Clustering

Lisa Skalon

6/12/2020

```
library('ggplot2')
library('ggpubr')
library('dplyr')
library('tidyr')
library(class)
library(psych)
library(reshape2)
library(boot)
library(tidyverse)
library(cluster)
library(factoextra)
library(cluster.datasets)
set.seed(42)
```

We are going to analyze dataset cake.ingredients.1961 from package cluster.datasets. The dataset identifies for each cake which ingredient is used and the quantity.

```
# data cleaning
data("cake.ingredients.1961")
df <- cake.ingredients.1961
str(df)
```

```
## 'data.frame':    18 obs. of  35 variables:
##  $ Cake: chr  "Angel                " "Babas au Rhum        " "Sweet Chocolate      " "Buche de Noel
##  $ AE  : num  0.25 NA NA NA NA NA NA NA NA NA ...
##  $ BM  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ BP  : num  NA NA NA NA NA NA NA 2 NA NA ...
##  $ BR  : num  NA 0.25 1 NA 0.25 NA NA NA NA NA ...
##  $ BS  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ CA  : num  NA NA NA NA NA NA NA 10 0.3 NA ...
##  $ CC  : num  NA NA NA NA 1.5 1.5 0.5 NA NA NA ...
##  $ CE  : num  NA NA 4 NA NA NA NA NA NA 3 ...
##  $ CI  : num  NA NA NA NA NA NA 1 NA NA NA ...
##  $ CS  : num  NA NA NA NA NA NA 0.3 NA NA NA ...
##  $ CT  : num  1.25 NA NA NA NA NA NA NA NA NA ...
##  $ DC  : num  NA 3 NA NA NA NA NA NA NA NA ...
##  $ EG  : num  NA 1 4 4 6 2 NA 2 2 2 ...
##  $ EY  : num  NA 2 NA NA NA NA 2 NA NA NA ...
##  $ EW  : num  10 NA NA NA NA NA NA NA NA NA ...
##  $ FR  : num  1 1.75 2.5 1 0.25 NA NA 2 1.75 2 ...
##  $ GN  : num  NA NA NA NA NA 2 1 NA NA NA ...
```

```
##  $ HC  : num  NA NA NA NA NA 1 NA NA NA NA ...
##  $ LJ  : num  NA NA NA NA 1 1 1 NA NA NA ...
##  $ LR  : num  NA 0.5 NA NA 1 1 1 NA NA NA ...
##  $ MK  : num  NA 0.25 1 NA NA 1 NA 0.7 1 NA ...
##  $ NG  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ NS  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ RM  : num  NA 2 NA NA NA 1 NA NA NA NA ...
##  $ SA  : num  NA NA 1 NA NA NA NA 0.5 1.25 1 ...
##  $ SC  : num  NA NA NA NA 1 NA 1 NA NA 1 ...
##  $ SG  : num  NA NA NA NA NA NA NA 10 0.5 0.3 ...
##  $ SR  : num  1.5 0.25 2 1.3 1 1 NA 1.5 1.5 1.5 ...
##  $ SS  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ ST  : num  0.25 NA 0.5 0.5 0.25 0.25 NA 0.25 1 1 ...
##  $ VE  : num  1 NA 1 1 NA NA NA 1 1 1 ...
##  $ WR  : num  NA 0.25 0.5 NA NA NA 0.5 0.5 NA 0.25 ...
##  $ YT  : num  NA 0.6 NA NA NA NA NA NA NA NA ...
##  $ ZH  : num  NA NA NA NA 6 NA NA NA NA NA ...
```

```
df[is.na(df)] <- 0
summary(df)
```

```
##      Cake                AE              BM          BP
##  Length:18          Min.   :0.000   Min.   :0   Min.   :0.0000
##  Class :character   1st Qu.:0.000   1st Qu.:0   1st Qu.:0.0000
##  Mode  :character   Median :0.000   Median :0   Median :0.0000
##                     Mean   :0.125   Mean   :0   Mean   :0.9306
##                     3rd Qu.:0.000   3rd Qu.:0   3rd Qu.:1.7500
##                     Max.   :1.500   Max.   :0   Max.   :4.0000
##        BR               BS               CA                CC
##  Min.   :0.0000   Min.   :0.0000   Min.   : 0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median : 0.0000   Median :0.0000
##  Mean   :0.2178   Mean   :0.1667   Mean   : 0.5722   Mean   :0.2333
##  3rd Qu.:0.2500   3rd Qu.:0.0000   3rd Qu.: 0.0000   3rd Qu.:0.1500
##  Max.   :1.0000   Max.   :3.0000   Max.   :10.0000   Max.   :1.5000
##        CE               CI                CS               CT
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.0000   Median :0.00000   Median :0.0000   Median :0.00000
