Group Coursework Assignment

Dave Matt

Table of Contents

Tasks:1

Read and inspect the data set. Provide a descriptive analysis for each of the variables in the data set.

- **Anwser**

```
# Load packages
library("plotly")
```

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

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

library("tidyverse")

## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks plotly::filter(), stats::filter()
## x dplyr::lag()    masks stats::lag()

library("data.table")

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

library("dplyr")
library("gridExtra")

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

library("knitr")
library("scales")

##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

# Load Cloud Data
df.office <- read_csv("office.csv")

## Rows: 200 Columns: 10

## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr (1): professional
## dbl (9): respondent_id, variety_of_choice, electronics, furniture,
quality_o...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.

# view first 6 rows of cloud.csv
head(df.office)

## # A tibble: 6 x 10
##   respondent_id variety_of_choice electronics furniture quality_of_service
##           <dbl>             <dbl>       <dbl>     <dbl>               <dbl>
## 1             1                 8           3         6                   3
## 2             2                 6           3         1                   4
## 3             3                 6           1         2                   4
## 4             4                 8           3         3                   4
## 5             5                 4           6         3                   4
## 6             6                 8           4         3                   5
## # ... with 5 more variables: low_prices <dbl>, return_policy <dbl>,
## #   professional <chr>, income <dbl>, age <dbl>

# reveal cloud.csv data analysis
str(df.office)

## spec_tbl_df [200 x 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ respondent_id     : num [1:200] 1 2 3 4 5 6 7 8 9 10 ...
##  $ variety_of_choice : num [1:200] 8 6 6 8 4 8 7 7 10 8 ...
##  $ electronics       : num [1:200] 3 3 1 3 6 4 2 5 7 4 ...
##  $ furniture         : num [1:200] 6 1 2 3 3 3 2 3 5 0 ...
##  $ quality_of_service: num [1:200] 3 4 4 4 4 5 2 2 1 4 ...
##  $ low_prices        : num [1:200] 2 7 9 8 2 10 8 2 5 9 ...
##  $ return_policy     : num [1:200] 2 1 6 7 5 6 7 3 4 1 ...
##  $ professional      : chr [1:200] "non-professional" "non-professional"
"non-professional" "non-professional" ...
```

```
##  $ income            : num [1:200] 16 22 18 18 35 13 22 19 14 16 ...
##  $ age               : num [1:200] 28 27 22 29 51 24 27 26 27 28 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   respondent_id = col_double(),
##   ..   variety_of_choice = col_double(),
##   ..   electronics = col_double(),
##   ..   furniture = col_double(),
##   ..   quality_of_service = col_double(),
##   ..   low_prices = col_double(),
##   ..   return_policy = col_double(),
##   ..   professional = col_character(),
##   ..   income = col_double(),
##   ..   age = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# descriptive analysis of cloud.csv
summary(df.office)
```

```
##  respondent_id    variety_of_choice  electronics      furniture
##  Min.   :  1.00   Min.   : 4.000    Min.   : 1.00   Min.   :0.00
##  1st Qu.: 50.75   1st Qu.: 6.000    1st Qu.: 3.00   1st Qu.:1.00
##  Median :100.50   Median : 8.000    Median : 4.50   Median :2.00
##  Mean   :100.50   Mean   : 7.565    Mean   : 4.45   Mean   :3.27
##  3rd Qu.:150.25   3rd Qu.:10.000    3rd Qu.: 6.00   3rd Qu.:6.00
##  Max.   :200.00   Max.   :10.000    Max.   :10.00   Max.   :7.00
##  quality_of_service   low_prices     return_policy   professional
##  Min.   :1.00       Min.   : 1.000   Min.   : 1.00   Length:200
##  1st Qu.:2.00       1st Qu.: 2.000   1st Qu.: 3.00   Class :character
##  Median :3.00       Median : 5.000   Median : 4.00   Mode  :character
##  Mean   :3.53       Mean   : 4.795   Mean   : 4.25
##  3rd Qu.:4.00       3rd Qu.: 7.000   3rd Qu.: 6.00
##  Max.   :9.00       Max.   :10.000   Max.   :10.00
##      income           age
##  Min.   :13.00   Min.   :21.00
##  1st Qu.:15.00   1st Qu.:24.00
##  Median :19.50   Median :27.00
##  Mean   :32.17   Mean   :32.52
##  3rd Qu.:54.25   3rd Qu.:38.00
##  Max.   :95.00   Max.   :68.00
```

Tasks:2

Make a new data object (e.g., a data.frame or tibble) for clustering that includes only the attitudinal variables from the original data set. Then normalise (use z-score standardisation) all variables in this new data object. Which variable has the smallest minimum value and which variable has the largest maximum value in the normalized data set?

- **Anwser**

```
# make a new data frame with only attitudinal varibales
df.officen <- df.office[,2:7]
head(df.officen)
```

```
## # A tibble: 6 x 6
##   variety_of_choice electronics furniture quality_of_service low_prices
##               <dbl>       <dbl>     <dbl>              <dbl>      <dbl>
## 1                 8           3         6                  3          2
## 2                 6           3         1                  4          7
## 3                 6           1         2                  4          9
## 4                 8           3         3                  4          8
## 5                 4           6         3                  4          2
## 6                 8           4         3                  5         10
## # ... with 1 more variable: return_policy <dbl>
```

```
# Then normalise (use z-score standardisation)
df.officen<-scale(df.officen, center = TRUE, scale = FALSE)
summary(df.officen)
```

```
##  variety_of_choice  electronics      furniture     quality_of_service
##  Min.   :-3.565    Min.   :-3.45   Min.   :-3.27   Min.   :-2.53
##  1st Qu.:-1.565    1st Qu.:-1.45   1st Qu.:-2.27   1st Qu.:-1.53
##  Median : 0.435    Median : 0.05   Median :-1.27   Median :-0.53
##  Mean   : 0.000    Mean   : 0.00   Mean   : 0.00   Mean   : 0.00
##  3rd Qu.: 2.435    3rd Qu.: 1.55   3rd Qu.: 2.73   3rd Qu.: 0.47
##  Max.   : 2.435    Max.   : 5.55   Max.   : 3.73   Max.   : 5.47
##    low_prices     return_policy
##  Min.   :-3.795   Min.   :-3.25
##  1st Qu.:-2.795   1st Qu.:-1.25
##  Median : 0.205   Median :-0.25
##  Mean   : 0.000   Mean   : 0.00
##  3rd Qu.: 2.205   3rd Qu.: 1.75
##  Max.   : 5.205   Max.   : 5.75
```

low_prices has the smallest min value of -3.795 electronics has the largest maximum value of 5.55

Tasks:3

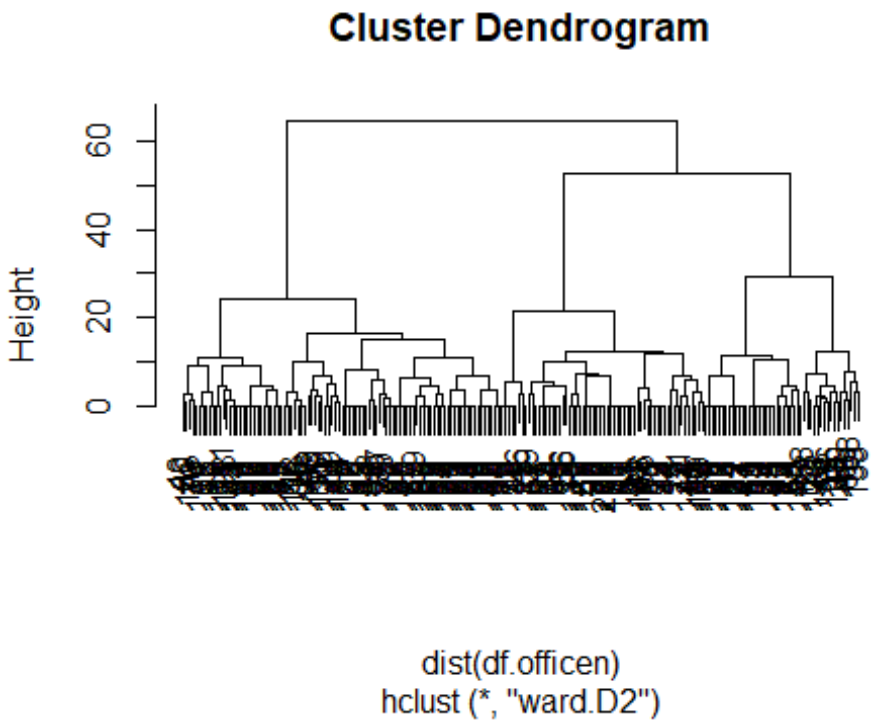
Run the hierarchical clustering algorithm using method = “ward.D2” on the normalised data and use set.seed(123)for reproducibility. Plot the dendogram.

- **Anwser**

