WholeSaleCustomer

Truc Huynh

11/16/2020

Table of Contents

Goal:

The goal is to segment the clients of a wholesale distributor based on their annual pending on diverse product categories. For the data analysis, I am going to use R and R markdown.

```
knitr::opts_chunk$set(echo = TRUE)

#library to use for the analysis
library(psych)
library(e1071)
library(rpart)
library(rpart.plot)
library(caTools)
library(readr)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha

library(RColorBrewer)
library(fpc)
```

Data Description

Download a customer data (named with Wholesale customers data.csv). For the detail of the dataset, refer to http://archive.ics.uci.edu/ml/datasets/Wholesale+customers# The data includes the annual spending in monetary units on diverse product categories.There are 8 attributes. Two attributes, CHANNEL and REGION, are nominal, and the others are continuous. (1) FRESH: annual spending (m.u.) on fresh products (Continuous); (2) MILK: annual spending (m.u.) on milk products (Continuous); (3) GROCERY: annual spending (m.u.) on grocery products (Continuous); (4) FROZEN: annual spending (m.u.) on frozen products (Continuous) (5) DETERGENTS_PAPER: annual spending (m.u.) on detergents and paper products (Continuous) (6) DELICATESSEN: annual spending (m.u.) on and delicatessen products (Continuous); (7) CHANNEL: customers Channel — Horeca (Hotel/Restaurant/Caf??) or Retail channel (Nominal) (8) REGION: customers Region of Lisbon, Oporto or Other (Nominal) For this study, we exclude the nominal attributes. Prepare a datatset (called org_data) with 6 continuous attributes.

```
# Import data
WholesaleData <-
read.csv("~/R/DataMining/WholeSale/WholesaleCustomersData.csv")

# copy import data to new data frame
org_data <- WholesaleData

# remove the nominal attributes from the dataframe
org_data$Channel <- NULL
org_data$Region  <- NULL
```

Data Exploration

Explore min, max, mean, standard deviation, correlation, and else using describe function.

```
#Explore the data
describe(org_data)

##                 vars   n     mean       sd median trimmed     mad min
max
## Fresh              1 440 12000.30 12647.33 8504.0 9864.61 8776.25   3
112151
## Milk               2 440  5796.27  7380.38 3627.0 4375.52 3647.20  55
73498
```

```
## Grocery              3 440  7951.28  9503.16 4755.5 6158.43 4586.42   3
92780
## Frozen               4 440  3071.93  4854.67 1526.0 2144.07 1607.88  25
60869
## Detergents_Paper     5 440  2881.49  4767.85  816.5 1849.73 1060.80   3
40827
## Delicassen           6 440  1524.87  2820.11  965.5 1113.24  945.16   3
47943
##                    range  skew kurtosis     se
## Fresh             112148  2.54    11.33 602.94
## Milk               73443  4.03    24.25 351.85
## Grocery            92777  3.56    20.56 453.05
## Frozen             60844  5.87    53.80 231.44
## Detergents_Paper   40824  3.61    18.68 227.30
## Delicassen         47940 11.08   167.97 134.44
```

Data Transformation I (Standardization)

- There is a lot of variation in the magnitude of the original data (org_data). To bring all the features to the same magnitude, standardize the features.
- Also, showing first 10 rows in the transformed data (called trans_data).

```
#To standardize
library(BBmisc)

##
## Attaching package: 'BBmisc'

## The following object is masked from 'package:base':
##
##     isFALSE

# Data
# Normalized Data
trans_data <-
  normalize(
    org_data,
    method = "standardize",
    range = c(0, 1),
    margin = 1L,
    on.constant = "quiet"
  )
head(trans_data,10)

##           Fresh        Milk     Grocery     Frozen Detergents_Paper
Delicassen
## 1   0.052873004  0.52297247 -0.04106815 -0.5886970      -0.04351919 -
0.06626363
## 2  -0.390857056  0.54383861  0.17012470 -0.2698290       0.08630859
0.08904969
## 3  -0.446520984  0.40807319 -0.02812509 -0.1373793       0.13308016
2.24074190
```

```
## 4   0.099997579 -0.62331041 -0.39253008  0.6863630      -0.49802132
0.09330484
## 5   0.839284120 -0.05233688 -0.07926595  0.1736612      -0.23165413
1.29786952
## 6  -0.204572662  0.33368675 -0.29729863 -0.4955909      -0.22787885 -
0.02619421
## 7   0.009939037 -0.35191506 -0.10273183 -0.5339045       0.05421869 -
0.34745874
## 8  -0.349583519 -0.11385135  0.15518231 -0.2889858       0.09218126
0.36918101
## 9  -0.477357535 -0.29107807 -0.18512545 -0.5452338      -0.24444815 -
0.27476643
## 10 -0.473957607  0.71767797  1.15011422 -0.3940392       0.95294579
0.20322979

# Can also use the scale function to perform the same as normalize
# trans_data2 <- scale(org_data)
# head(trans_data2,10)
```

Data Transformation II (Dimensionality Reduction)

