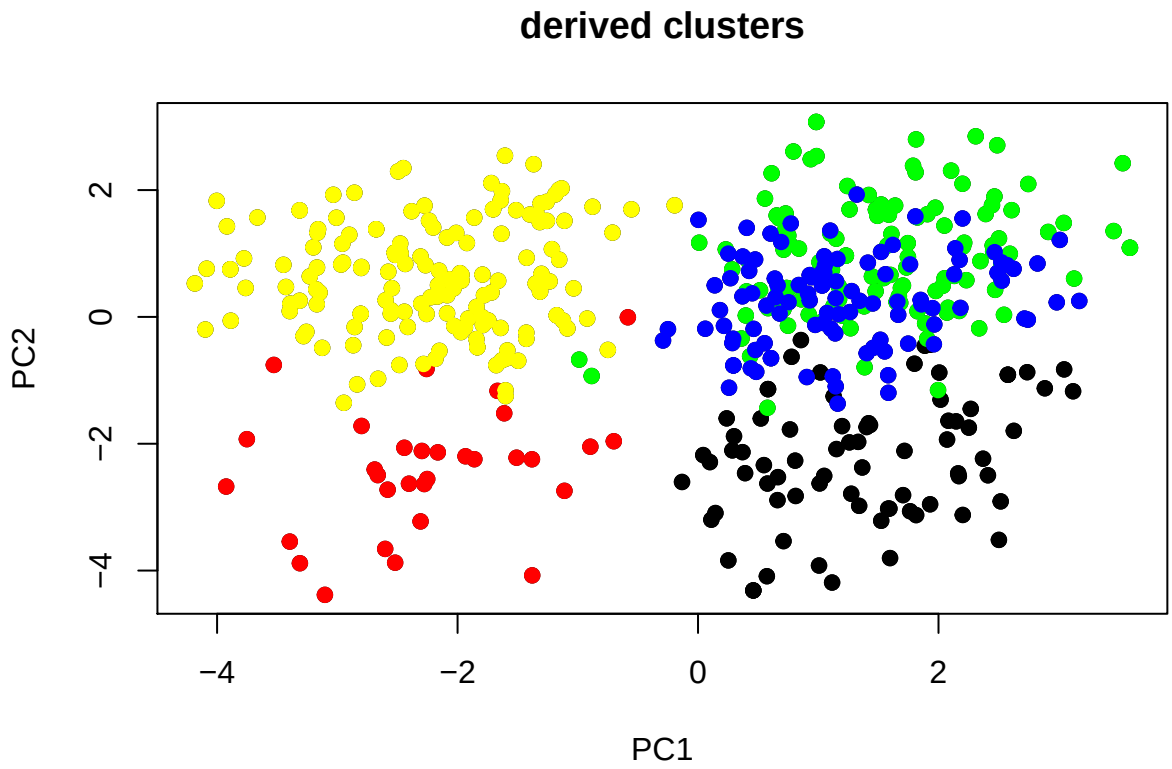
```
set.seed(0829539)
hddc6.out<-hddc(shopping,K=6,model="all")
hddc6.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: AJBQD
  Posterior probabilities of groups
```

```
       1     2     3    4     5     6
  0.0516 0.146 0.058 0.14 0.167 0.437
     Intrinsic dimensions of the classes:
      1 2 3 4 5 6
 dim: 2 2 2 2 2 2
     a1    a2
Aj: 1.22 0.875

B: 0.425
BIC:  -13395.17
```

Reasonably, we believe in the hddc() function and choose 5 clusters here as our initial solution. We visualize it in the space of the first two principal components as below:

```
#plot clusters extracted with HDDC in space of first two principal components
plot(comp,main="derived clusters")
points(comp[hddc5.out$class==1,],col="black",pch=19)
points(comp[hddc5.out$class==2,],col="red",pch=19)
points(comp[hddc5.out$class==3,],col="green",pch=19)
points(comp[hddc5.out$class==4,],col="blue",pch=19)
points(comp[hddc5.out$class==5,],col="yellow",pch=19)
```