```
set.seed(123)  # Setting seed
Hierar_office <- hclust(dist(df.officen),method = "ward.D2")
Hierar_office
```

```
##
## Call:
## hclust(d = dist(df.officen), method = "ward.D2")
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 200
```

```
# Plott dendrogram
plot(Hierar_office)
```



Tasks:4

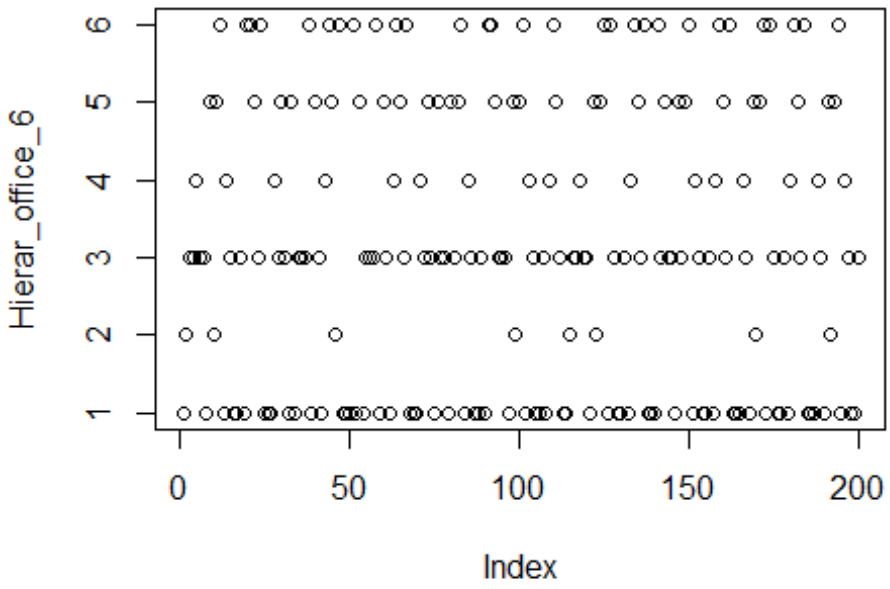
Suppose that after looking at the dendrogram and discussing with the marketing department, you decide to proceed with a 6-cluster solution. Divide the data points into 6 clusters. How many observations are assigned to each cluster?

- **Anwser**

```
# 6-cluster solution
Hierar_office_6 <- cutree(Hierar_office, k = 6 )

# display 6-cluster solution
Hierar_office_6
```

```
##   [1] 1 2 3 3 4 3 3 1 5 2 5 6 1 4 3 1 1 3 1 6 6 5 3 6 1 1 1 4 3 5 3 1 5 1
3 3 3
```

```
##  [38] 6 1 5 3 1 4 6 5 2 6 1 1 1 6 1 5 1 3 3 3 6 1 5 3 1 4 6 5 3 6 1 1 1 4
3 5 3
##  [75] 1 5 3 3 1 5 3 5 6 1 4 3 1 1 3 1 6 6 5 3 3 3 1 5 2 5 6 1 4 3 1 1 3 1
4 6 5
## [112] 3 1 1 2 3 3 4 3 3 1 5 2 5 6 1 6 3 1 1 3 1 4 6 5 3 6 1 1 1 6 3 5 3 3
1 5 3
## [149] 5 6 1 4 3 1 1 3 1 4 6 5 3 6 1 1 1 4 3 1 5 2 5 6 1 6 3 1 1 3 1 4 6 5
3 6 1
## [186] 1 1 4 3 1 5 2 5 6 1 4 3 1 1 3

table(Hierar_office_6)

## Hierar_office_6
##  1  2  3  4  5  6
## 64  8 52 17 30 29

plot.new()
plot(Hierar_office_6)
```



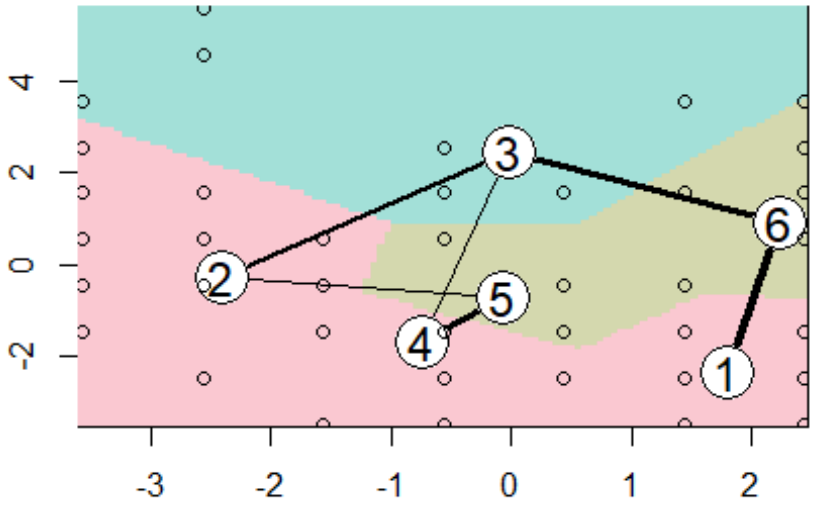
Tasks:5

Use the normalised data to calculate the means for each of the attitudinal variables per cluster. Use the flexclust package to generate a segment profile plot. Comment on whether any cluster memberships have changed, if any. Check the concordance between the hclust and as.kcca procedures.

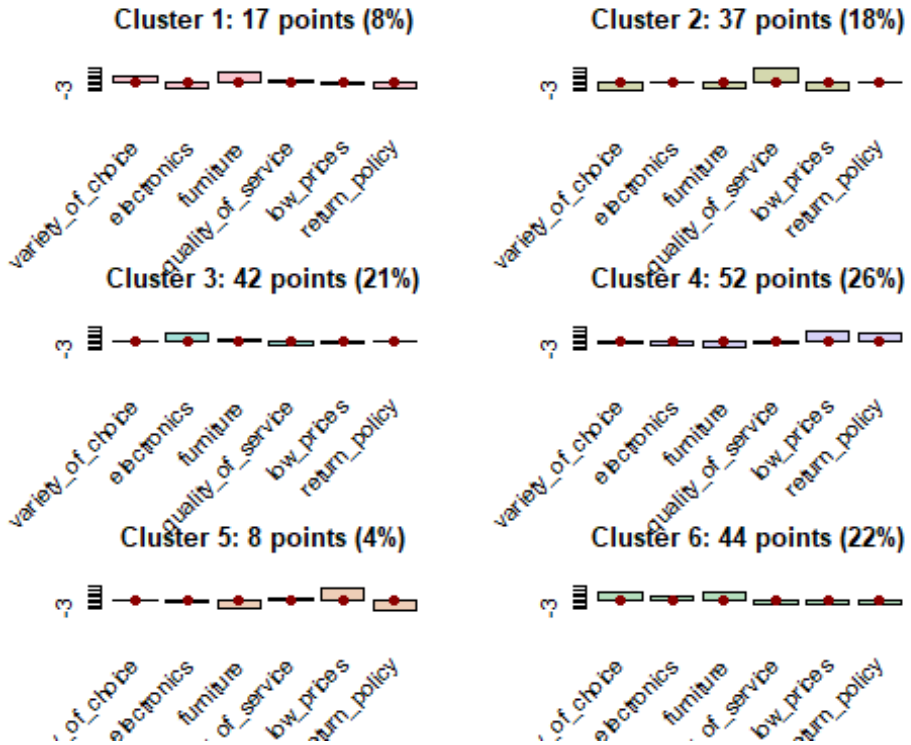
- **Anwser**

