City Clustering base on city GDP and retail sales from 2010 to 2018

Fortuna Zhang

5/15/2020

```r
# https://uc-r.github.io/kmeans_clustering
# https://www.statsandr.com/blog/clustering-analysis-k-means-and-hierarchical-clustering-by-hand-and-in-r/#elbow-method
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
GDP = read.csv("Data/GDP中文.csv",skip = 3, header = T)
GDP = head(GDP,-1)
retail = read.csv("Data/retail中文.csv",skip = 3,header = T)
retail = head(retail,-1)
colnames(GDP) = gsub("X","",colnames(GDP))
colnames(retail) = gsub("X","",colnames(retail))

df = merge(GDP,retail,by.x = "地区",by.y = "地区")
# x is GDP, y is retail sales
df$`2019年.x` = NULL
df$`2019年.y` = NULL
df = data.frame(df[,-1], row.names = df[,1])
colnames(df) = gsub("X","",colnames(df))
colnames(df)
```

```
##  [1] "2018年.x" "2017年.x" "2016年.x" "2015年.x" "2014年.x" "2013年.x"
##  [7] "2012年.x" "2011年.x" "2010年.x" "2018年.y" "2017年.y" "2016年.y"
## [13] "2015年.y" "2014年.y" "2013年.y" "2012年.y" "2011年.y" "2010年.y"
```

```r
df =  replace(df, is.na(df), 0)
```

```r
GDP2 = read.csv("Data/GDP.csv",skip = 3, header = T)
GDP2 = head(GDP2,-1)
retail2 = read.csv("Data/retail.csv",skip = 3,header = T)
```

```r
retail2 = head(retail2,-1)
colnames(GDP2) = gsub("X","",colnames(GDP2))
colnames(retail2) = gsub("X","",colnames(retail2))
colnames(retail2)
```

```
##  [1] "City" "2019" "2018" "2017" "2016" "2015" "2014" "2013" "2012" "2011"
## [11] "2010"
```

```r
df2 = merge(GDP2,retail2,by.x = "City",by.y = "City")
# x is GDP, y is retail sales
df2$`2019.x` = NULL
df2$`2019.y` = NULL
df2 = data.frame(df2[,-1], row.names = df2[,1])
colnames(df2) = gsub("X","",colnames(df2))
colnames(df2)
```

```
##  [1] "2018.x" "2017.x" "2016.x" "2015.x" "2014.x" "2013.x" "2012.x"
"2011.x"
##  [9] "2010.x" "2018.y" "2017.y" "2016.y" "2015.y" "2014.y" "2013.y"
"2012.y"
## [17] "2011.y" "2010.y"
```

```r
df2 =  replace(df2, is.na(df2), 0)
```

Kendall correlation distance: Kendall correlation method measures the correspondence between the ranking of x and y variables.

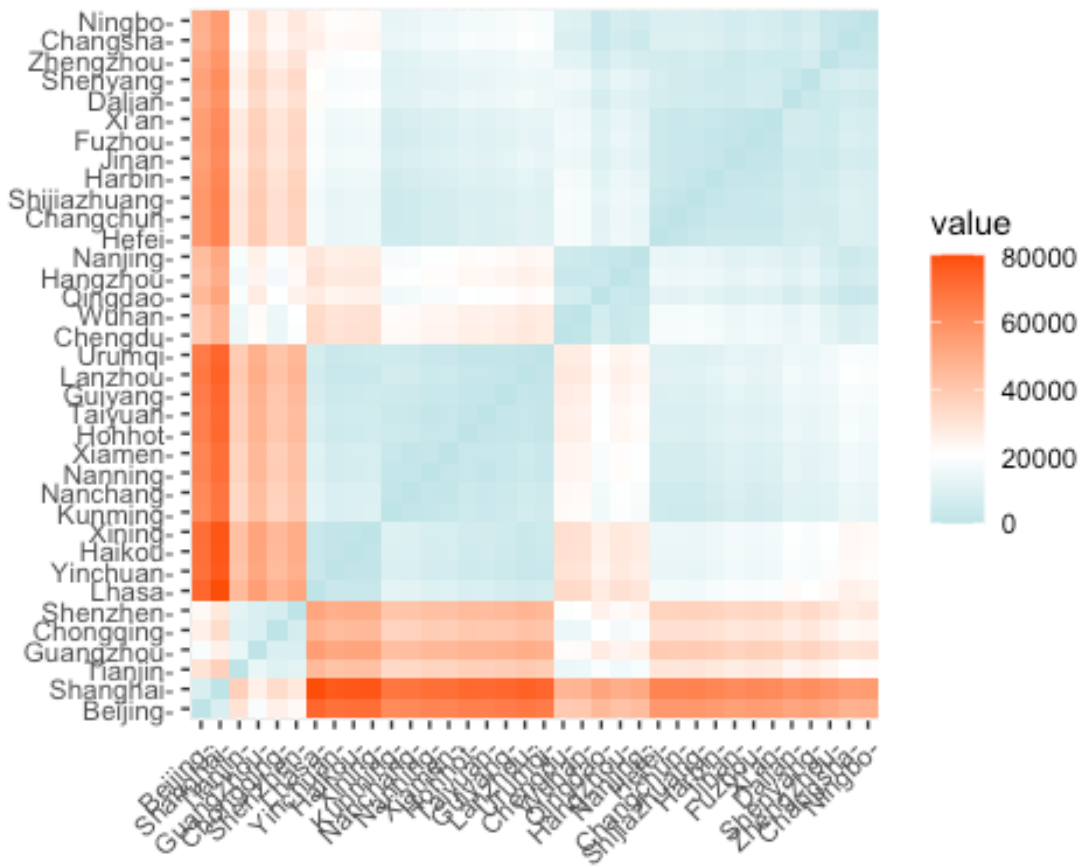
```r
#install.packages("factoextra")
# Install from CRAN
#install.packages("tidyverse")
library(tidyverse)  # data manipulation
```

```
## ── Attaching packages ───────────────────────────────────────────
tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.0     ✓ purrr   0.3.4
## ✓ tibble  3.0.1     ✓ stringr 1.4.0
## ✓ tidyr   1.0.3     ✓ forcats 0.5.0
## ✓ readr   1.3.1

## ── Conflicts ──────────────────────────────────────────────
tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(cluster)    # clustering algorithms
library(factoextra) # clustering algorithms & visualization
```