Hopefully, we can interpret in the same way as we have done for the kmeans method with 5 clusters, but here the green part and the blue part are almost overlapped, which is not very good for the interpretation. We hold on and test the stability of hddc() method with 5 clusters.

```r
# cluster train data hddc
set.seed(0829539)
hddcT.out<-hddc(train,K=5,model="all")
hddcT.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: ABQD
  Posterior probabilities of groups
       1     2      3     4     5
   0.0723 0.454 0.0836 0.287 0.104
      Intrinsic dimensions of the classes:
       1 2 3 4 5
  dim: 2 2 2 2 2

A: 1.06

B: 0.447
BIC:  -7088.109
```

```r
# cluster validation data hddc
set.seed(0829539)
hddcV.out<-hddc(valid,K=5,model="all")
hddcV.out
```

```
HIGH DIMENSIONAL DATA CLUSTERING
MODEL: ABQD
  Posterior probabilities of groups
       1     2     3     4     5
   0.309 0.092 0.134 0.277 0.188
      Intrinsic dimensions of the classes:
       1 2 3 4 5
  dim: 2 2 2 2 2

A: 0.989

B: 0.447
BIC:  -6742.602
```

```r
classif1<-clusters(valid, hddcT.out[["mu"]])
classif2<-hddcV.out$class
table(classif1,classif2)
```

```
        classif2
classif1  1  2  3  4  5
       1  7 18  0  0  0
       2  0  0  1 66 41
       3  0  2 15  0  2
       4 66  1  0  0  0
       5  0  1 16  1  0
```

```
cl1<-as.factor(classif1)
cl2<-factor(classif2,levels=c("2","4","5","1","3"))
mat<-table(cl1,cl2)
#print(mat)
kbl(mat)
```

2	4	5	1	3
18	0	0	7	0
0	66	41	0	1
2	0	2	0	15
1	0	0	66	0
1	1	0	0	16

```
percent5=sum(diag(mat))/sum(mat)
round(percent5,2)
```

```
[1] 0.71
```

```
rand5<-adjustedRandIndex(cl1,cl2)
round(rand5,2)
```