```
# library flexclust
library(flexclust)

## Warning: package 'flexclust' was built under R version 4.1.3

## Loading required package: grid

## Loading required package: lattice

## Loading required package: modeltools

## Loading required package: stats4

# means for each of the attitudinal variables per cluster
k2 <- kmeans(df.officen, centers = 6, nstart = 20)
str(k2)

## List of 9
##  $ cluster     : int [1:200] 4 2 1 1 5 1 1 6 6 2 ...
##  $ centers     : num [1:6, 1:6] -0.738 -0.065 -2.393 2.022 -2.624 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:6] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:6] "variety_of_choice" "electronics" "furniture"
"quality_of_service" ...
##  $ totss       : num 5856
##  $ withinss    : num [1:6] 346.8 16.9 162.2 523.4 178 ...
##  $ tot.withinss: num 1427
##  $ betweenss   : num 4429
##  $ size        : int [1:6] 52 8 29 63 17 31
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

k2

## K-means clustering with 6 clusters of sizes 52, 8, 29, 63, 17, 31
##
## Cluster means:
##   variety_of_choice electronics  furniture quality_of_service  low_prices
## 1        -0.7380769 -1.68076923 -1.9430769         -0.6069231  3.33961538
## 2        -0.0650000 -0.70000000 -2.8950000          0.4700000  3.45500000
## 3        -2.3925862 -0.72586207 -1.9596552          4.7458621 -2.65706897
## 4         2.0223016  0.02619048  2.7141270         -0.9903175 -1.44579365
## 5        -2.6238235  2.02058824 -1.1523529          0.1758824 -2.03029412
## 6         0.8220968  2.51774194  0.9558065         -1.6267742  0.04370968
##   return_policy
## 1     2.2115385
## 2    -3.2500000
## 3    -0.6293103
## 4    -1.4880952
## 5     2.5735294
```

```
## 6    -0.6693548
##
## Clustering vector:
##   [1] 4 2 1 1 5 1 1 6 6 2 6 3 4 5 1 4 4 1 4 3 3 6 1 3 4 4 4 5 1 6 1 4 6 4
1 1 1
##  [38] 3 4 6 1 4 5 3 6 2 3 4 4 4 3 4 6 4 1 1 1 3 4 6 1 4 5 3 6 1 3 4 4 4 5
1 6 1
##  [75] 4 6 1 1 4 6 1 6 3 4 5 1 4 4 1 4 3 3 6 1 1 1 4 6 2 6 3 4 5 1 4 4 1 4
5 3 6
## [112] 1 4 4 2 1 1 5 1 1 4 6 2 6 3 4 3 1 4 4 1 4 5 3 6 1 3 4 4 4 3 1 6 1 1
4 6 1
## [149] 6 3 4 5 1 4 4 1 4 5 3 6 1 3 4 4 4 5 1 4 6 2 6 3 4 3 1 4 4 1 4 5 3 6
1 3 4
## [186] 4 4 5 1 4 6 2 6 3 4 5 1 4 4 1
##
## Within cluster sum of squares by cluster:
## [1] 346.7885  16.8750 162.2069 523.3651 178.0000 200.1935
##  (between_SS / total_SS =  75.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

k2$size

## [1] 52  8 29 63 17 31

cl1 <- kcca(df.officen, k=6)
cl1

## kcca object of family 'kmeans'
##
## call:
## kcca(x = df.officen, k = 6)
##
## cluster sizes:
##
##  1  2  3  4  5  6
## 17 37 42 52  8 44

plot.new()
image(cl1)
points(df.officen)
```


4
2
0
-2
-3
-2
-1
0
1
2
3
6
2
5
4
1

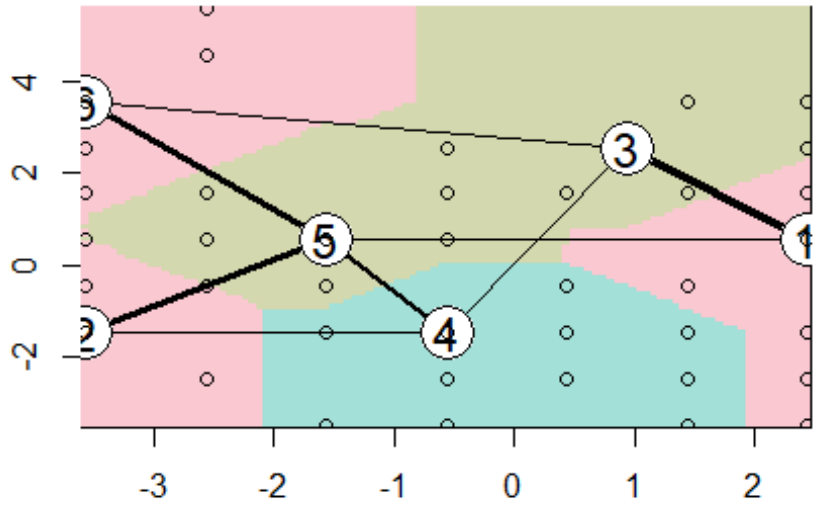
```
barplot(cl1)
```


Cluster 1: 17 points (8%)
Cluster 2: 37 points (18%)
Cluster 3: 42 points (21%)
Cluster 4: 52 points (26%)
Cluster 5: 8 points (4%)
Cluster 6: 44 points (22%)
-3
variety_of_choice
electronics
furniture
quality_of_service
low_prices
return_policy

```
cl2 <- kcca(df.officen, k=6, family=kccaFamily("kmedians"),
           control=list(initcent="kmeanspp"))
cl2
```

```
## kcca object of family 'kmedians'
##
## call:
## kcca(x = df.officen, k = 6, family = kccaFamily("kmedians"),
##     control = list(initcent = "kmeanspp"))
##
## cluster sizes:
##
##  1  2  3  4  5  6
## 62 24 32 60 13  9
```

```
image(cl2)
points(df.officen)
```




```
k2a <- as.kcca(k2, df.officen)
k2a
```

```
## kcca object of family 'kmeans'
##
## call:
## as.kcca(object = k2, data = df.officen)
##
## cluster sizes:
```