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

```r
distance = get_dist(df2)
fviz_dist(distance,gradient = list(low = "#00AFBB", mid = "white", high =
"#FC4E07"))
```



K-means Clustering: K-means clustering is the most commonly used unsupervised machine learning algorithm for partitioning a given data set into a set of k groups (i.e. k clusters), where k represents the number of groups pre-specified by the analyst. It classifies objects in multiple groups (i.e., clusters), such that objects within the same cluster are as similar as possible (i.e., high intra-class similarity), whereas objects from different clusters are as dissimilar as possible (i.e., low inter-class similarity). In k-means clustering, each cluster is represented by its center (i.e, centroid) which corresponds to the mean of points assigned to the cluster.

cluster: A vector of integers (from 1:k) indicating the cluster to which each point is allocated. centers: A matrix of cluster centers. totss: The total sum of squares. withinss: Vector of within-cluster sum of squares, one component per cluster. tot.withinss: Total within-cluster sum of squares, i.e. sum(withinss). betweenss: The between-cluster sum of squares, i.e. $totss - tot.withinss$. size: The number of points in each cluster.

```r
# English version clustering
# k-means clustering, target at 2 centroids, starts at 36 initial centroids
k2 <- kmeans(df2, centers = 2, nstart = 36)
k2
```

```
## K-means clustering with 2 clusters of sizes 30, 6
##
## Cluster means:
##      2018.x    2017.x    2016.x   2015.x    2014.x    2013.x   2012.x
2011.x
## 1  6892.579  6398.246  5746.908  5398.72  5044.526  4646.762  4203.77
3714.198
## 2 25191.807 23435.843 21418.967 19332.80 17932.792 16460.767 14810.98
13449.207
##     2010.x   2018.y   2017.y   2016.y   2015.y   2014.y   2013.y   2012.y
## 1  3084.67 3217.723 3064.047 2794.633 2529.797 2256.197 1972.140 1739.597
## 2 11459.90 8891.917 8770.317 8179.700 7526.100 6896.567 6378.617 5712.400
##     2011.y   2010.y
## 1 1509.340 1265.940
## 2 5058.217 4319.233
##
## Clustering vector:
##      Beijing    Changchun     Changsha      Chengdu    Chongqing
Dalian
##            2            1            1            1            2
1
##       Fuzhou    Guangzhou      Guiyang       Haikou     Hangzhou
Harbin
##            1            2            1            1            1
1
##        Hefei       Hohhot        Jinan      Kunming      Lanzhou
Lhasa
##            1            1            1            1            1
1
##     Nanchang      Nanjing      Nanning       Ningbo      Qingdao
Shanghai
##            1            1            1            1            1
2
##     Shenyang     Shenzhen Shijiazhuang      Taiyuan      Tianjin
Urumqi
##            1            2            1            1            2
1
##        Wuhan        Xi'an       Xiamen       Xining     Yinchuan
Zhengzhou
##            1            1            1            1            1
1
##
## Within cluster sum of squares by cluster:
## [1] 2813364028 1163499735
##  (between_SS / total_SS =  69.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
```

```
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# k-means clustering, target at 3 centroids, starts at 36 initial centroids
k3 <- kmeans(df2, centers = 3, nstart = 36)
k3
```

```
## K-means clustering with 3 clusters of sizes 13, 17, 6
##
## Cluster means:
##      2018.x    2017.x    2016.x    2015.x    2014.x    2013.x    2012.x
## 1  3150.632  2940.045  2669.328  2496.382  2313.938  2095.224  1882.154
## 2  9754.068  9042.752  8100.351  7618.155  7132.623  6597.938  5979.124
## 3 25191.807 23435.843 21418.967 19332.795 17932.792 16460.767 14810.983
##      2011.x    2010.x   2018.y   2017.y   2016.y   2015.y   2014.y
2013.y
## 1  1640.983  1336.785 1405.146 1364.969 1245.838 1119.531 1002.154
891.0538
## 2  5299.597  4421.288 4603.812 4363.341 3979.006 3608.235 3215.171
2798.8529
## 3 13449.207 11459.900 8891.917 8770.317 8179.700 7526.100 6896.567
6378.6167
##      2012.y    2011.y    2010.y
## 1  787.6308  683.9538  559.3846
## 2 2467.5706 2140.5176 1806.2471
## 3 5712.4000 5058.2167 4319.2333
##
## Clustering vector:
##      Beijing    Changchun     Changsha      Chengdu    Chongqing
Dalian
##            3            2            2            2            3
2
##       Fuzhou    Guangzhou      Guiyang       Haikou     Hangzhou
Harbin
##            2            3            1            1            2
2
##        Hefei       Hohhot        Jinan      Kunming      Lanzhou
Lhasa
##            2            1            2            1            1
1
##     Nanchang      Nanjing      Nanning       Ningbo      Qingdao
Shanghai
##            1            2            1            2            2
3
##     Shenyang     Shenzhen Shijiazhuang      Taiyuan      Tianjin
Urumqi
##            2            3            2            1            3
1
##        Wuhan        Xi'an       Xiamen       Xining     Yinchuan
Zhengzhou
```

```
##            2            2            1            1            1
2
##
## Within cluster sum of squares by cluster:
## [1]  167100852  674375589 1163499735
##  (between_SS / total_SS =  84.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(k3$cluster)
```

```
##
##  1  2  3
## 13 17  6
```

```
# k-means clustering, target at 4 centroids, starts at 36 initial centroids
k4 <- kmeans(df2, centers = 4, nstart = 36)
k4
```

```