##  Mean   :0.3889   Mean   :0.05556   Mean   :0.1833   Mean   :0.06944
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :4.0000   Max.   :1.00000   Max.   :3.0000   Max.   :1.25000
##        DC               EG               EY               EW
##  Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   : 0.0000
##  1st Qu.:0.0000   1st Qu.:1.250   1st Qu.:0.0000   1st Qu.: 0.0000
##  Median :0.0000   Median :2.500   Median :0.0000   Median : 0.0000
##  Mean   :0.1667   Mean   :2.611   Mean   :0.2222   Mean   : 0.5556
##  3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.:0.0000   3rd Qu.: 0.0000
##  Max.   :3.0000   Max.   :6.000   Max.   :2.0000   Max.   :10.0000
##        FR               GN               HC               LJ
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.750   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.625   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :1.403   Mean   :0.1667   Mean   :0.2222   Mean   :0.1667
```

```
##  3rd Qu.:2.000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :3.000   Max.   :2.0000   Max.   :2.0000   Max.   :1.0000
##        LR              MK               NG               NS
##  Min.   :0.000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.000   Median :0.1250   Median :0.00000   Median :0.00000
##  Mean   :0.250   Mean   :0.3722   Mean   :0.08333   Mean   :0.05556
##  3rd Qu.:0.375   3rd Qu.:0.9250   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.000   Max.   :1.0000   Max.   :1.50000   Max.   :1.00000
##        RM              SA               SC               SG
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median : 0.0000
##  Mean   :0.1667   Mean   :0.2083   Mean   :0.2222   Mean   : 0.6556
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.: 0.0000
##  Max.   :2.0000   Max.   :1.2500   Max.   :1.0000   Max.   :10.0000
##        SR              SS               ST               VE
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.6375   1st Qu.:0.0000   1st Qu.:0.2500   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000   Median :0.5000   Median :0.5000
##  Mean   :1.0322   Mean   :0.1111   Mean   :0.4722   Mean   :0.6111
##  3rd Qu.:1.5000   3rd Qu.:0.0000   3rd Qu.:0.6875   3rd Qu.:1.0000
##  Max.   :2.0000   Max.   :1.0000   Max.   :1.0000   Max.   :2.0000
##        WR              YT               ZH
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.0000   Median :0.00000   Median :0.0000
##  Mean   :0.2361   Mean   :0.08889   Mean   :0.3333
##  3rd Qu.:0.2500   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :2.0000   Max.   :1.00000   Max.   :6.0000
```

```r
row.names(df) <- df$Cake
df <- df[,-1]
df <- scale(df)
df <- df[,-2]
```