```
##
##  1  2  3  4  5  6
## 52  8 29 63 17 31

k2b <- as(k2a, "kmeans")
k2b

## K-means clustering with 6 clusters of sizes 52, 8, 29, 63, 17, 31
##
## Cluster means:
##   variety_of_choice electronics  furniture quality_of_service  low_prices
## 1        -0.7380769 -1.68076923 -1.9430769         -0.6069231  3.33961538
## 2        -0.0650000 -0.70000000 -2.8950000          0.4700000  3.45500000
## 3        -2.3925862 -0.72586207 -1.9596552          4.7458621 -2.65706897
## 4         2.0223016  0.02619048  2.7141270         -0.9903175 -1.44579365
## 5        -2.6238235  2.02058824 -1.1523529          0.1758824 -2.03029412
## 6         0.8220968  2.51774194  0.9558065         -1.6267742  0.04370968
##   return_policy
## 1     2.2115385
## 2    -3.2500000
## 3    -0.6293103
## 4    -1.4880952
## 5     2.5735294
## 6    -0.6693548
##
## Clustering vector:
##   [1] 4 2 1 1 5 1 1 6 6 2 6 3 4 5 1 4 4 1 4 3 3 6 1 3 4 4 4 5 1 6 1 4 6 4
1 1 1
##  [38] 3 4 6 1 4 5 3 6 2 3 4 4 4 3 4 6 4 1 1 1 3 4 6 1 4 5 3 6 1 3 4 4 4 5
1 6 1
##  [75] 4 6 1 1 4 6 1 6 3 4 5 1 4 4 1 4 3 3 6 1 1 1 4 6 2 6 3 4 5 1 4 4 1 4
5 3 6
## [112] 1 4 4 2 1 1 5 1 1 4 6 2 6 3 4 3 1 4 4 1 4 5 3 6 1 3 4 4 4 3 1 6 1 1
4 6 1
## [149] 6 3 4 5 1 4 4 1 4 5 3 6 1 3 4 4 4 5 1 4 6 2 6 3 4 3 1 4 4 1 4 5 3 6
1 3 4
## [186] 4 4 5 1 4 6 2 6 3 4 5 1 4 4 1
##
## Within cluster sum of squares by cluster:
## [1] 346.7885  16.8750 162.2069 523.3651 178.0000 200.1935
##
## Available components:
##
## [1] "cluster"  "centers"  "size"     "withinss"
```

All concordance changed

Tasks:6

Describe the 6-cluster solution using the cluster numbers corresponding to the hierarchical clustering procedure.

- **Anwser**

  The 6-Cluster Solution visually differentiated the 6 clusters in groups while th heirarchical clustering procedure was clumsy.

Tasks:7

Comment on why you may decide to NOT proceed with this 6-cluster solution.

- **Anwser**

  The accuracy and qaulity of clustering of the 6-Clustered Solution is impaired after then we may decide not to proceed with the 6-Cluster solutions

Tasks:8

Generate a 5-cluster solution. How many observations are assigned to each cluster?

- **Anwser**

```
# 5-cluster solution
Hierar_office_5 <- cutree(Hierar_office, k = 5 )

# display 5-cluster solution
Hierar_office_5
```

```
##   [1] 1 2 2 2 3 2 2 1 4 2 4 5 1 3 2 1 1 2 1 5 5 4 2 5 1 1 1 3 2 4 2 1 4 1
2 2 2
##  [38] 5 1 4 2 1 3 5 4 2 5 1 1 1 5 1 4 1 2 2 2 5 1 4 2 1 3 5 4 2 5 1 1 1 3
2 4 2
##  [75] 1 4 2 2 1 4 2 4 5 1 3 2 1 1 2 1 5 5 4 2 2 2 1 4 2 4 5 1 3 2 1 1 2 1
3 5 4
## [112] 2 1 1 2 2 2 3 2 2 1 4 2 4 5 1 5 2 1 1 2 1 3 5 4 2 5 1 1 1 5 2 4 2 2
1 4 2
## [149] 4 5 1 3 2 1 1 2 1 3 5 4 2 5 1 1 1 3 2 1 4 2 4 5 1 5 2 1 1 2 1 3 5 4
2 5 1
## [186] 1 1 3 2 1 4 2 4 5 1 3 2 1 1 2
```

```
table(Hierar_office_5)
```

```
## Hierar_office_5
##  1  2  3  4  5
## 64 60 17 30 29
```

Tasks:9

Repeat the steps performed previously to describe the clusters for the 5-cluster solution (i.e., calculate cluster means and segmentation plot). Describe the 5-cluster solution using the cluster numbers corresponding to the hierarchical clustering procedure. Give "expressive" labels to the clusters.

- **Anwser**

```
k3 <- kmeans(df.officen, centers = 5, nstart = 20)
str(k3)

## List of 9
##  $ cluster     : int [1:200] 1 2 2 2 3 2 2 5 5 2 ...
##  $ centers     : num [1:5, 1:6] 2.022 -0.648 -2.624 -2.393 0.822 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:6] "variety_of_choice" "electronics" "furniture"
"quality_of_service" ...
##  $ totss       : num 5856
##  $ withinss    : num [1:5] 523 595 178 162 200
##  $ tot.withinss: num 1658
##  $ betweenss   : num 4198
##  $ size        : int [1:5] 63 60 17 29 31
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

k3

## K-means clustering with 5 clusters of sizes 63, 60, 17, 29, 31
##
## Cluster means:
##   variety_of_choice electronics  furniture quality_of_service  low_prices
## 1         2.0223016  0.02619048  2.7141270         -0.9903175 -1.44579365
## 2        -0.6483333 -1.55000000 -2.0700000         -0.4633333  3.35500000
## 3        -2.6238235  2.02058824 -1.1523529          0.1758824 -2.03029412
## 4        -2.3925862 -0.72586207 -1.9596552          4.7458621 -2.65706897
## 5         0.8220968  2.51774194  0.9558065         -1.6267742  0.04370968
##   return_policy
## 1    -1.4880952
## 2     1.4833333
## 3     2.5735294
## 4    -0.6293103
## 5    -0.6693548
##
## Clustering vector:
##   [1] 1 2 2 2 3 2 2 5 5 2 5 4 1 3 2 1 1 2 1 4 4 5 2 4 1 1 1 3 2 5 2 1 5 1
2 2 2
##  [38] 4 1 5 2 1 3 4 5 2 4 1 1 1 4 1 5 1 2 2 2 4 1 5 2 1 3 4 5 2 4 1 1 1 3
2 5 2
```

```
##  [75] 1 5 2 2 1 5 2 5 4 1 3 2 1 1 2 1 4 4 5 2 2 2 1 5 2 5 4 1 3 2 1 1 2 1
3 4 5
## [112] 2 1 1 2 2 2 3 2 2 1 5 2 5 4 1 4 2 1 1 2 1 3 4 5 2 4 1 1 1 4 2 5 2 2
1 5 2
## [149] 5 4 1 3 2 1 1 2 1 3 4 5 2 4 1 1 1 3 2 1 5 2 5 4 1 4 2 1 1 2 1 3 4 5
2 4 1
## [186] 1 1 3 2 1 5 2 5 4 1 3 2 1 1 2
##
## Within cluster sum of squares by cluster:
## [1] 523.3651 594.7000 178.0000 162.2069 200.1935
##  (between_SS / total_SS =  71.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