- Conduct Principal Component Analysis (PCA) analysis to the trans_data.
- Also, show first 10 data rows with principle components.

```
# Find the covariance matrix S of the data.
S <- cov(trans_data[])

S # View the data
```

```
##                        Fresh      Milk     Grocery      Frozen
Detergents_Paper
## Fresh             1.00000000 0.1005098 -0.01185387  0.34588146       -
0.1019529
## Milk              0.10050977 1.0000000  0.72833512  0.12399376
0.6618157
## Grocery          -0.01185387 0.7283351  1.00000000 -0.04019274
0.9246407
## Frozen            0.34588146 0.1239938 -0.04019274  1.00000000       -
0.1315249
## Detergents_Paper -0.10195294 0.6618157  0.92464069 -0.13152491
1.0000000
## Delicassen        0.24468997 0.4063683  0.20549651  0.39094747
0.0692913
##                  Delicassen
## Fresh             0.2446900
## Milk              0.4063683
## Grocery           0.2054965
## Frozen            0.3909475
## Detergents_Paper  0.0692913
## Delicassen        1.0000000
```

```r
# The total variance Which is also equal to the sum of the eigenvalues of S
sum (diag(S))
```

```
## [1] 6
```

```r
s.eigen <- eigen(S)
s.eigen
```

```
## eigen() decomposition
## $values
## [1] 2.64497357 1.70258397 0.74006477 0.56373023 0.28567634 0.06297111
##
## $vectors
##             [,1]        [,2]        [,3]        [,4]        [,5]
[,6]
## [1,] -0.04288396  0.52793212  0.81225657  0.23668559  0.04868278 -
0.03602539
## [2,] -0.54511832  0.08316765 -0.06038798  0.08718991 -0.82657929 -
0.03804019
## [3,] -0.57925635 -0.14608818  0.10838401 -0.10598745  0.31499943
0.72174458
## [4,] -0.05118859  0.61127764 -0.17838615 -0.76868266  0.02793224 -
0.01563715
## [5,] -0.54864020 -0.25523316  0.13619225 -0.17174406  0.33964012 -
0.68589373
## [6,] -0.24868198  0.50420705 -0.52390412  0.55206472  0.31470051 -
0.07513412
```

- The percent of the total variance in the dataset the principle component 1 and principle component 2 account:
- The eigen-vectors represent the principal components of S.
- The eigenvalues of S are used to find the proportion of the total variance explained by the components.

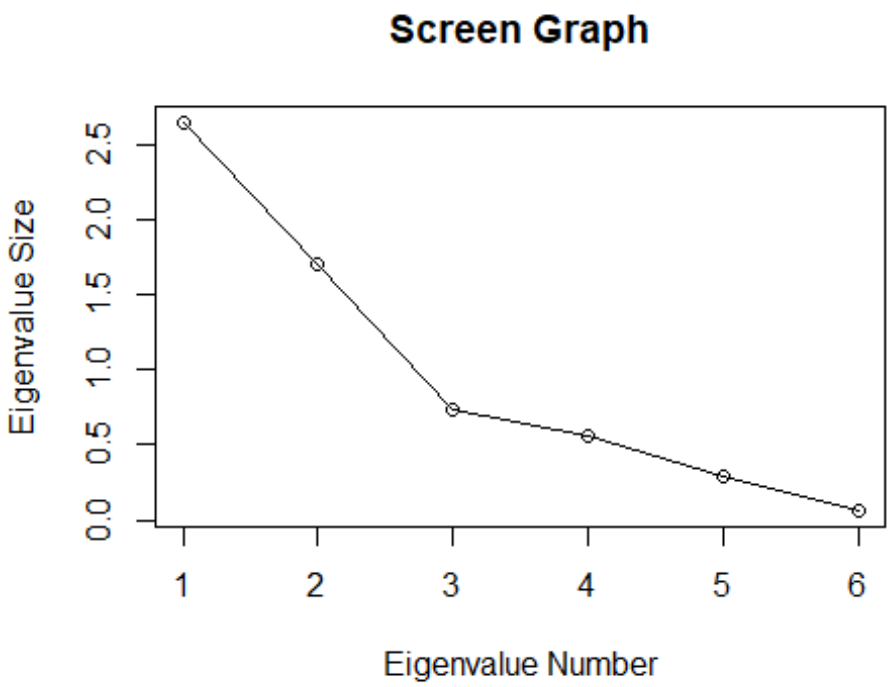
```r
for (s in s.eigen$values) {
  print(s / sum(s.eigen$values))
}
```

```
## [1] 0.4408289
## [1] 0.283764
## [1] 0.1233441
## [1] 0.09395504
## [1] 0.04761272
## [1] 0.01049519
```

- The first two principal components account for 72.45% (0.4408 + 0.2837) of the total variance.
- Generate a data set (called reduced_data) with 2 dimensions.

```r
trans_data.pca <- prcomp(trans_data[])

trans_data.pca
```