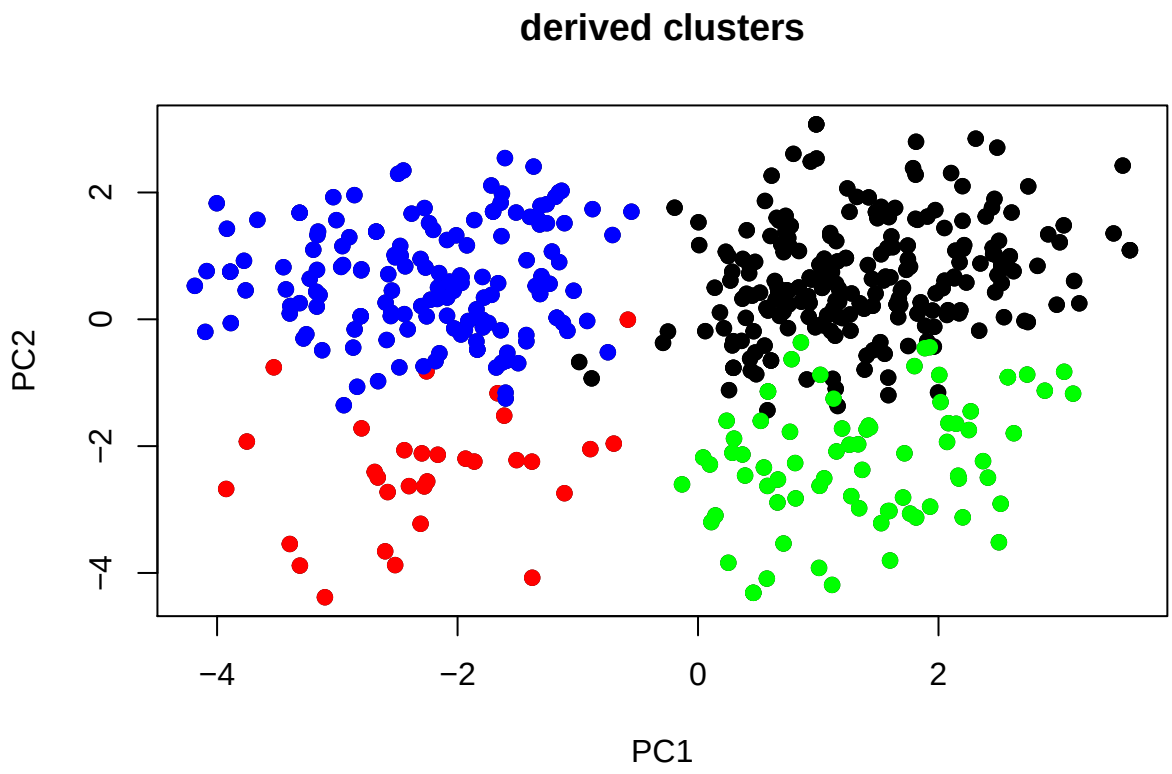
```
[1] 0.63
```

The match rate here is 71% and the ARI index here is 0.63, which are not quite good but still acceptable. However, if we continue to test the stability for this method with 1 up to 6 clusters, and obtain the curve for match rate as well as ARI index, we can see that:

```
#ari match
```

A sharp decrease occurs at 5, for both match rate and ARI index, and at 4 we get the maximal value for both match rate and ARI index, thus the most stable. We take a look at the solution with 4 clusters to check if we can get a result easy to interpret.

```
#plot clusters extracted with HDDC in space of first two principal components
plot(comp,main="derived clusters")
points(comp[hddc4.out$class==1,],col="black",pch=19)
points(comp[hddc4.out$class==2,],col="red",pch=19)
points(comp[hddc4.out$class==3,],col="green",pch=19)
points(comp[hddc4.out$class==4,],col="blue",pch=19)
```



The borders are clear now and we can give an explicit interpretation for this figure. The customers are classified into four clusters according to our two principal components as below: The blue part: Positive for the "organized" factor but negative for the "enjoy" factor. The customers who dislike shopping but are organized. The red part: Negative for the "organized" factor and negative for the "enjoy" factor. The customers who dislike shopping and are not organized. The green part: Negative for the "organized" factor but positive for the "enjoy" factor. The customers who enjoy shopping but are not organized. The black part: Positive for the "organized" factor and positive for the "enjoy" factor. The customers who enjoy shopping and are organized. So finally, we take 4 clusters as our final decision.

III) Model-based clustering using Mclust()

Like hddc(), we firstly let the function Mclust() itself to choose an optimal number of clusters.

```
#number of clusters chosen by Mclust
set.seed(0829539)
mclust1.out<-Mclust(shopping,G=1:6)
summary(mclust1.out)
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVE (ellipsoidal, equal orientation) model with 4 components:

 log-likelihood   n  df       BIC       ICL
```

```
      -5996.532 487 146 -12896.55 -12942.22

Clustering table:
  1   2   3   4
 79 115 160 133
```

Here Mclust() has chosen 4 as the optimal number of clusters. Now we continue to obtain the solutions with 1 up to 6 clusters.

a) 1 cluster

```
#fit all models with G=1
set.seed(0829539)
mclust1.out<-Mclust(shopping,G=1)
summary(mclust1.out)
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust XXX (ellipsoidal multivariate normal) model with 1 component:

 log-likelihood   n df       BIC       ICL
      -6532.38 487 77 -13541.26 -13541.26

Clustering table:
  1
487
```

b) 2 clusters

```
#fit all models with G=2
set.seed(0829539)
mclust2.out<-Mclust(shopping,G=2)
summary(mclust2.out)
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust EVE (ellipsoidal, equal volume and orientation) model with 2 components:

 log-likelihood   n df       BIC       ICL
      -6376.903 487 99 -13366.44 -13392.97

Clustering table:
  1   2
321 166
```

c) 3 clusters

```
#fit all models with G=3
set.seed(0829539)
mclust3.out<-Mclust(shopping,G=3)
summary(mclust3.out)
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVE (ellipsoidal, equal orientation) model with 3 components:

 log-likelihood   n  df       BIC       ICL
      -6152.089 487 123 -13065.33 -13093.78

Clustering table:
  1   2   3
190 160 137
```

d) 4 clusters

```
#fit all models with G=4
set.seed(0829539)
mclust4.out<-Mclust(shopping,G=4)
summary(mclust4.out)
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVE (ellipsoidal, equal orientation) model with 4 components:

 log-likelihood   n  df       BIC       ICL
      -5996.532 487 146 -12896.55 -12942.22

Clustering table:
  1   2   3   4
 79 115 160 133
```

e) 5 clusters

```
#fit all models with G=5
set.seed(0829539)
mclust5.out<-Mclust(shopping,G=5)
summary(mclust5.out)
```

```
----------------------------------------------------
```

```
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVE (ellipsoidal, equal orientation) model with 5 components:

 log-likelihood   n  df       BIC       ICL
      -5936.425 487 169 -12918.67 -12968.95

Clustering table:
  1   2   3   4   5
 72  55 155  73 132
```