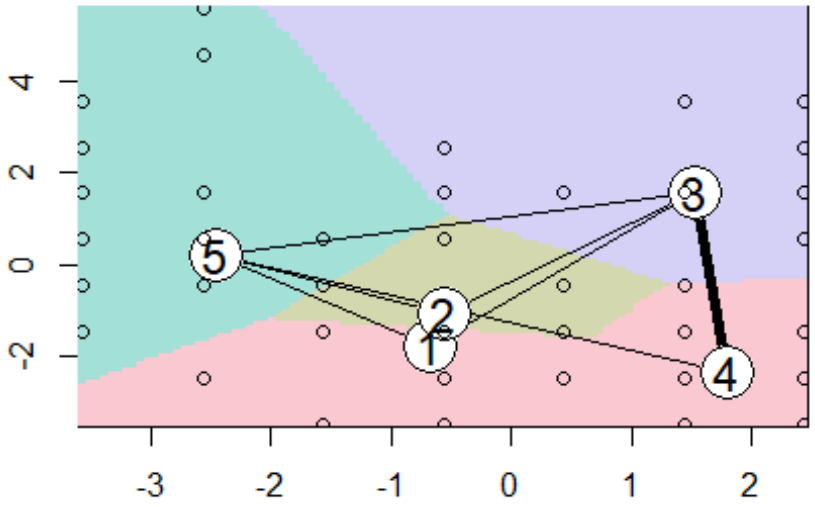
```
k3$centers
```

```
##   variety_of_choice electronics  furniture quality_of_service  low_prices
## 1         2.0223016  0.02619048  2.7141270         -0.9903175 -1.44579365
## 2        -0.6483333 -1.55000000 -2.0700000         -0.4633333  3.35500000
## 3        -2.6238235  2.02058824 -1.1523529          0.1758824 -2.03029412
## 4        -2.3925862 -0.72586207 -1.9596552          4.7458621 -2.65706897
## 5         0.8220968  2.51774194  0.9558065         -1.6267742  0.04370968
##   return_policy
## 1    -1.4880952
## 2     1.4833333
## 3     2.5735294
## 4    -0.6293103
## 5    -0.6693548
```

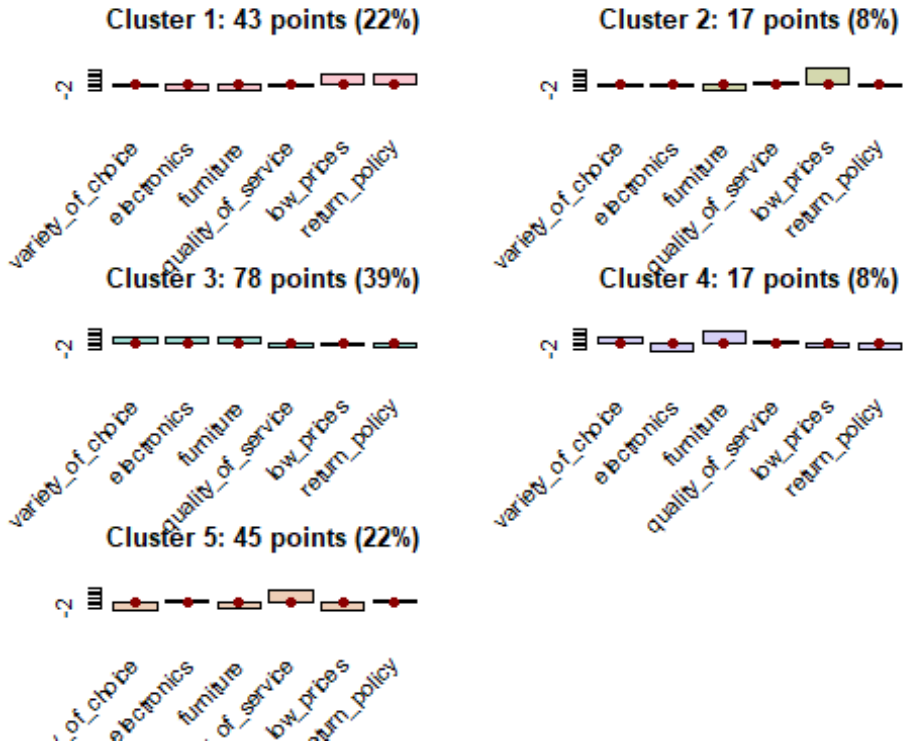
```
cl3 <- kcca(df.officen, k=5)
cl3
```

```
## kcca object of family 'kmeans'
##
## call:
## kcca(x = df.officen, k = 5)
##
## cluster sizes:
##
##  1  2  3  4  5
## 43 17 78 17 45
```

```
plot.new()
image(cl3)
points(df.officen)
```