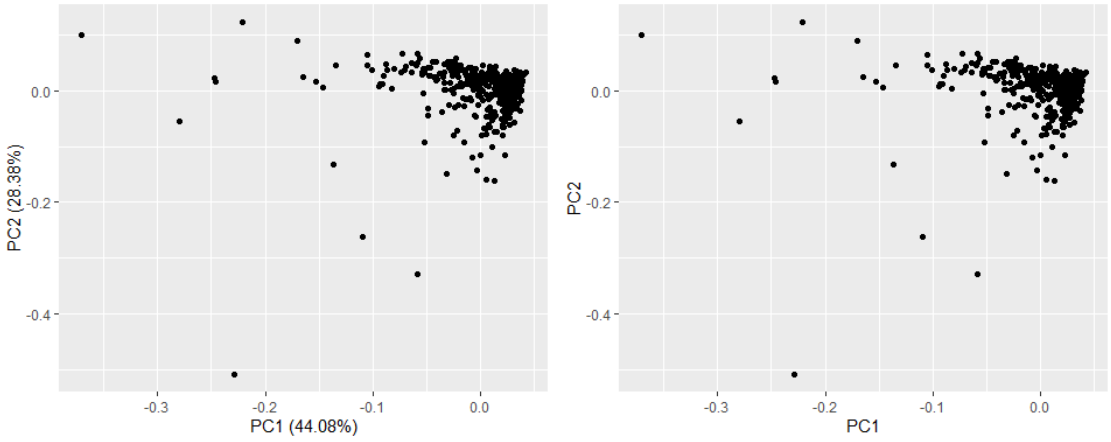
```
## Standard deviations (1, .., p=6):
## [1] 1.6263375 1.3048310 0.8602702 0.7508197 0.5344870 0.2509405
##
## Rotation (n x k) = (6 x 6):
##                          PC1         PC2         PC3         PC4
PC5
## Fresh            -0.04288396 -0.52793212 -0.81225657 -0.23668559
0.04868278
## Milk             -0.54511832 -0.08316765  0.06038798 -0.08718991 -
0.82657929
## Grocery          -0.57925635  0.14608818 -0.10838401  0.10598745
0.31499943
## Frozen           -0.05118859 -0.61127764  0.17838615  0.76868266
0.02793224
## Detergents_Paper -0.54864020  0.25523316 -0.13619225  0.17174406
0.33964012
## Delicassen       -0.24868198 -0.50420705  0.52390412 -0.55206472
0.31470051
##                          PC6
## Fresh             0.03602539
## Milk              0.03804019
## Grocery          -0.72174458
## Frozen            0.01563715
## Detergents_Paper  0.68589373
## Delicassen        0.07513412

#The summary method of prcomp() also outputs the proportion of variance
explained by the components.
summary(trans_data.pca)

## Importance of components:
##                            PC1    PC2    PC3     PC4     PC5    PC6
## Standard deviation     1.6263 1.3048 0.8603 0.75082 0.53449 0.2509
## Proportion of Variance 0.4408 0.2838 0.1233 0.09396 0.04761 0.0105
## Cumulative Proportion  0.4408 0.7246 0.8479 0.94189 0.98950 1.0000

scaling <- trans_data.pca$sdev[1:2] * sqrt(nrow(trans_data))
pc1 <- rowSums(t(t(sweep(trans_data[], 2 ,colMeans(trans_data[]))) *
s.eigen$vectors[,1]) / scaling[1])
pc2 <- rowSums(t(t(sweep(trans_data[], 2, colMeans(trans_data[]))) *
s.eigen$vectors[,2])*-1 / scaling[2])

reduced_data <- data.frame(pc1, pc2)
colnames(reduced_data) <- c('PC1', 'PC2')

# Data
plot(s.eigen$values, xlab = 'Eigenvalue Number', ylab = 'Eigenvalue Size',
main = 'Screen Graph')
lines(s.eigen$values)
```



Data Visualization

- Plot original features with the principal components in the reduced_data with 2 features (dimensions).
- Plot 1 using the whole trans_data data set
- Plot 2 using the reduced_data set

```
library(ggfortify)

## Warning: package 'ggfortify' was built under R version 4.0.3

# Data create using prcomp() of R library
plot1 <- autoplot(trans_data.pca, data = trans_data, color= 'Group')

## Warning: `select_()` is deprecated as of dplyr 0.7.0.
## Please use `select()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

#Data create using eigenvalue
plot2 <- ggplot(reduced_data, aes(x=PC1, y=PC2)) +
  geom_point()

#Add the 2 grid for comparison
library(gridExtra)
grid.arrange(plot1, plot2, nrow = 1)
```



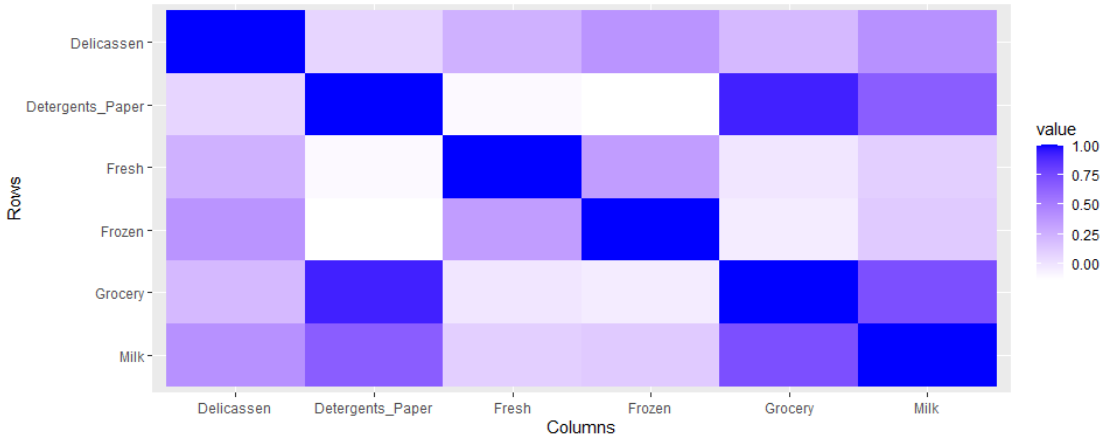
(2) Can you find any cluster tendency visually: Yes

Cluster tendency

(1) Compute Hopkins statistic.