Firstly we will try to claster the dataset with k-means method.

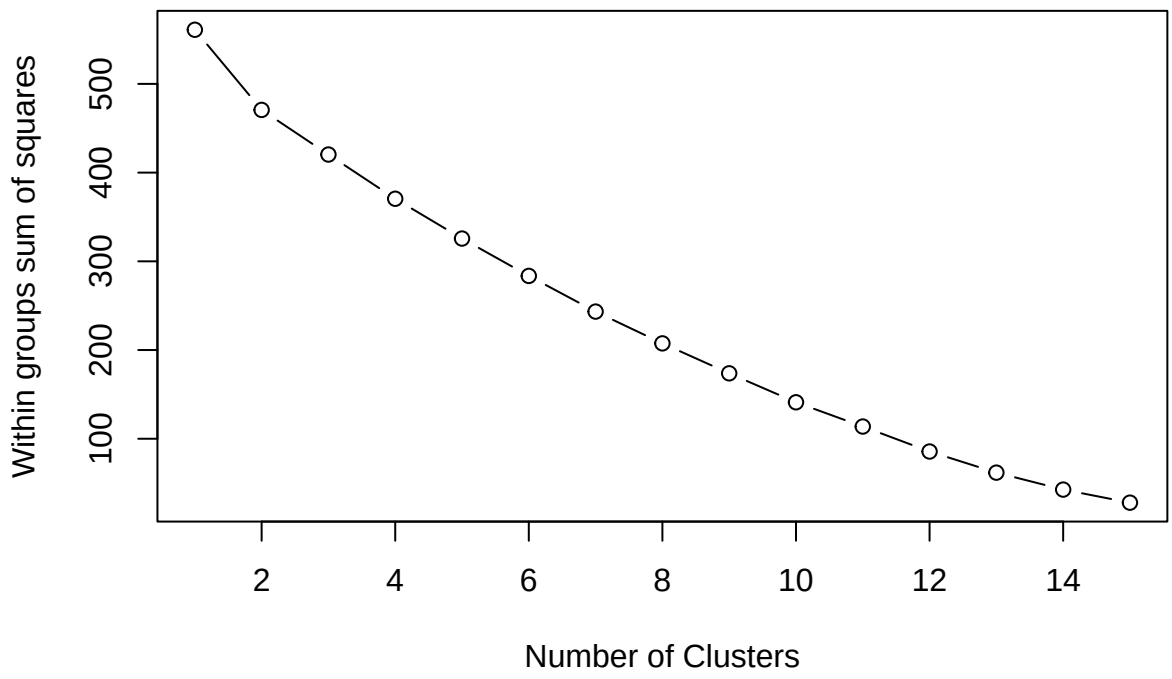
K-means clustering is unsupervised machine learning algorithm for partitioning a given data set into a set of k groups, k is tuned hyperparameter. It classifies objects in multiple groups, such that objects within the same cluster are as similar as possible (i.e., high intra-class similarity), whereas 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.

A plot of the within groups sum of squares by number of clusters extracted can help determine the appropriate number of clusters. An “Elbow method” - a method to determine the proper number of clusters.

Increasing the number of clusters will explain more of the variation, since there are more clusters to use, but that at some point this is over-fitting, and the elbow reflects this. The idea is that the first clusters will add much information (explain a lot of variation), since the data actually consist of that many groups (so these clusters are necessary), but once the number of clusters exceeds the actual number of groups in the data, the added information will drop sharply, because it is just subdividing the actual groups.