4
2
0
-2
-3
-2
-1
0
1
2
5
3
2
1
4

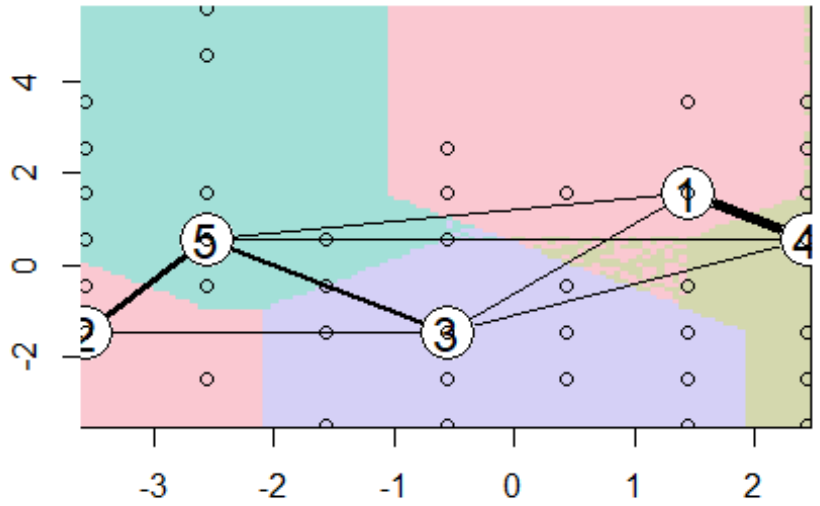
```
barplot(cl3)
```


Cluster 1: 43 points (22%)
Cluster 2: 17 points (8%)
Cluster 3: 78 points (39%)
Cluster 4: 17 points (8%)
Cluster 5: 45 points (22%)
-2
variety_of_choice
electronics
furniture
quality_of_service
low_prices
return_policy

```
cl4 <- kcca(df.officen, k=5, family=kccaFamily("kmedians"),
           control=list(initcent="kmeanspp"))
cl4
```

```
## kcca object of family 'kmedians'
##
## call:
## kcca(x = df.officen, k = 5, family = kccaFamily("kmedians"),
##     control = list(initcent = "kmeanspp"))
##
## cluster sizes:
##
##  1  2  3  4  5
## 35 23 62 57 23
```

```
image(cl4)
points(df.officen)
```




```
k3a <- as.kcca(k3, df.officen)
k3a
```

```
## kcca object of family 'kmeans'
##
## call:
## as.kcca(object = k3, data = df.officen)
##
## cluster sizes:
```

```
##
##  1  2  3  4  5
## 63 60 17 29 31

k3b <- as(k3a, "kmeans")
k3b

## K-means clustering with 5 clusters of sizes 63, 60, 17, 29, 31
##
## Cluster means:
##   variety_of_choice electronics  furniture quality_of_service  low_prices
## 1         2.0223016  0.02619048  2.7141270         -0.9903175 -1.44579365
## 2        -0.6483333 -1.55000000 -2.0700000         -0.4633333  3.35500000
## 3        -2.6238235  2.02058824 -1.1523529          0.1758824 -2.03029412
## 4        -2.3925862 -0.72586207 -1.9596552          4.7458621 -2.65706897
## 5         0.8220968  2.51774194  0.9558065         -1.6267742  0.04370968
##   return_policy
## 1    -1.4880952
## 2     1.4833333
## 3     2.5735294
## 4    -0.6293103
## 5    -0.6693548
##
## Clustering vector:
##   [1] 1 2 2 2 3 2 2 5 5 2 5 4 1 3 2 1 1 2 1 4 4 5 2 4 1 1 1 3 2 5 2 1 5 1
2 2 2
##  [38] 4 1 5 2 1 3 4 5 2 4 1 1 1 4 1 5 1 2 2 2 4 1 5 2 1 3 4 5 2 4 1 1 1 3
2 5 2
##  [75] 1 5 2 2 1 5 2 5 4 1 3 2 1 1 2 1 4 4 5 2 2 2 1 5 2 5 4 1 3 2 1 1 2 1
3 4 5
## [112] 2 1 1 2 2 2 3 2 2 1 5 2 5 4 1 4 2 1 1 2 1 3 4 5 2 4 1 1 1 4 2 5 2 2
1 5 2
## [149] 5 4 1 3 2 1 1 2 1 3 4 5 2 4 1 1 1 3 2 1 5 2 5 4 1 4 2 1 1 2 1 3 4 5
2 4 1
## [186] 1 1 3 2 1 5 2 5 4 1 3 2 1 1 2
##
## Within cluster sum of squares by cluster:
## [1] 523.3651 594.7000 178.0000 162.2069 200.1935
##
## Available components:
##
## [1] "cluster"  "centers"  "size"     "withinss"
```

Tasks:10

Comment on why you may find this 5-cluster solution better than the previous 6-cluster solution..

- **Anwser**

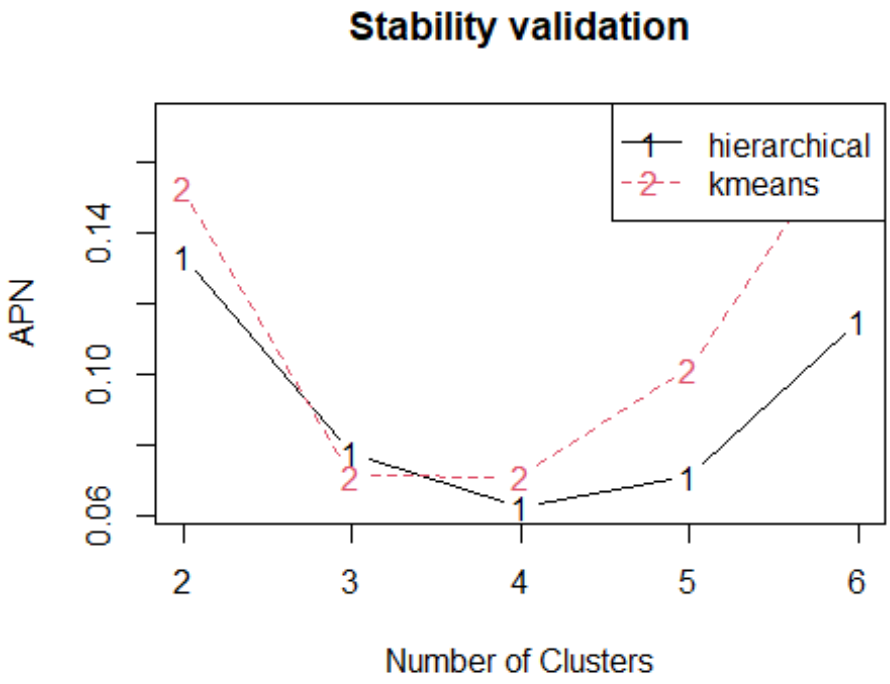
The 5-Cluster Solution shows more accuracy and detects the similarity better closer to the center

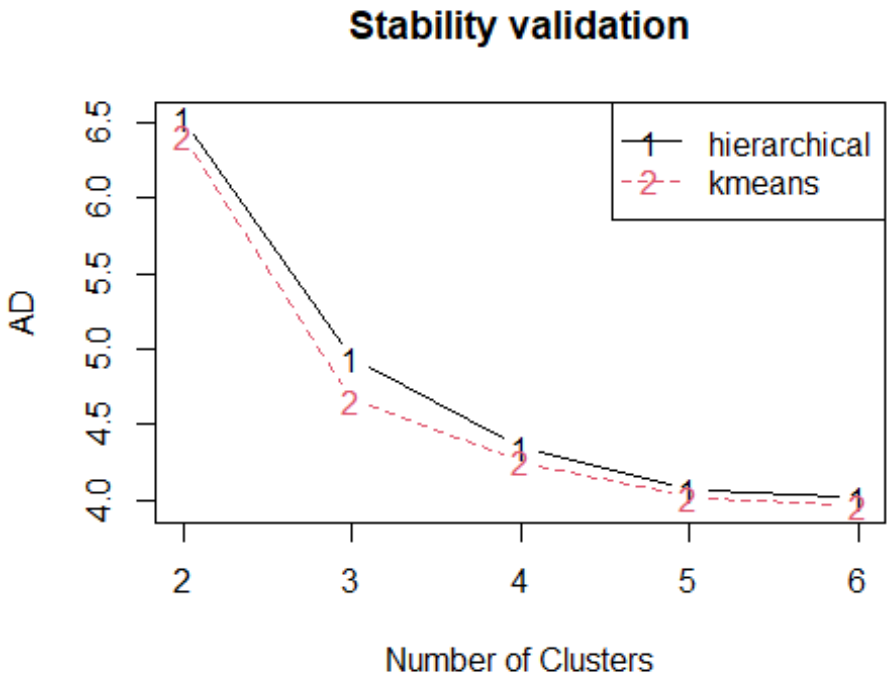
Tasks:11

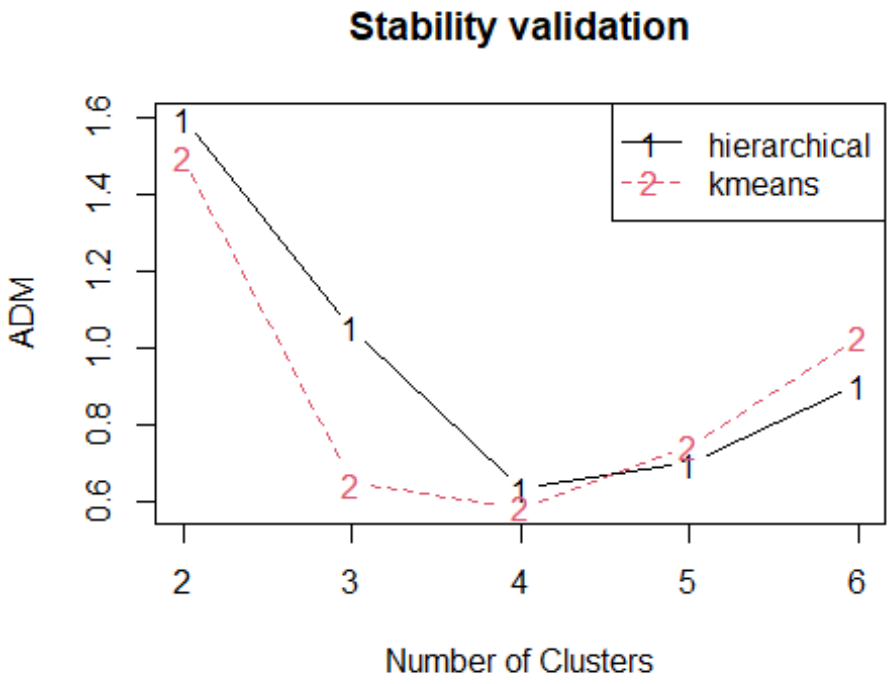
Use all the variables not included in the clustering procedure to evaluate whether the 5-cluster solution is meaningful. Generate ideas on how to target each segment (at least one idea per segment).