(2) Determine whether the warehouse customer data shows useful clustering tendencies using the Hopkins statistic value.

```
#
plot1 <- autoplot(S, data = trans_data, color= 'Group')

## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.

plot1
```

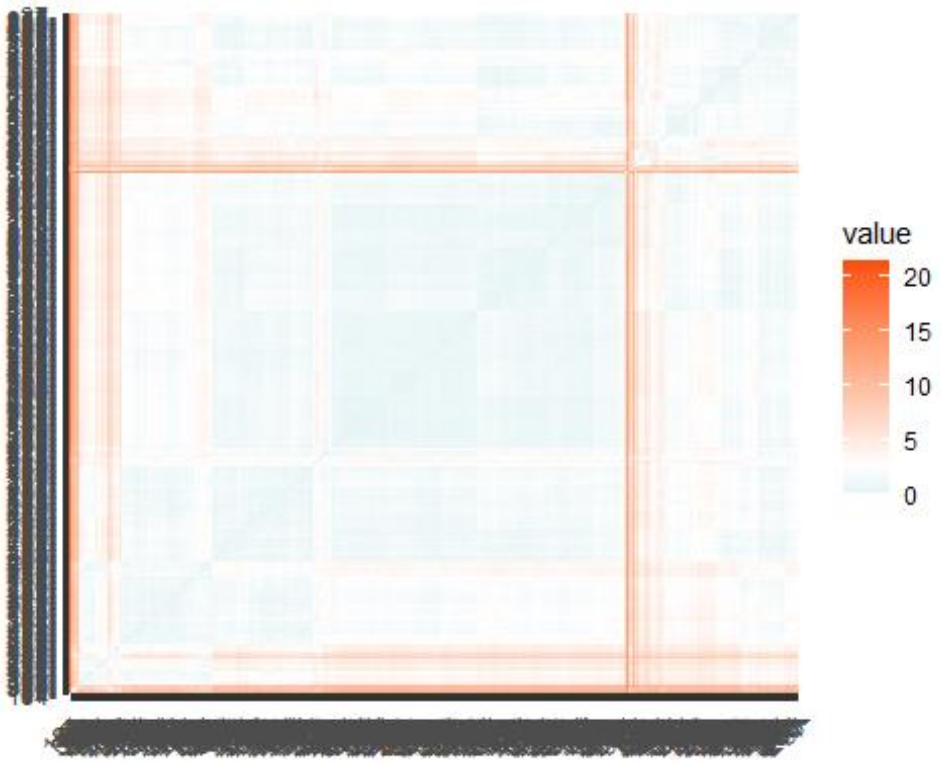


##

Optimal number of clusters (k) Before the actual clustering, identify the optimal number of clusters (k) for the data with the trans_data data using (1) Elbow method and (2) Shilhouette method. For each result, draw the plot figure and give an optimal number of clusters with your reason.

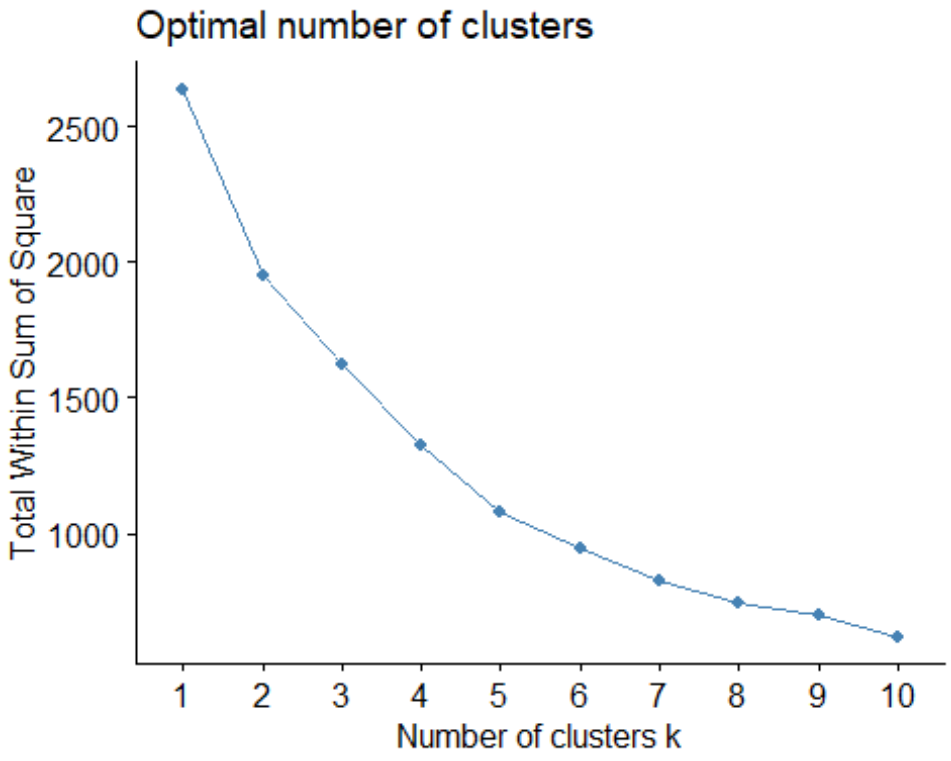
Get_dist()

```
# The get distnc function()
library(factoextra)  # library for get_dist

## Warning: package 'factoextra' was built under R version 4.0.3

## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa

distance <- get_dist(trans_data)
# Visualizing distance matrix
fviz_dist(distance, gradient = list(low = "#00AFBB", mid = "white", high =
"#FC4E07"))
```