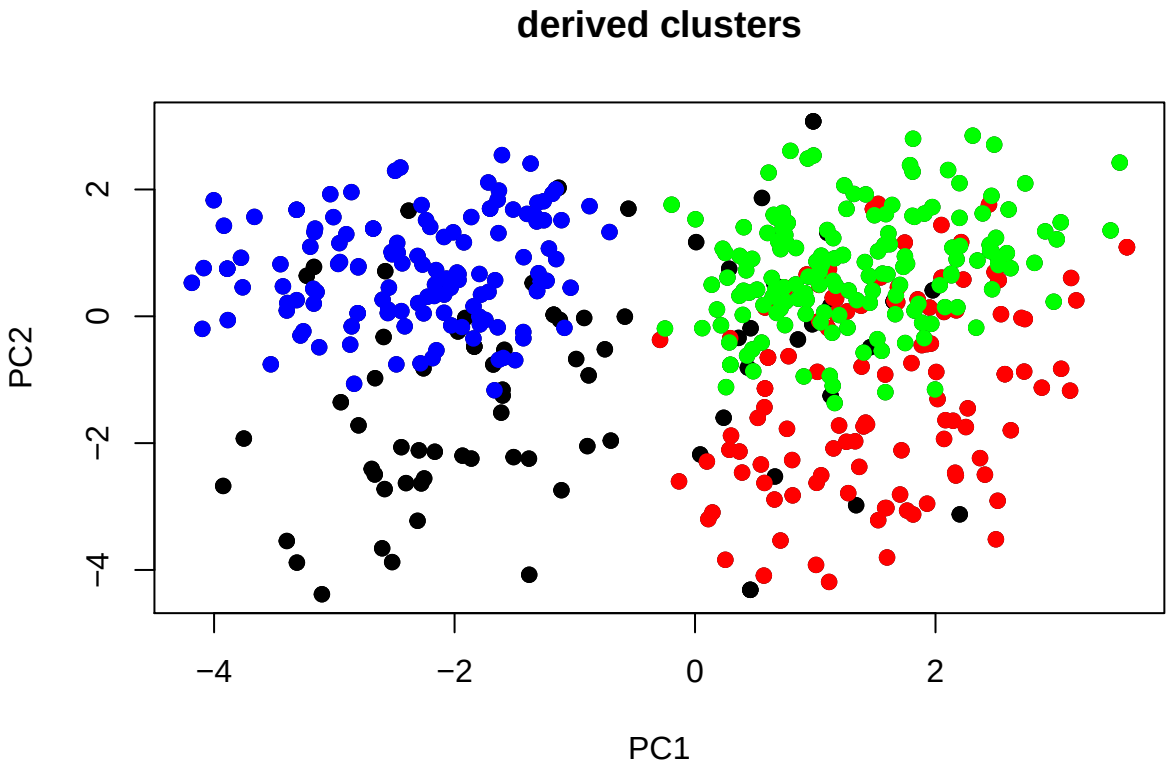
f) 6 clusters

```
#fit all models with G=6
set.seed(0829539)
mclust6.out<-Mclust(shopping,G=6)
summary(mclust6.out)
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVE (ellipsoidal, equal orientation) model with 6 components:

 log-likelihood   n  df       BIC       ICL
      -5895.384 487 192 -12978.92 -13041.74

Clustering table:
  1   2   3   4   5   6
 71  53 122  68 132  41
```

We start with believing that Mclust() function will give us the optimal number of clusters, and visualize our result for Mclust() method with 4 clusters in the space of the first two principal components as below:

```
#plot clusters extracted with HDDC in space of first two principal components
plot(comp,main="derived clusters")
points(comp[mclust4.out$class==1,],col="black",pch=19)
points(comp[mclust4.out$class==2,],col="red",pch=19)
points(comp[mclust4.out$class==3,],col="green",pch=19)
points(comp[mclust4.out$class==4,],col="blue",pch=19)
```