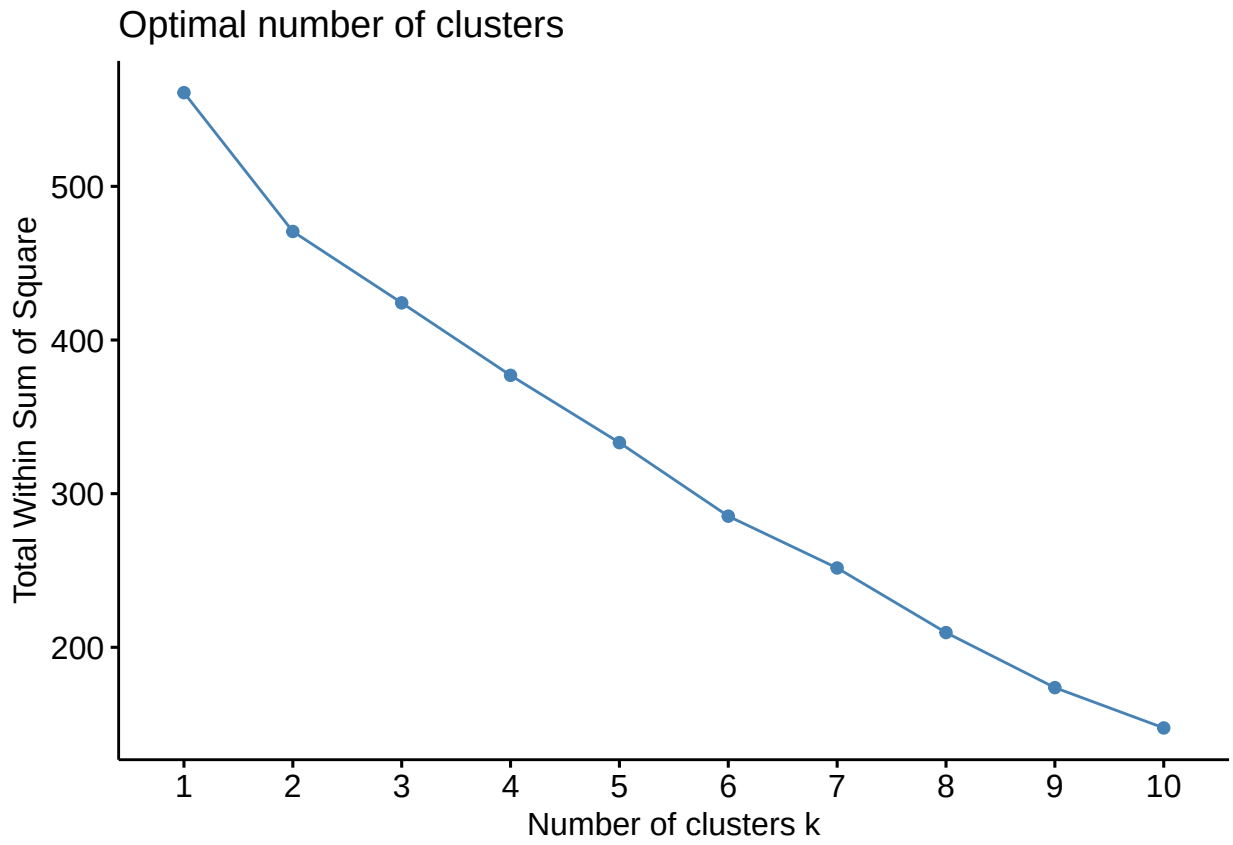
```r
# Determine number of clusters
wss <- (nrow(df)-1)*sum(apply(df,2,var))
```

```r
for (i in 2:15) wss[i] <- sum(kmeans(df,
   centers=i, nstart = 20)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
  ylab="Within groups sum of squares")
```



We can use build-in function as well

```r
fviz_nbclust(df, kmeans, method = "wss")
```



The algorithm starts by randomly selecting k objects from the data set to serve as the initial centers for the clusters. The selected objects are also known as cluster means or centroids. Next, each of the remaining objects is assigned to it's closest centroid, where closest is defined using the Euclidean distance between the object and the cluster mean.