Elbow method

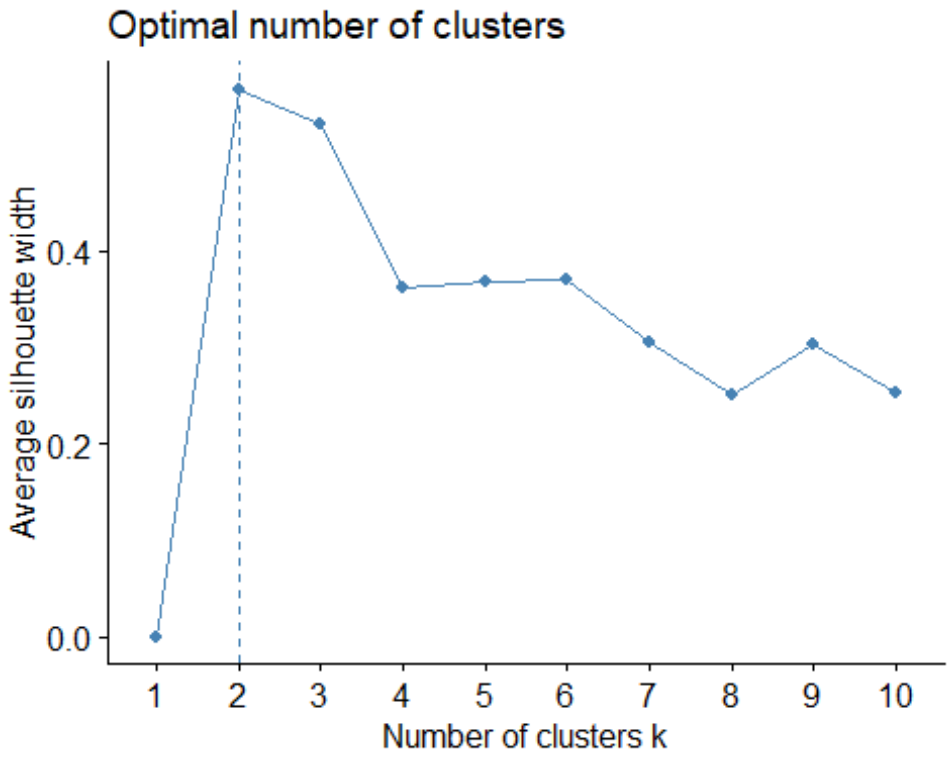
```
set.seed(123)
fviz_nbclust(trans_data, kmeans, method = "wss")
```



Optimal number of clusters: 10 Reason: It make sense to have 10 clusters according to the analyst.

Silhoute Method

```
fviz_nbclust(trans_data, kmeans, method = "silhouette")
```



Optimal number of clusters: 10 Reason: It make sense to have 10 clusters according to the analyst.

Representative-based clustering

Perform the cluster analysis using k-means with k=6. For the input data, use the trans_data data. (1) Report the representative of each cluster.

```
# Clustering
k2 <- kmeans(trans_data, centers = 6, nstart = 25)
str(k2)

## List of 9
##  $ cluster     : int [1:440] 4 2 2 4 3 4 4 4 4 2 ...
##  $ centers     : num [1:6, 1:6] 0.313 -0.504 1.261 -0.332 1.965 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:6] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:6] "Fresh" "Milk" "Grocery" "Frozen" ...
##  $ totss       : num 2634
##  $ withinss    : num [1:6] 149 224 236 192 0 ...
##  $ tot.withinss: num 920
##  $ betweenss   : num 1714
##  $ size        : int [1:6] 10 94 84 246 1 5
##  $ iter        : int 4
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

fviz_cluster(k2, data = trans_data)
```

(2) Compute SSE of each cluster and total SSE of the clustering.

```
# Sum of Square of each Cluster
k2[["withinss"]]
```

```
## [1] 149.4481 224.3460 236.4930 192.1842   0.0000 117.8370
```

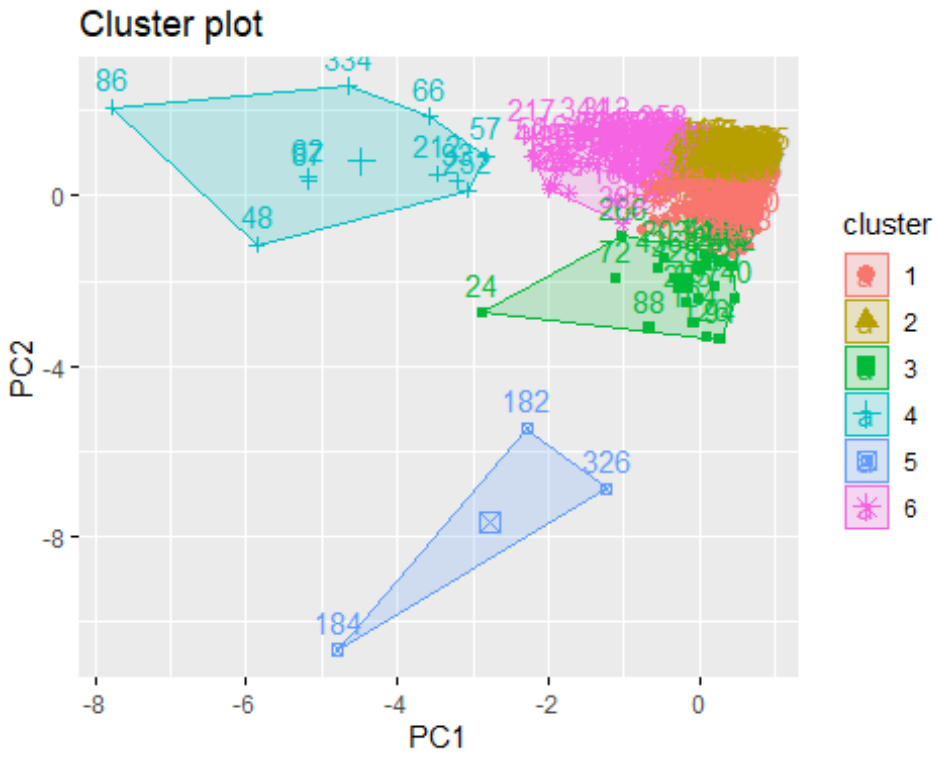
```
# Total SSE
k2[["tot.withinss"]]
```

```
## [1] 920.3082
```