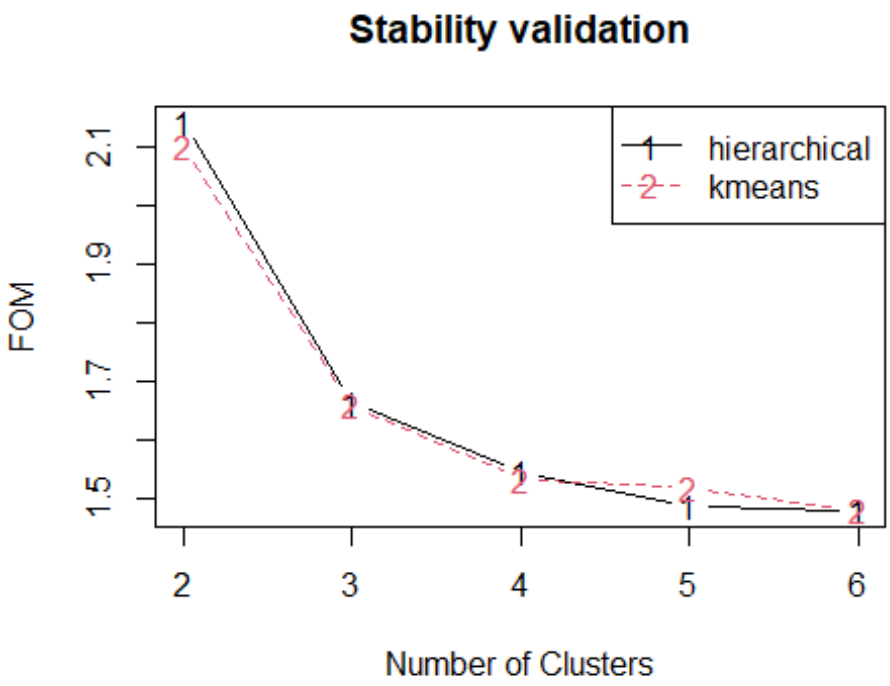
- **Anwser**

```
library(clValid)

## Warning: package 'clValid' was built under R version 4.1.3

## Loading required package: cluster

##
## Attaching package: 'clValid'

## The following object is masked from 'package:flexclust':
##
##     clusters

## The following object is masked from 'package:modeltools':
##
##     clusters

rownames(df.officen) <- 1:nrow( df.officen)
stab <- clValid(df.officen, 2:6,
clMethods=c("hierarchical","kmeans"),validation="stability")
plot(stab)
```

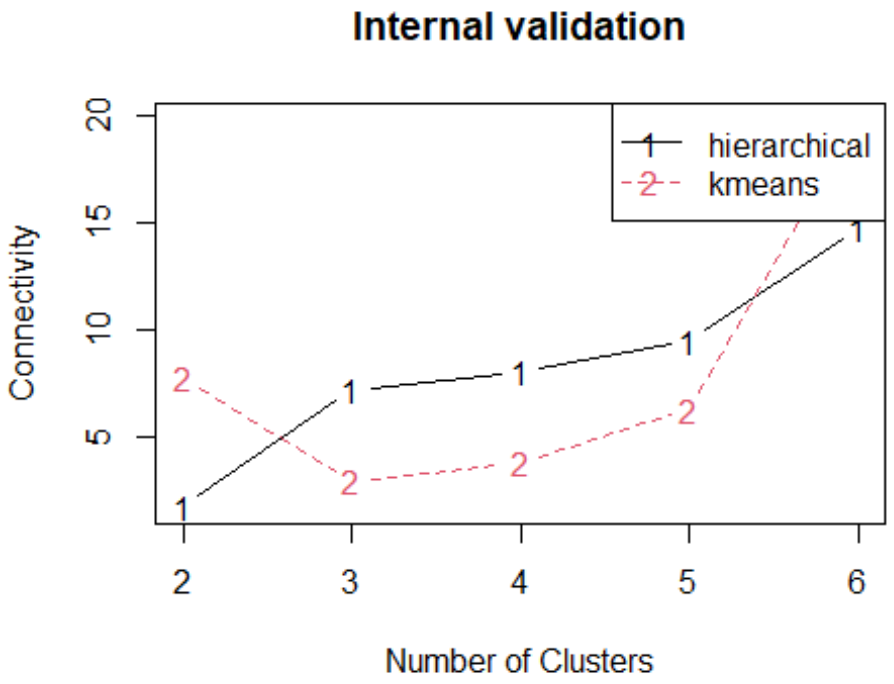

Stability validation
hierarchical
kmeans
APN
0.06
0.10
0.14
2
3
4
5
6
Number of Clusters

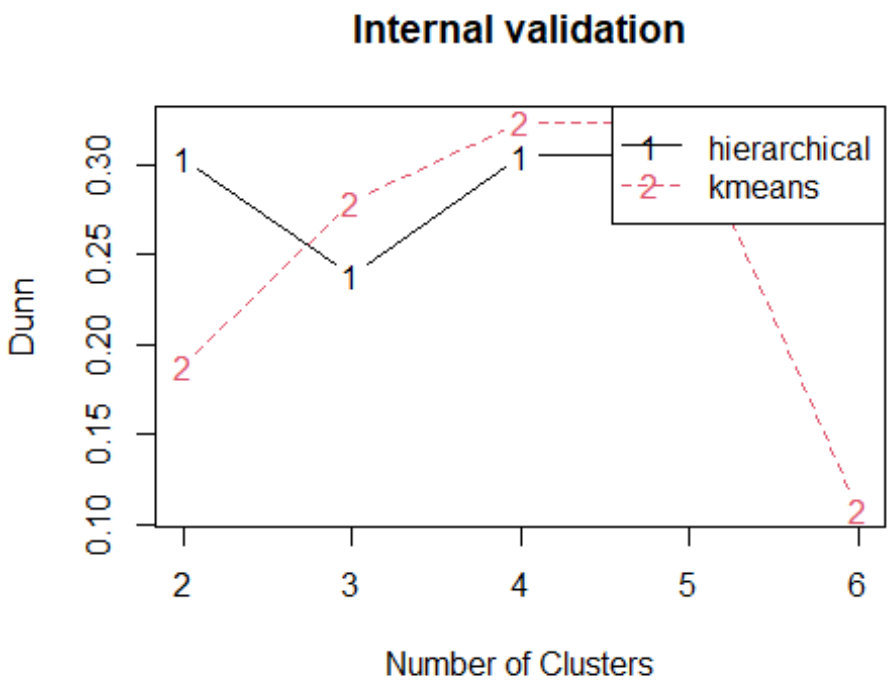

Stability validation
hierarchical
kmeans
AD
4.0
4.5
5.0
5.5
6.0
6.5
2
3
4
5
6
Number of Clusters