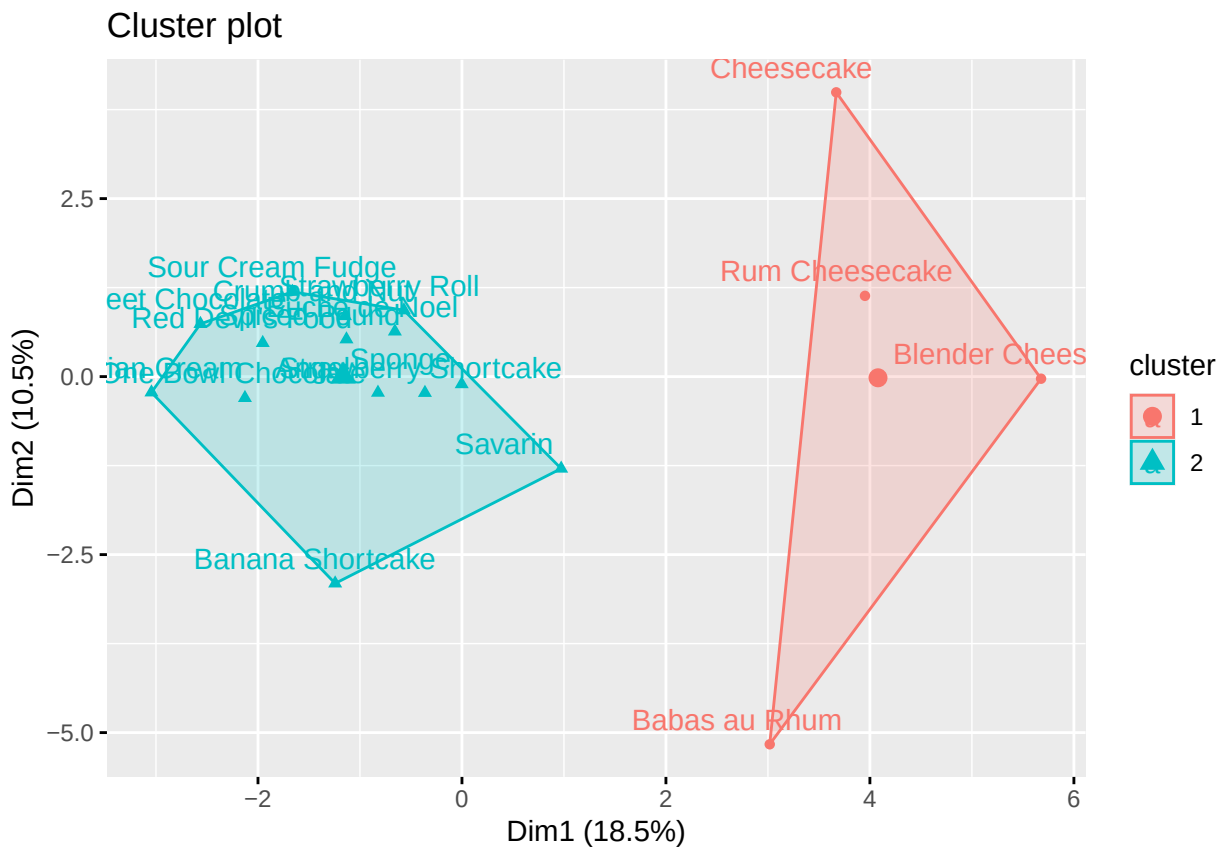
totss: The total sum of squares. withinss: Vector of within-cluster sum of squares, one component per cluster. tot.withinss: Total within-cluster sum of squares, i.e. sum(withinss). betweenss: The between-cluster sum of squares, i.e. $totss - tot.withinss$.

```
# K-Means Cluster Analysis
k2 <- kmeans(df, 2, nstart = 20) # 2 cluster solution
k2
```

```
## K-means clustering with 2 clusters of sizes 4, 14
##
## Cluster means:
##            AE         BP          BR         BS          CA         CC
## 1 -0.34123065 -0.6681436 -0.26175632 -0.2357023 -0.24309313  1.3130187
## 2  0.09749447  0.1908982  0.07478752  0.0673435  0.06945518 -0.3751482
##            CE         CI          CS         CT         DC          EG
## 1 -0.33971448  0.8249579 -0.15334183 -0.2357023  0.8249579 -0.20209440
## 2  0.09706128 -0.2357023  0.04381195  0.0673435 -0.2357023  0.05774126
##           EY         EW         FR         GN          HC         LJ         LR
## 1  1.2025725 -0.2357023 -0.9193578  1.1337962  0.05065990  1.5211472  1.4577380
## 2 -0.3435921  0.0673435  0.2626737 -0.3239418 -0.01447426 -0.4346135 -0.4164966
##            MK         NG          NS         RM          SA         SC         SG
## 1 -0.13411393 -0.2357023 -0.2357023  1.1337962 -0.4933696  0.6493281 -0.2793738
```

```