## K-means clustering with 4 clusters of sizes 13, 4, 17, 2
##
## Cluster means:
##      2018.x    2017.x    2016.x    2015.x    2014.x    2013.x    2012.x
## 1  3150.632  2940.045  2669.328  2496.382  2313.938  2095.224  1882.154
## 2 20508.262 20491.783 18666.505 16964.682 15674.555 14286.410 12701.195
## 3  9754.068  9042.752  8100.351  7618.155  7132.623  6597.938  5979.124
## 4 34558.895 29323.965 26923.890 24069.020 22449.265 20809.480 19030.560
##      2011.x    2010.x    2018.y    2017.y    2016.y    2015.y   2014.y
## 1  1640.983  1336.785  1405.146  1364.969  1245.838  1119.531 1002.154
## 2 11311.905  9369.960  7233.775  7304.050  6781.625  6171.775 5609.475
## 3  5299.597  4421.288  4603.812  4363.341  3979.006  3608.235 3215.171
## 4 17723.810 15639.780 12208.200 11702.850 10975.850 10234.750 9470.750
##      2013.y    2012.y    2011.y    2010.y
## 1  891.0538  787.6308  683.9538  559.3846
## 2 5210.6500 4577.6250 3985.3250 3347.1250
## 3 2798.8529 2467.5706 2140.5176 1806.2471
## 4 8714.5500 7981.9500 7204.0000 6263.4500
##
## Clustering vector:
##      Beijing    Changchun     Changsha      Chengdu    Chongqing
Dalian
##            4            3            3            3            2
3
##       Fuzhou    Guangzhou      Guiyang       Haikou     Hangzhou
Harbin
##            3            2            1            1            3
3
```

```
##        Hefei       Hohhot        Jinan      Kunming      Lanzhou
Lhasa
##            3            1            3            1            1
1
##     Nanchang      Nanjing      Nanning       Ningbo      Qingdao
Shanghai
##            1            3            1            3            3
4
##     Shenyang     Shenzhen Shijiazhuang      Taiyuan      Tianjin
Urumqi
##            3            2            3            1            2
1
##        Wuhan        Xi'an       Xiamen       Xining     Yinchuan
Zhengzhou
##            3            3            1            1            1
3
##
## Within cluster sum of squares by cluster:
## [1] 167100852 148529262 674375589  29837739
##  (between_SS / total_SS =  92.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# k-means clustering, target at 5 centroids, starts at 36 initial centroids
k5 <- kmeans(df2, centers = 5, nstart = 36)
k5
```

```
## K-means clustering with 5 clusters of sizes 7, 2, 10, 13, 4
##
## Cluster means:
##      2018.x    2017.x    2016.x    2015.x    2014.x    2013.x    2012.x
## 1 12895.714 11861.869 10564.896  9613.079  8897.167  8114.866  7347.214
## 2 34558.895 29323.965 26923.890 24069.020 22449.265 20809.480 19030.560
## 3  7554.915  7069.371  6375.170  6221.709  5897.442  5536.088  5021.460
## 4  3150.632  2940.045  2669.328  2496.382  2313.938  2095.224  1882.154
## 5 20508.262 20491.783 18666.505 16964.682 15674.555 14286.410 12701.195
##      2011.x    2010.x    2018.y    2017.y    2016.y    2015.y   2014.y
## 1  6439.361  5367.629  5565.129  5294.057  4786.757  4298.529 3817.529
## 2 17723.810 15639.780 12208.200 11702.850 10975.850 10234.750 9470.750
## 3  4501.762  3758.850  3930.890  3711.840  3413.580  3125.030 2793.520
## 4  1640.983  1336.785  1405.146  1364.969  1245.838  1119.531 1002.154
## 5 11311.905  9369.960  7233.775  7304.050  6781.625  6171.775 5609.475
##      2013.y    2012.y    2011.y    2010.y
## 1 3308.1286 2897.8714 2523.0714 2136.1286
## 2 8714.5500 7981.9500 7204.0000 6263.4500
## 3 2442.3600 2166.3600 1872.7300 1575.3300
```

```
## 4  891.0538  787.6308  683.9538  559.3846
## 5 5210.6500 4577.6250 3985.3250 3347.1250
##
## Clustering vector:
##      Beijing    Changchun     Changsha      Chengdu    Chongqing
Dalian
##            2            3            1            1            5
3
##       Fuzhou    Guangzhou      Guiyang       Haikou     Hangzhou
Harbin
##            3            5            4            4            1
3
##        Hefei       Hohhot        Jinan      Kunming      Lanzhou
Lhasa
##            3            4            3            4            4
4
##     Nanchang      Nanjing      Nanning       Ningbo      Qingdao
Shanghai
##            4            1            4            1            1
2
##     Shenyang     Shenzhen Shijiazhuang      Taiyuan      Tianjin
Urumqi
##            3            5            3            4            5
4
##        Wuhan        Xi'an       Xiamen       Xining     Yinchuan
Zhengzhou
##            1            3            4            4            4
3
##
## Within cluster sum of squares by cluster:
## [1]  89875318  29837739  92626925 167100852 148529262
##  (between_SS / total_SS =  96.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Compare Total within-cluster sum of squares & Between-cluster sum of
squares across different k values
compare =
matrix(c(k2$tot.withinss,k3$tot.withinss,k4$tot.withinss,k5$tot.withinss,k2$b
etweenss,k3$betweenss,k4$betweenss,k5$betweenss),ncol=4,byrow=TRUE)
colnames(compare) = c('k2','k3','k4','k5')
rownames(compare) = c("Total within-cluster sum of squares","Between-cluster
sum of squares")
comparetable = as.table(compare)
comparetable
```

```
##                                                k2          k3          k4
## Total within-cluster sum of squares  3976863763  2004976175  1019843441
## Between-cluster sum of squares       9226948191 11198835779 12183968514
##                                                k5
## Total within-cluster sum of squares   527970095
## Between-cluster sum of squares      12675841860
```