Visualization of clusters I

Visualize the clustering result with the reduce_data with two features from PCA and the cluster label from the k-mean clustering. When you plot the data points, use different color per cluster.

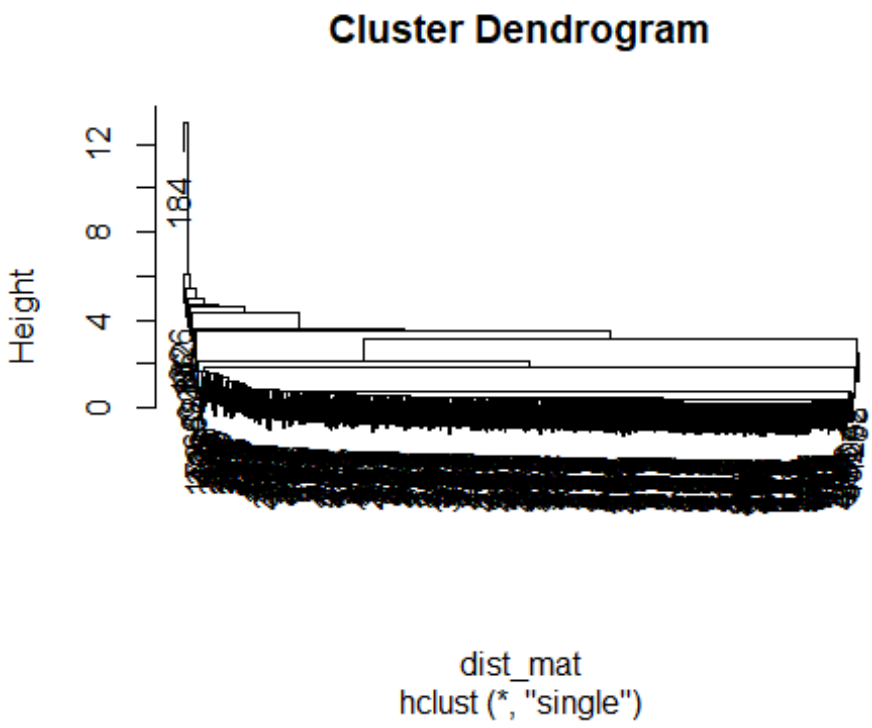
```
# Data
 # Clustering
k2 <- kmeans(reduced_data, centers = 6, nstart = 25)
fviz_cluster(k2, data = reduced_data)
```



Hierarchal clustering

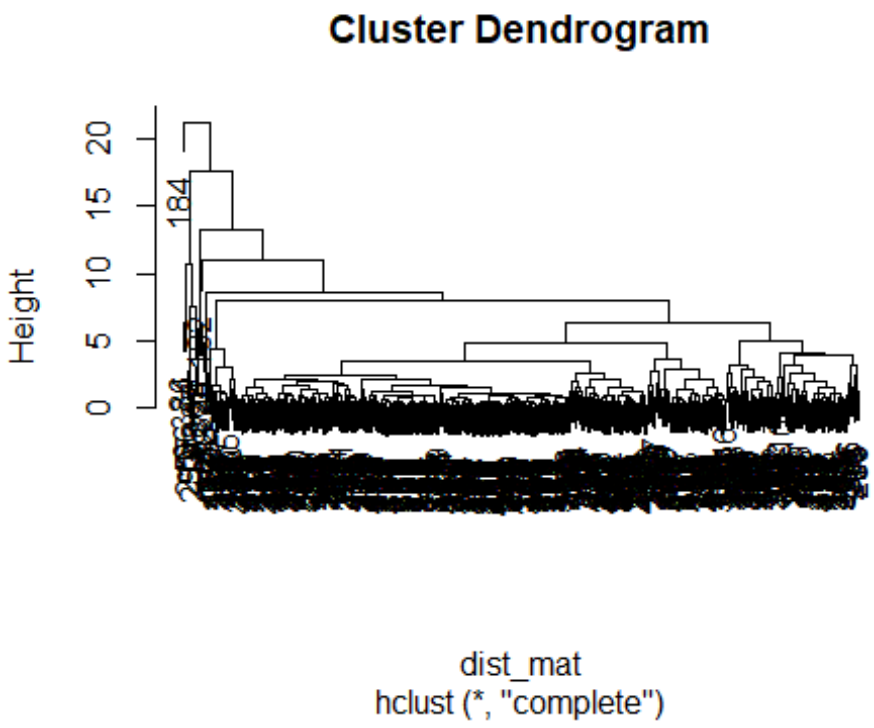
(1) Perform the cluster analysis using a single link agglomerative hierarchical clustering algorithm. For the input data, use the trans_data data. Show the cluster dendogram.

```
dist_mat <- dist(trans_data, method = 'euclidean')
hclust_ <- hclust(dist_mat, method = 'single')
plot(hclust_)
```