## 2  0.03831827  0.0673435  0.0673435 -0.3239418  0.1409627 -0.1855223  0.0798211
##           SR          SS         ST         VE          WR          YT
## 1 -0.8201918 -0.34359214 -0.9853180 -0.8757654 -0.10105116  0.22832876
## 2  0.2343405  0.09816918  0.2815194  0.2502187  0.02887176 -0.06523679
##           ZH
## 1  0.8249579
## 2 -0.2357023
##
## Clustering vector:
## Angel                 Babas au Rhum         Sweet Chocolate
##                     2                     1                     2
## Buche de Noel         Cheesecake            Rum Cheesecake
##                     2                     1                     1
## Blender Cheesecake    One Bowl Chocolate    Red Devil's Food
##                     1                     2                     2
## Sour Cream Fudge      Hungarian Cream       Crumb and Nut
##                     2                     2                     2
## Spiced Pound          Strawberry Roll       Savarin
##                     2                     2                     2
## Banana Shortcake      Strawberry Shortcake  Sponge
##                     2                     2                     2
##
## Within cluster sum of squares by cluster:
## [1] 112.1534 358.4889
##  (between_SS / total_SS =  16.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

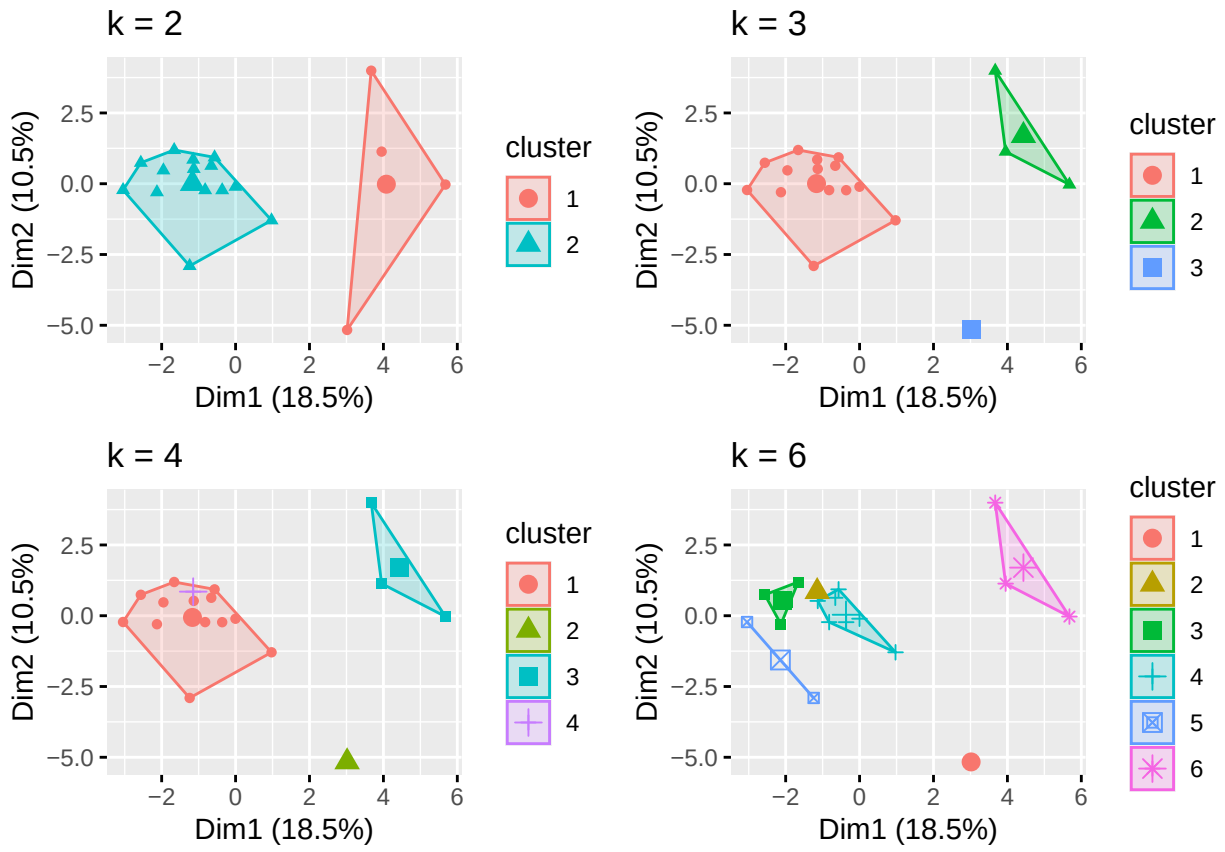
```
fviz_cluster(k2, data=df)
```