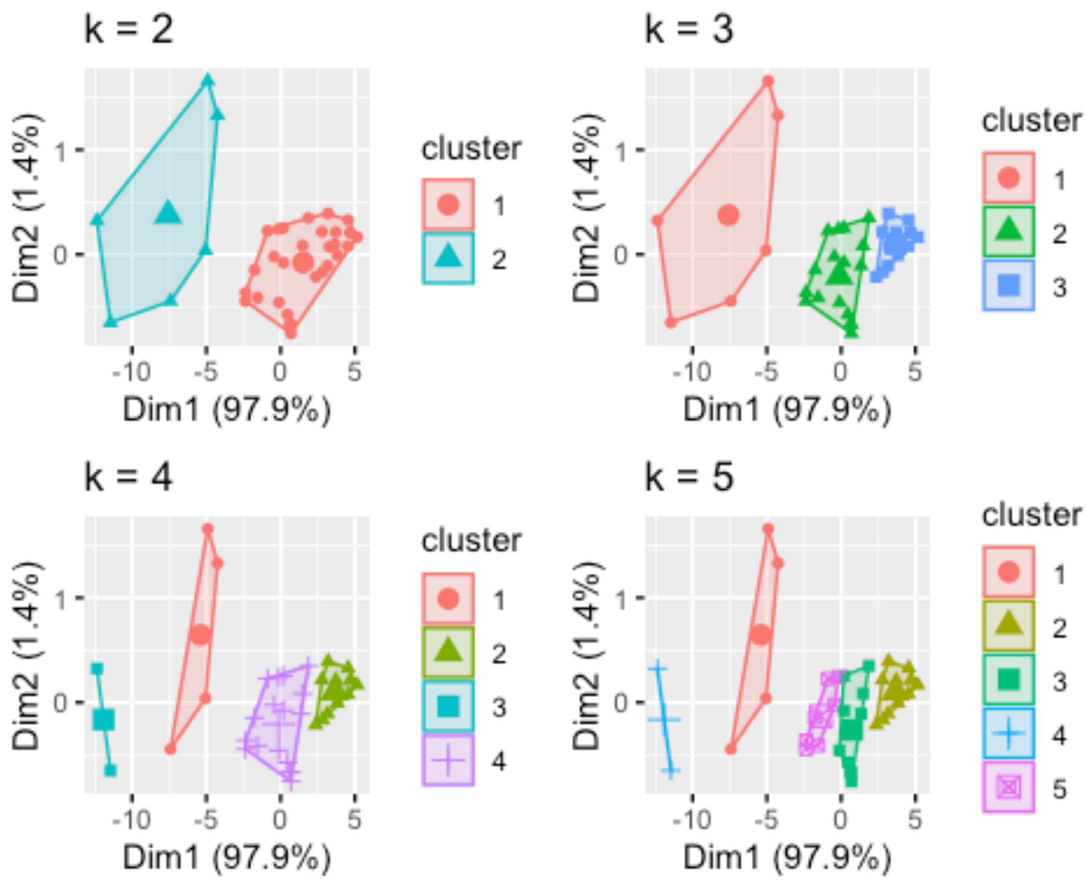
Compare different k values:

```
k2 <- kmeans(df2, centers = 2, nstart = 36)
k3 <- kmeans(df2, centers = 3, nstart = 36)
k4 <- kmeans(df2, centers = 4, nstart = 36)
k5 <- kmeans(df2, centers = 5, nstart = 36)

# plots to compare
p1 <- fviz_cluster(k2, geom = "point", data = df2) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = df2) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = df2) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = df2) + ggtitle("k = 5")

library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

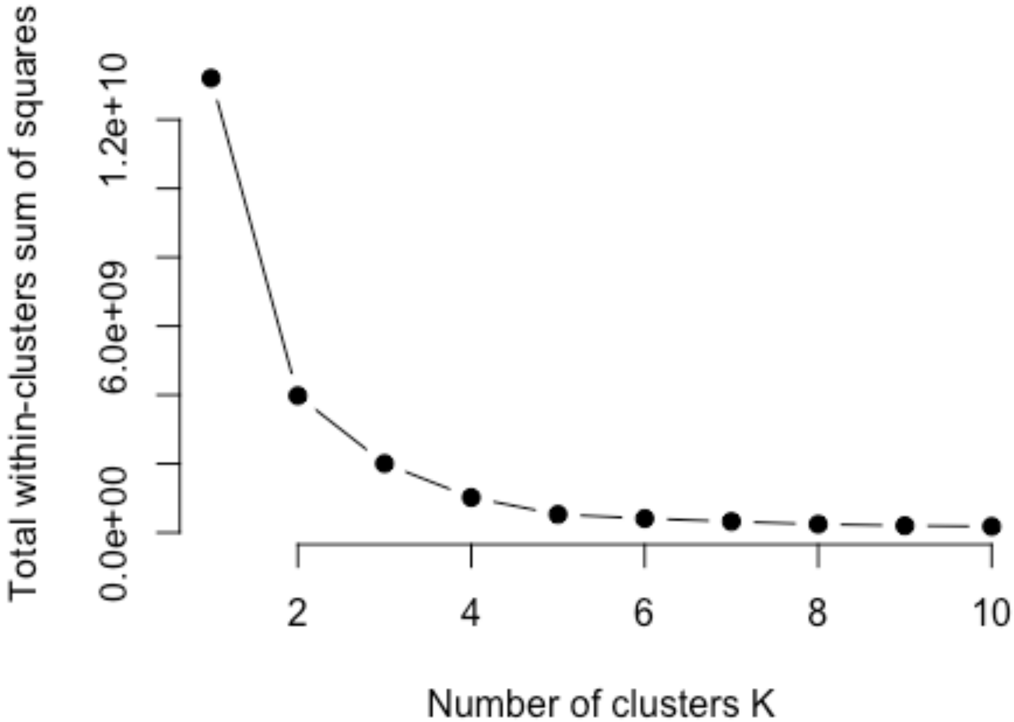
```
grid.arrange(p1, p2, p3, p4, nrow = 2)
```



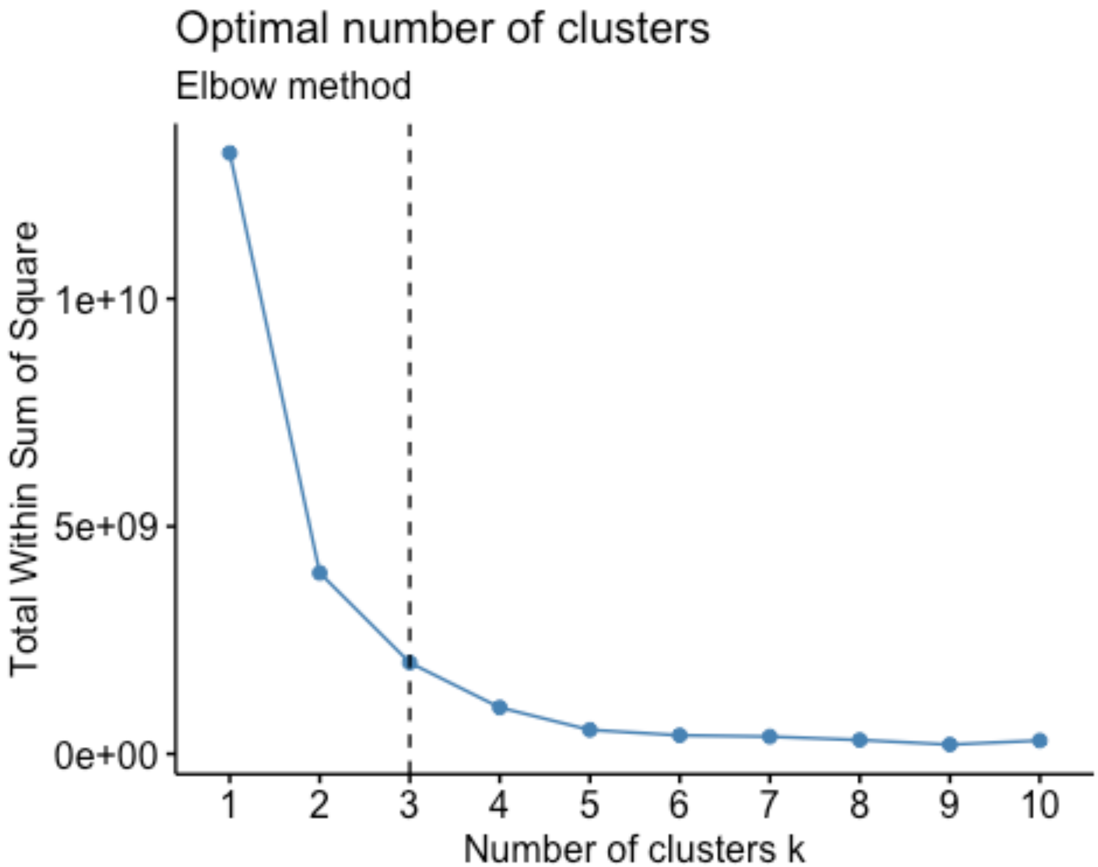
We can find large change of clustering pattern when k reduces from 4 to 3.This shows 3 can be the optimal k value.

Use three methods to select the optimal k value: Elbow method, Average Silhouette Method, and Gap Statistic Method.