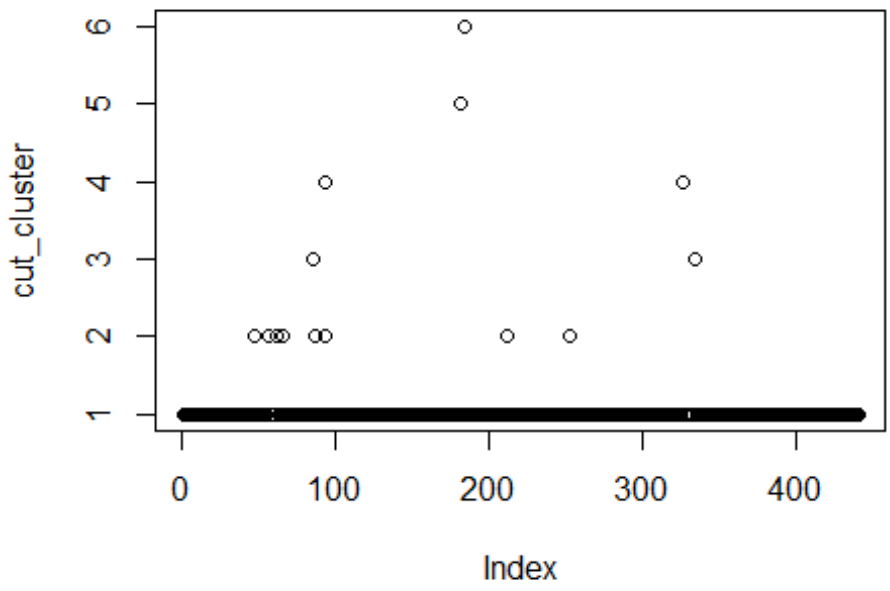
(2) Perform the cluster analysis using a complete link agglomerative hierarchical clustering algorithm. For the input data, use the trans_data data. Show the cluster dendogram.

```
hclust_ <- hclust(dist_mat, method = 'complete')
plot(hclust_)
```

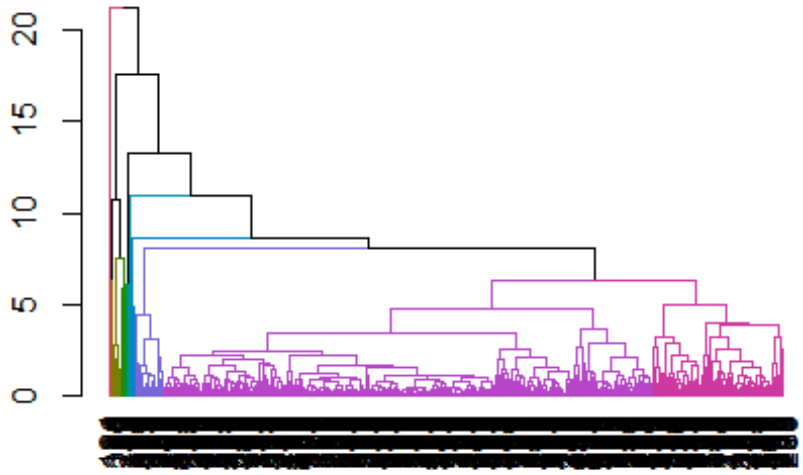


(3) From the complete link hierarchical clustering result, report 6 clusters with their data points. And plot the clustering result with two features from PCA and the cluster label from the clustering result.

```
# Data
cut_cluster <- cutree(hclust_, k = 6)
plot(cut_cluster)
```