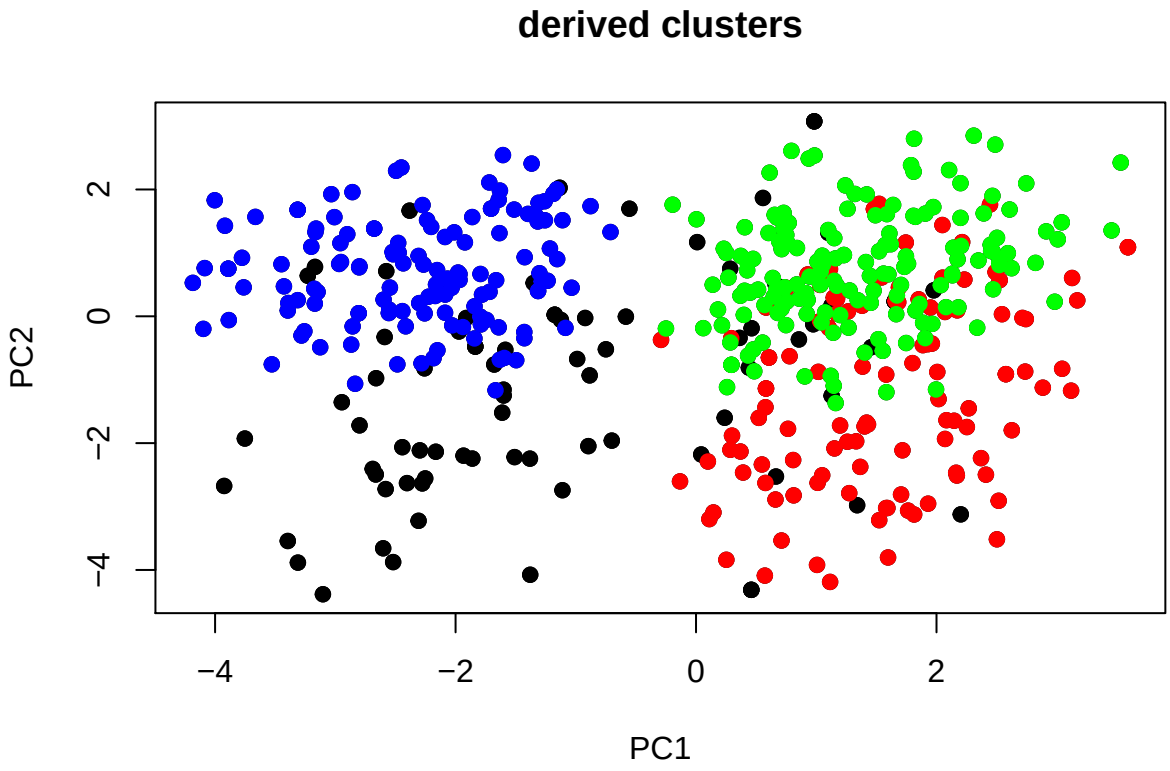


Actually, the black points here are very divergent, but its centroid is still located in the left-bottom corner. Actually, if we also take the visualized result with other numbers of clusters, we can see that:

6 clusters

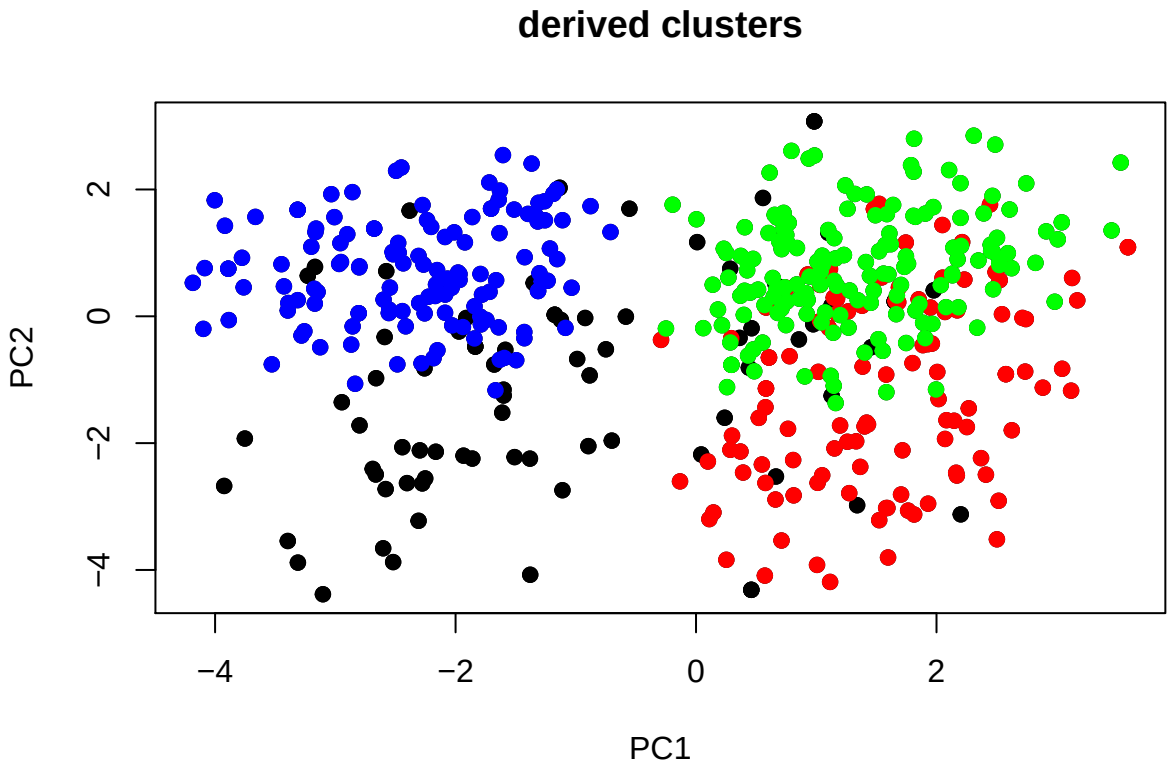
5 clusters

```
#plot clusters extracted with HDDC in space of first two principal components
plot(comp,main="derived clusters")
points(comp[mclust4.out$class==1,],col="black",pch=19)
points(comp[mclust4.out$class==2,],col="red",pch=19)
points(comp[mclust4.out$class==3,],col="green",pch=19)
points(comp[mclust4.out$class==4,],col="blue",pch=19)
```



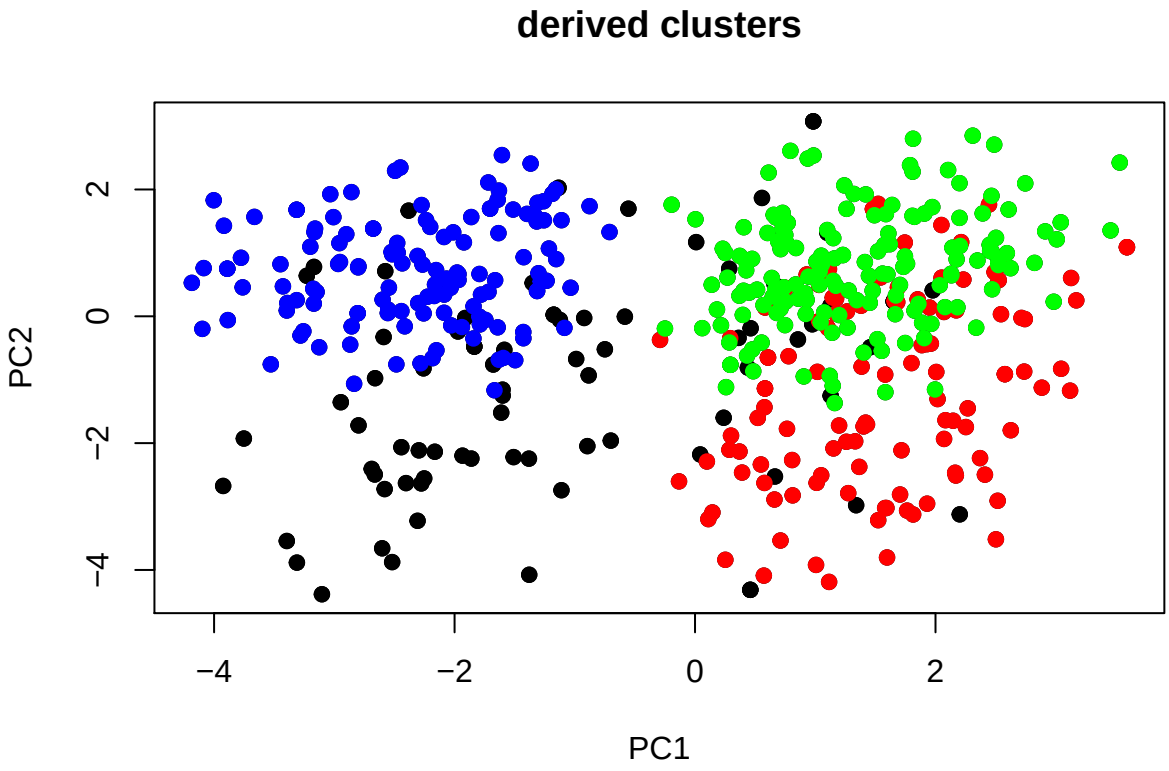
3 clusters

```
#plot clusters extracted with HDDC in space of first two principal components
plot(comp,main="derived clusters")
points(comp[mclust4.out$class==1,],col="black",pch=19)
points(comp[mclust4.out$class==2,],col="red",pch=19)
points(comp[mclust4.out$class==3,],col="green",pch=19)
points(comp[mclust4.out$class==4,],col="blue",pch=19)
```