```
# Elbow method
set.seed(123)
wss <- function(k) {
  kmeans(df2, k, nstart = 36 )$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:10

# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)

plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```

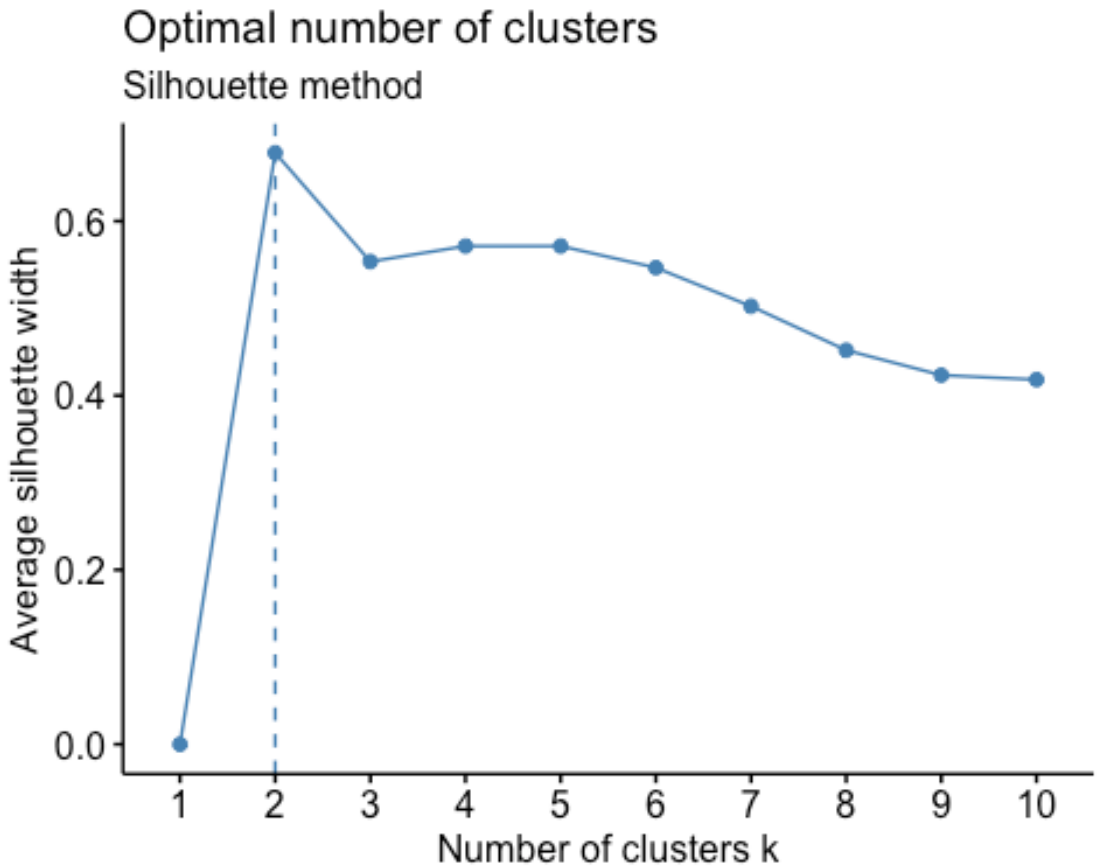



```
fviz_nbclust(df2, kmeans, method = "wss") + geom_vline(xintercept = 3,
linetype = 2)+
  labs(subtitle = "Elbow method")
```