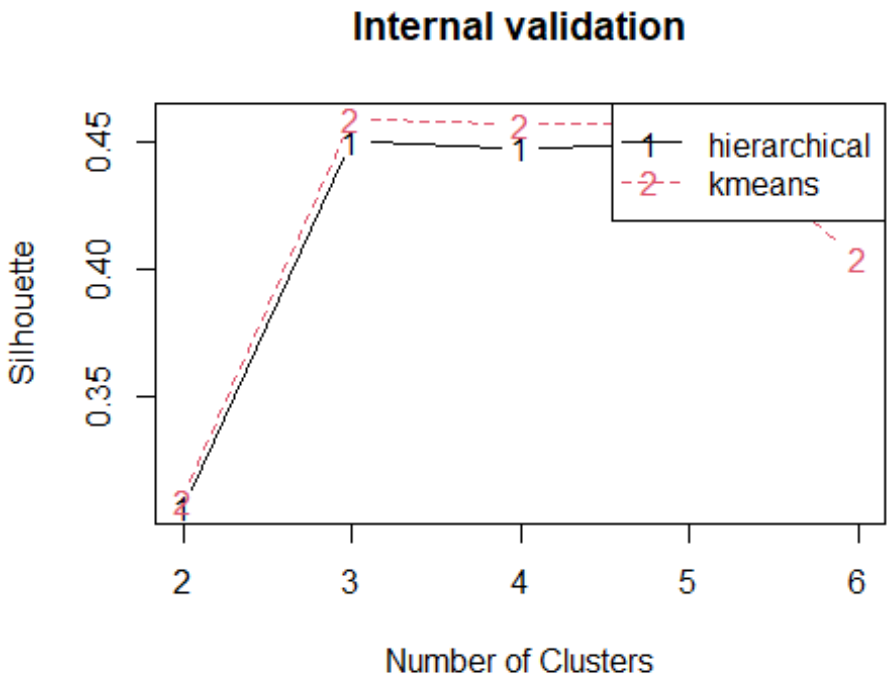





```
intern <- clValid(df.officen, 2:6,
clMethods=c("hierarchical","kmeans"),validation="internal")
plot(intern)
```


Internal validation
Connectivity
Number of Clusters
hierarchical
kmeans
5
10
15
20
2
3
4
5
6


Internal validation
Dunn
Number of Clusters
hierarchical
kmeans
0.10
0.15
0.20
0.25
0.30
2
3
4
5
6




```
summary(intern)

##
## Clustering Methods:
##  hierarchical kmeans
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2       3       4       5       6
##
## hierarchical Connectivity   1.7095  7.1786  8.0242  9.5020 14.7433
##              Dunn           0.3036  0.2382  0.3050  0.3050  0.3050
##              Silhouette     0.3067  0.4503  0.4477  0.4491  0.4285
## kmeans       Connectivity   7.7456  2.9492  3.7948  6.2726 19.7937
##              Dunn           0.1883  0.2793  0.3235  0.3235  0.1078
##              Silhouette     0.3097  0.4588  0.4569  0.4567  0.4048
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 1.7095 hierarchical 2
## Dunn         0.3235 kmeans       4
## Silhouette   0.4588 kmeans       3

optimalScores(stab)
```

```
##          Score       Method Clusters
## APN 0.06215602 hierarchical        4
## AD  3.95702148       kmeans        6
## ADM 0.58420959       kmeans        4
## FOM 1.47797553 hierarchical        6
```

Tasks:12

Run the k-means clustering algorithm on the normalised data, creating 5 clusters. Use iter.max = 1000 and nstart = 100 and set.seed(123)for reproducibility. How many observations are assigned to each cluster?

- **Anwser**

```
set.seed(123)
k5 <- kmeans(df.officen, centers = 5, nstart = 100,iter.max = 1000)
k5

## K-means clustering with 5 clusters of sizes 60, 17, 34, 29, 60
##
## Cluster means:
##   variety_of_choice   electronics  furniture quality_of_service low_prices
## 1        -0.6483333 -1.550000e+00 -2.0700000         -0.4633333  3.3550000
## 2        -2.6238235  2.020588e+00 -1.1523529          0.1758824 -2.0302941
## 3         0.7879412  2.344118e+00  0.8770588         -1.6182353 -0.2067647
## 4        -2.3925862 -7.258621e-01 -1.9596552          4.7458621 -2.6570690
## 5         2.1016667 -1.184238e-16  2.8466667         -0.9633333 -1.3783333
##   return_policy
## 1     1.4833333
## 2     2.5735294
## 3    -0.7205882
## 4    -0.6293103
## 5    -1.5000000
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
19  20
##   5   1   1   1   2   1   1   3   3   1   3   4   5   2   1   5   5   1
5   4
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
39  40
##   4   3   1   4   5   5   5   2   1   3   1   5   3   5   1   1   1   4
5   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
59  60
##   1   5   2   4   3   1   4   5   5   5   4   5   3   5   1   1   1   4
5   3
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80
##   1   5   2   4   3   1   4   5   5   5   2   1   3   1   5   3   1   1
```

```
5   3
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
99 100
##   1   3   4   5   2   1   5   5   1   5   4   4   3   1   1   1   3   3
1   3
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119 120
##   4   5   2   1   5   5   1   5   2   4   3   1   5   5   1   1   1   2
1   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138
139 140
##   5   3   1   3   4   5   4   1   5   5   1   5   2   4   3   1   4   5
5   5
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159 160
##   4   1   3   1   1   3   3   1   3   4   5   2   1   5   5   1   5   2
4   3
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
179 180
##   1   4   5   5   5   2   1   3   3   1   3   4   5   4   1   5   5   1
5   2
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
199 200
##   4   3   1   4   5   5   5   2   1   5   3   1   3   4   5   2   1   5
5   1
##
## Within cluster sum of squares by cluster:
## [1] 594.7000 178.0000 245.0294 162.2069 476.9333
##  (between_SS / total_SS =  71.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
k5$size
```

```
## [1] 60 17 34 29 60
```

Tasks:13

Check the concordance between the hclust and kmeans procedures. What is the Hit Rate?

- **Anwser**

```
k2$clusters
```

```
## NULL
```

```
Hierar_office_6
```

```
##   [1] 1 2 3 3 4 3 3 1 5 2 5 6 1 4 3 1 1 3 1 6 6 5 3 6 1 1 1 4 3 5 3 1 5 1
3 3 3
##  [38] 6 1 5 3 1 4 6 5 2 6 1 1 1 6 1 5 1 3 3 3 6 1 5 3 1 4 6 5 3 6 1 1 1 4
3 5 3
##  [75] 1 5 3 3 1 5 3 5 6 1 4 3 1 1 3 1 6 6 5 3 3 3 1 5 2 5 6 1 4 3 1 1 3 1
4 6 5
## [112] 3 1 1 2 3 3 4 3 3 1 5 2 5 6 1 6 3 1 1 3 1 4 6 5 3 6 1 1 1 6 3 5 3 3
1 5 3
## [149] 5 6 1 4 3 1 1 3 1 4 6 5 3 6 1 1 1 4 3 1 5 2 5 6 1 6 3 1 1 3 1 4 6 5
3 6 1
## [186] 1 1 4 3 1 5 2 5 6 1 4 3 1 1 3

cor.test(k2$cluster,Hierar_office_6)

##
##  Pearson's product-moment correlation
##
## data:  k2$cluster and Hierar_office_6
## t = 1.7918, df = 198, p-value = 0.0747
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.01264698  0.26049154
## sample estimates:
##       cor
## 0.1263157

print(cor.test(k2$cluster,Hierar_office_6))

##
##  Pearson's product-moment correlation
##
## data:  k2$cluster and Hierar_office_6
## t = 1.7918, df = 198, p-value = 0.0747
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.01264698  0.26049154
## sample estimates:
##       cor
## 0.1263157
```