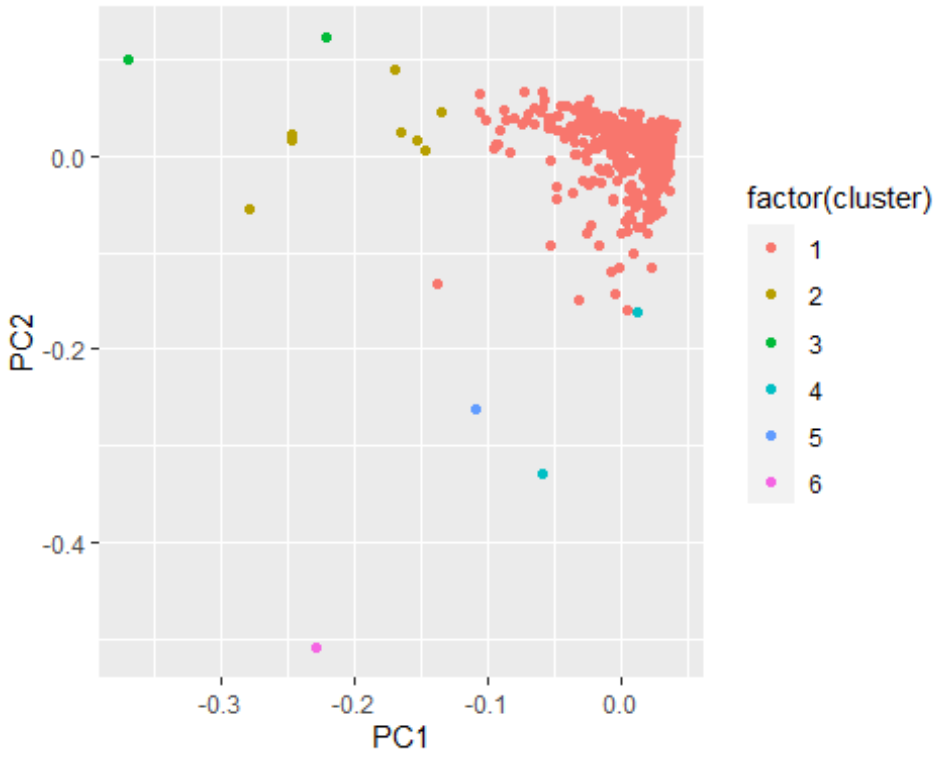

```
suppressPackageStartupMessages(library(dendextend))
avg_dend_obj <- as.dendrogram(hclust_)
avg_col_dend <- color_branches(avg_dend_obj, h = 6)
plot(avg_col_dend)
```




```
# Plot the clustering result with two features from PCA and the cluster label
from the clustering result.
suppressPackageStartupMessages(library(dplyr))

# Merge the cluster into reduce dataDBSCAN
seeds_df_cl <- mutate(reduced_data, cluster = cut_cluster)

# count how many observations were assigned to each cluster with the count()
function.
count(seeds_df_cl,cluster)
```

```
##   cluster   n
## 1       1 426
## 2       2   8
## 3       3   2
## 4       4   2
## 5       5   1
## 6       6   1
```

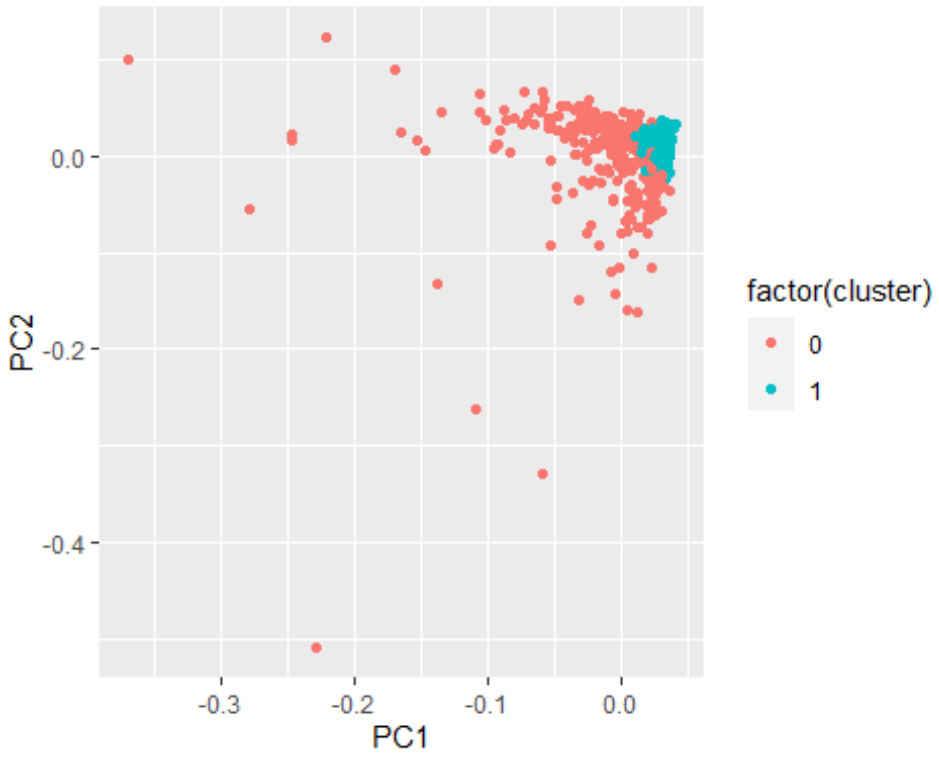
```
# Draw Blot
suppressPackageStartupMessages(library(ggplot2))
ggplot(seeds_df_cl, aes(x=PC1, y = PC2, color = factor(cluster))) +
geom_point()
```



Density-based clustering

Perform the cluster analysis using DBSCAN with ϵ = 0.5 and *minpts* = 15. For the input data, use the trans_data data. Plot the clusters with two features from PCA and the cluster label from DBSCAN clustering results.

```
# DBSCAN with eps= 0.5, MinPts = 15
dbscan_ <- dbscan(trans_data, eps= 0.5, MinPts = 15)
dbscan_cluster <-dbscan_[["cluster"]]

# Plot the clusters with two features from PCA
seeds_df_cl <- mutate(reduced_data, cluster = dbscan_cluster)

# count how many observations were assigned to each cluster with the count()
function.
count(seeds_df_cl,cluster)

##   cluster   n
## 1       0 270
## 2       1 170

# Draw plot
ggplot(seeds_df_cl, aes(x=PC1, y = PC2, color = factor(cluster))) +
geom_point()
```


0.0
-0.2
-0.4
PC2
-0.3
-0.2
-0.1
0.0
PC1
factor(cluster)
0
1