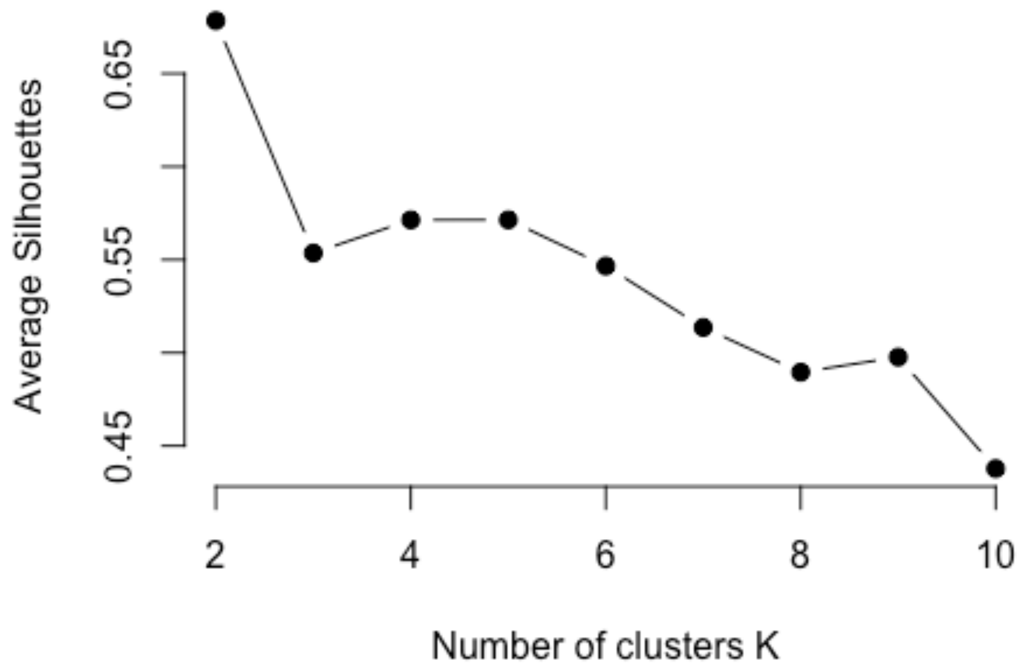


Elbow method shows that optimal k value is 4.

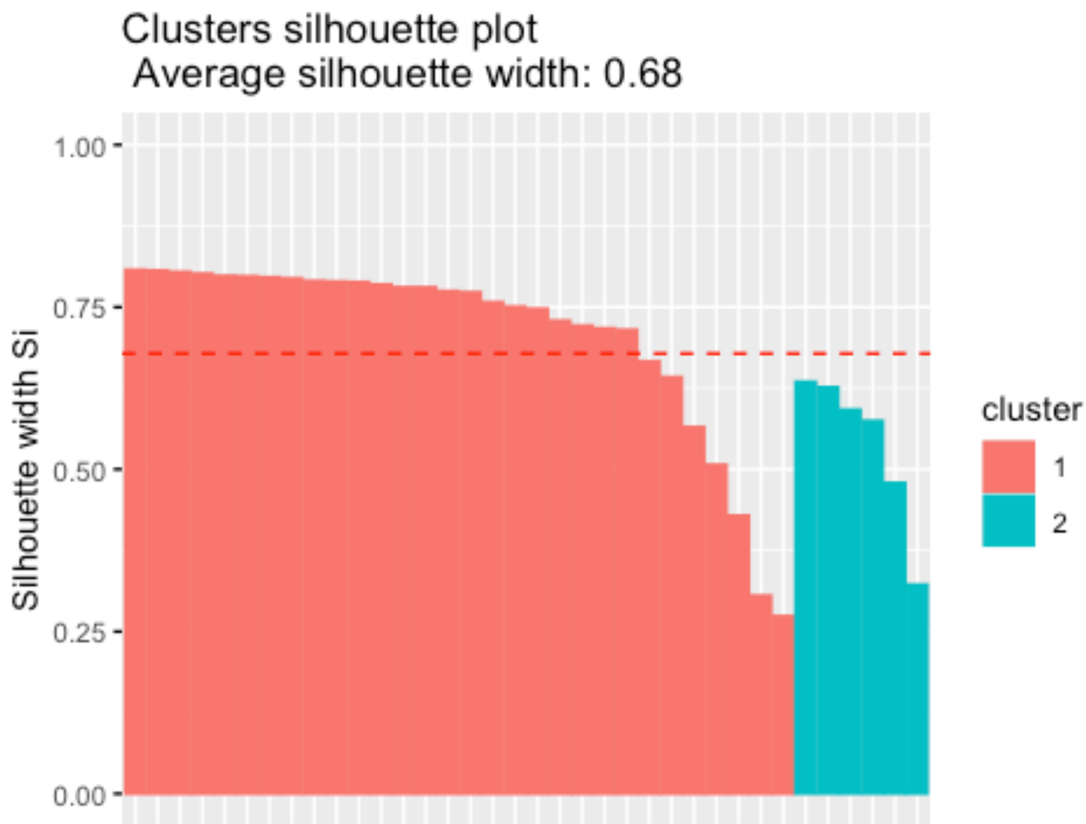
```
# Average Silhouette Method
fviz_nbclust(df2, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```




```
avg_sil <- function(k) {
  km.res <- kmeans(df2, centers = k, nstart = 36)
  ss <- silhouette(km.res$cluster, dist(df2))
  mean(ss[, 3])
}

# Compute and plot wss for k = 2 to k = 15
k.values <- 2:10

# extract avg silhouette for 2-15 clusters
avg_sil_values <- map_dbl(k.values, avg_sil)

plot(k.values, avg_sil_values,
       type = "b", pch = 19, frame = FALSE,
       xlab = "Number of clusters K",
       ylab = "Average Silhouettes")
```




```
sil <- silhouette(k2$cluster, dist(df2))
fviz_silhouette(sil)