2 clusters

```
#plot clusters extracted with HDDC in space of first two principal components
plot(comp,main="derived clusters")
points(comp[mclust4.out$class==1,],col="black",pch=19)
points(comp[mclust4.out$class==2,],col="red",pch=19)
points(comp[mclust4.out$class==3,],col="green",pch=19)
points(comp[mclust4.out$class==4,],col="blue",pch=19)
```



For the cases with 6 clusters and 5 clusters, different parts are totally mixed up with each other, which is too hard to interpret, and for the cases with 3 and even 2 clusters, also we can't avoid the black dots to be divergent. So, 4 clusters seem to be the best choice here, and we can roughly interpret the result as: The blue part: Positive for the "organized" factor but negative for the "enjoy" factor. The customers who dislike shopping but are organized.

The red part: Negative for the "organized" factor and negative for the "enjoy" factor. The customers who dislike shopping and are not organized. The green part: Negative for the "organized" factor but positive for the "enjoy" factor. The customers who enjoy shopping but are not organized. The black part: Positive for the "organized" factor and positive for the "enjoy" factor. The customers who enjoy shopping and are organized.

Then we take a look at the stability.

```
# cluster validation data Mclust
set.seed(0829539)
#mclustV.out<-Mclust(valid,G=4)
#summary(mclustV.out)


#classif1<-clusters(valid, t(as.matrix(mclustT.out$parameters$mean)))
#classif2<-mclustV.out$class
#table(classif1,classif2)

#cl1<-as.factor(classif1)
#cl2<-factor(classif2,levels=c("3","4","2","1"))
#mat<-table(cl1,cl2)
```

```
#print(mat)

#percent4=sum(diag(mat))/sum(mat)
#round(percent4,2)

#rand4<-adjustedRandIndex(cl1,cl2)
#round(rand4,2)
```

We can see that here the match rate is 73% and the ARI index is 0.66, which are both acceptable. If we again continue to test the stability for this method with 1 up to 6 clusters, and obtain the curve for match rate as well as ARI index, we can see that:

```
#plot
#plot(c(2:6), c(percent2,percent3,percent4,percent5,percent6))
#lines(c(2:6), c(percent2,percent3,percent4,percent5,percent6))

#plot
#plot(c(2:6), c(rand2,rand3,rand4,rand5,rand6))
#lines(c(2:6), c(rand2,rand3,rand4,rand5,rand6))
```

4 clusters are already the most stable case, and we take it as our final decision.

4. Conclusion

As conclusion, we end up choosing 4 clusters for all these three methods and obtained the same interpretation with our two principal components “enjoy” and “organized”. With analysis for all these three methods, we can conclude that, Mclust() method here performs better than hddc() method in stability, but the boundaries of the clusters obtained by hddc() method are clearer. The hierarchical clustering with Ward’s method on squared Euclidean distances followed by k-means with the centroid of the hierarchical clustering as starting point is the most stable method here for this task.