##   cluster size ave.sil.width
## 1       1   30          0.71
## 2       2    6          0.54
```



Average

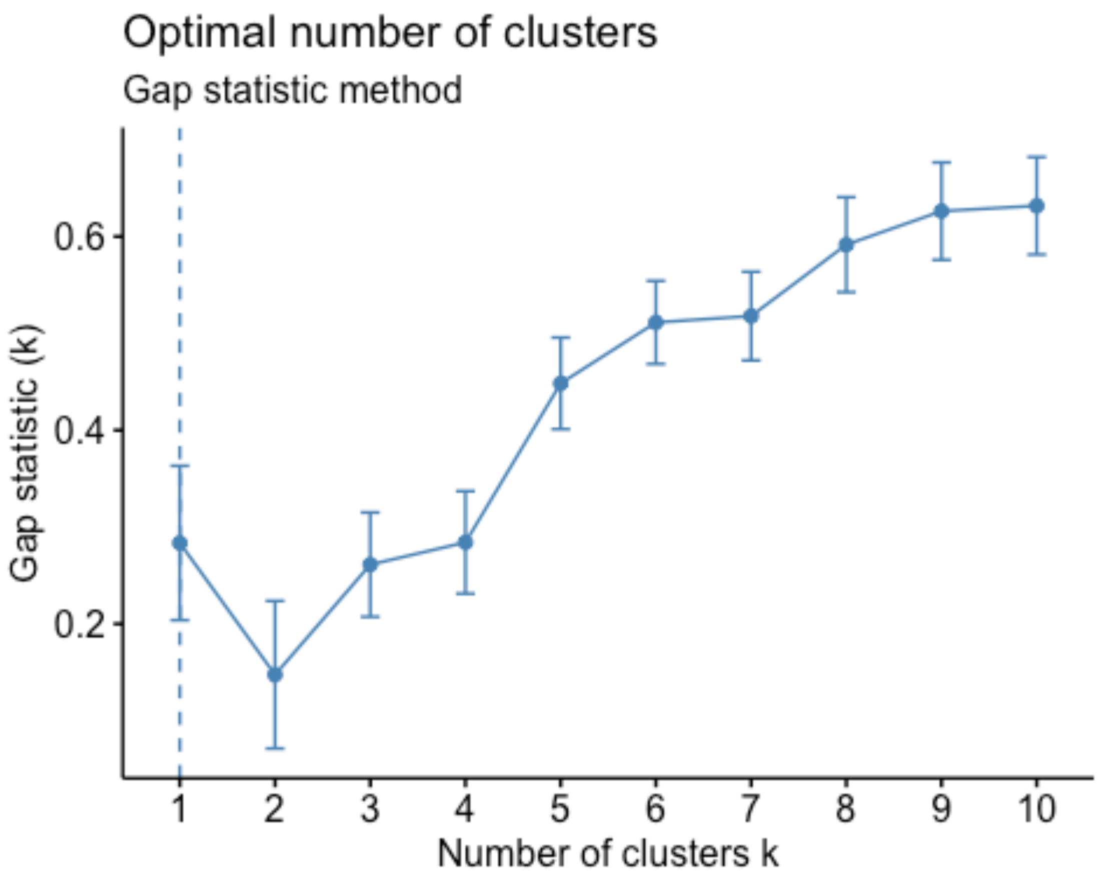
Silhouette method shows that optimal k value is 2.

```
# Gap Statistic Method
# compute gap statistic
set.seed(123)
gap_stat <- clusGap(df2, FUN = kmeans, nstart = 36,
                    K.max = 10, B = 50,d.power = 1)
# Print the result
print(gap_stat, method = "firstmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = df2, FUNcluster = kmeans, K.max = 10, B = 50, d.power = 1,
nstart = 36)
## B=50 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'firstmax'): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 12.117711 12.40119 0.2834748 0.07969861
##  [2,] 11.597149 11.74457 0.1474237 0.07623086
##  [3,] 11.147213 11.40836 0.2611470 0.05384942
##  [4,] 10.909490 11.19361 0.2841241 0.05292430
##  [5,] 10.587718 11.03606 0.4483396 0.04732062
##  [6,] 10.410050 10.92130 0.5112519 0.04293855
##  [7,] 10.302618 10.82046 0.5178411 0.04575513
##  [8,] 10.140022 10.73161 0.5915875 0.04904522
```

```
##  [9,] 10.020220 10.64641 0.6261901 0.05028436
## [10,]  9.936624 10.56834 0.6317128 0.05037724
```

```r
fviz_gap_stat(gap_stat)+
  labs(subtitle = "Gap statistic method")
```

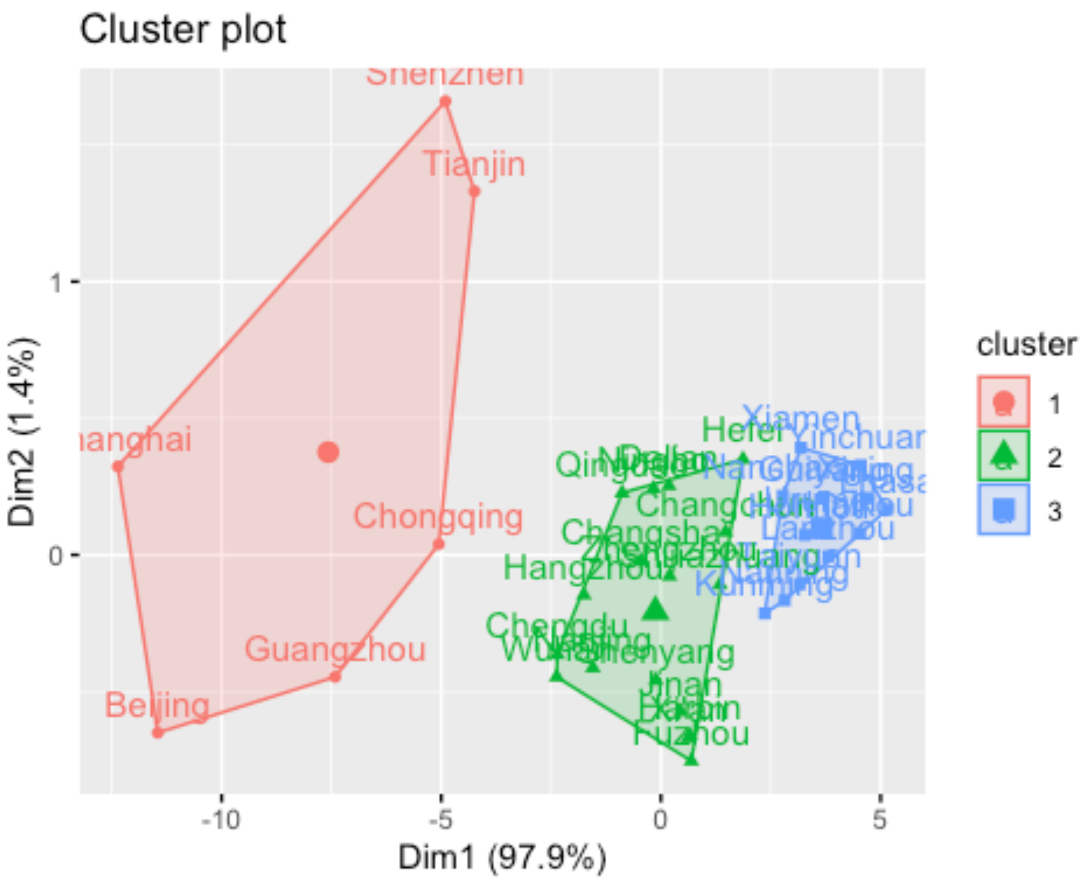


Gap Statistic Method shows that optimal k value is 1.

In conclusion, we should choose 3 as the optimal k value, so there are 3 clusters of cities base on city GDP and retail sales from 2010 to 2018.

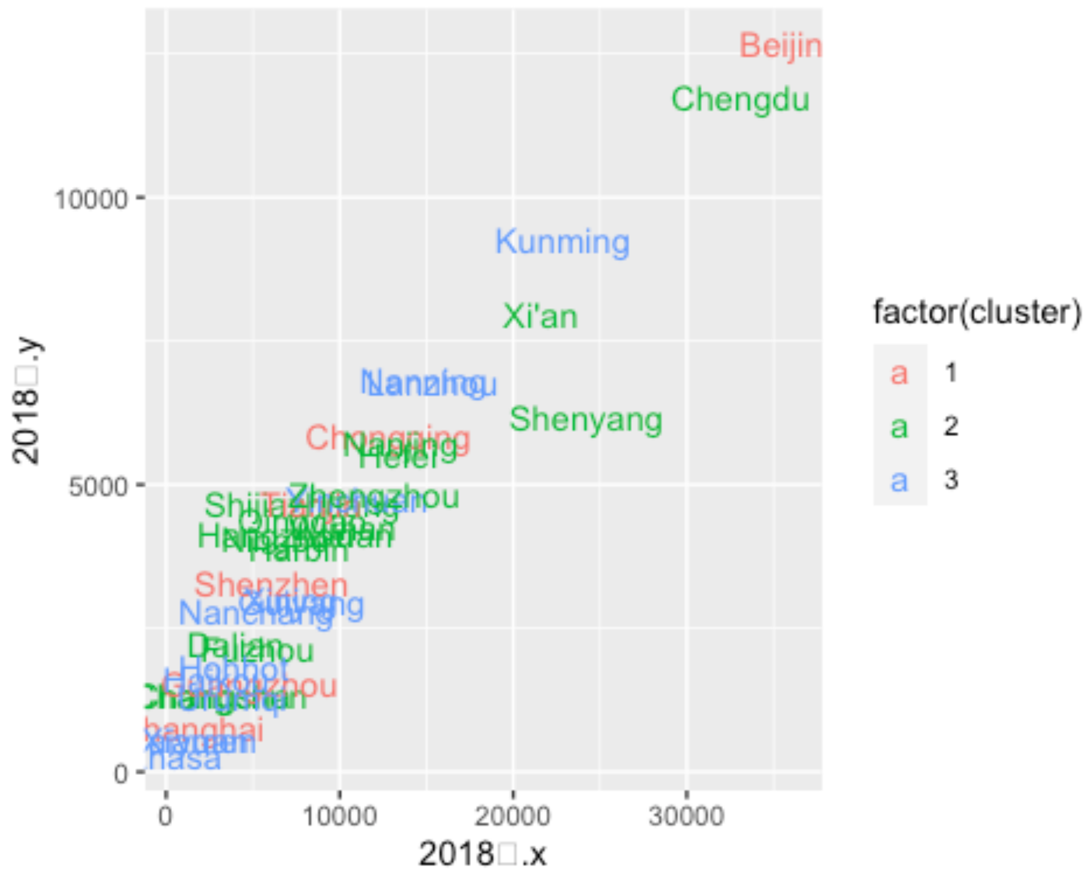
Clustering Visualizations: Principal component analysis (PCA) Visualization: plot the data points according to the first two principal components that explain the majority of the variance

```r
fviz_cluster(k3, data = df2)
```



Pairwise scatter plots:

```
df %>%
  as_tibble() %>%
  mutate(cluster = k3$cluster,
         city = row.names(df2)) %>%
  ggplot(aes(`2018年.x`, `2018年.y`, color = factor(cluster), label = city))
+
  geom_text()
```



City

Cluster Results Summary:

```
# Chinese version clustering
# k-means clustering, target at 3 centroids, starts at 36 initial centroids
k3chi <- kmeans(df, centers = 3, nstart = 36)
table(k3chi$cluster)

##
##  1  2  3
## 17 13  6

k3chi$cluster

##      上海 乌鲁木齐     兰州     北京     南京     南宁     南昌     厦门
##        3        2        2        3        1        2        2        2
##      合肥 呼和浩特   哈尔滨     大连     天津     太原     宁波     广州
##        1        2        1        1        3        2        1        3
##      成都     拉萨     昆明     杭州     武汉     沈阳     济南     海口
##        1        2        2        1        1        1        1        2
##      深圳   石家庄     福州     西宁     西安     贵阳     郑州     重庆
##        3        1        1        2        1        2        1        3
##      银川     长春     长沙     青岛
##        2        1        1        1
```

City clustering results are shown as above. In total there are 6 cities in the first cluster,17 cities in the second cluster, and 13 cities in the third cluster.