```
k3 <- kmeans(df, centers = 3, nstart = 20)
k4 <- kmeans(df, centers = 4, nstart = 20)
k6 <- kmeans(df, centers = 6, nstart = 20)

# plots to compare
p1 <- fviz_cluster(k2, geom = "point", data = df) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = df) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = df) + ggtitle("k = 4")
p4 <- fviz_cluster(k6, geom = "point",  data = df) + ggtitle("k = 6")

ggarrange(p1, p2, p3, p4, nrow = 2, ncol=2)
```



Hierarchical clustering

The key operation in hierarchical agglomerative clustering is to repeatedly combine the two nearest clusters into a larger cluster.

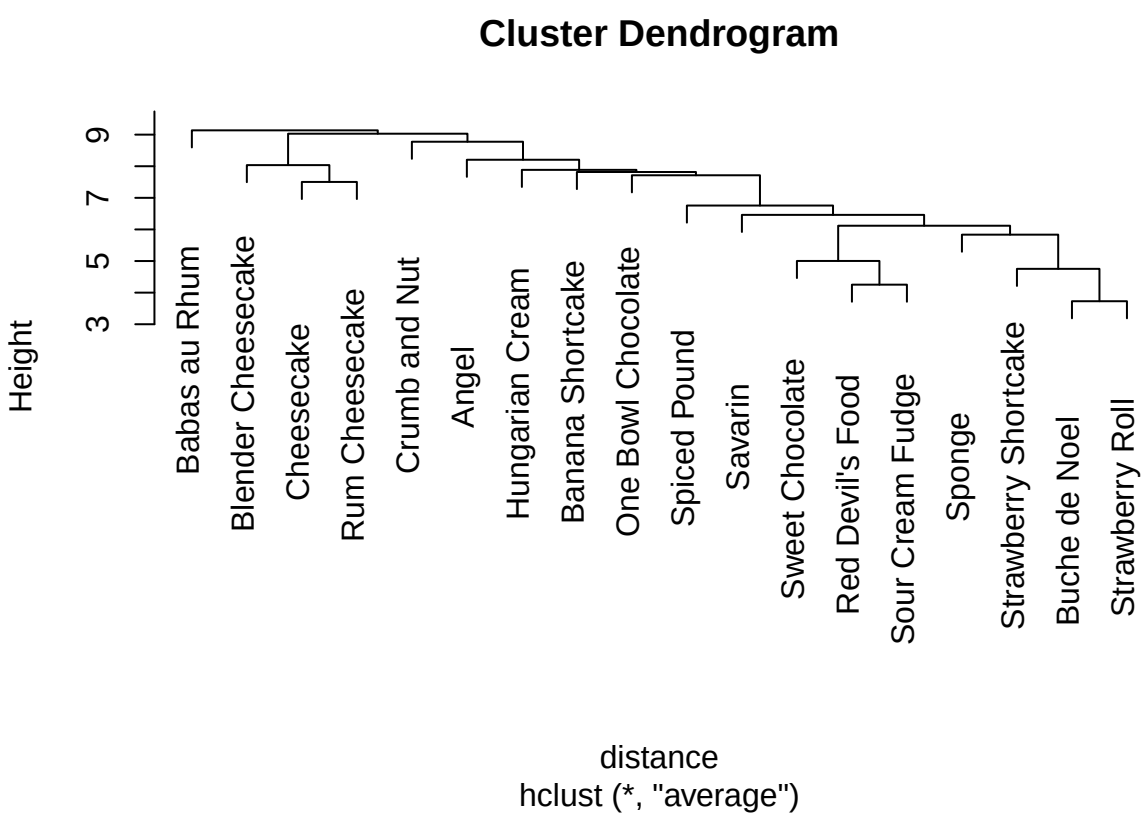
It starts by calculating the distance between every pair of observation points and store it in a distance matrix. It then puts every point in its own cluster. Then it starts merging the closest pairs of points based on the distances and the amount of clusters goes down by 1. Then it recomputes the distance between the new cluster and the old ones and stores them in a new distance matrix. Lastly it repeats steps 2 and 3 until all the clusters are merged into one single cluster.

Linkage Methods = ways to measure the distance between clusters:

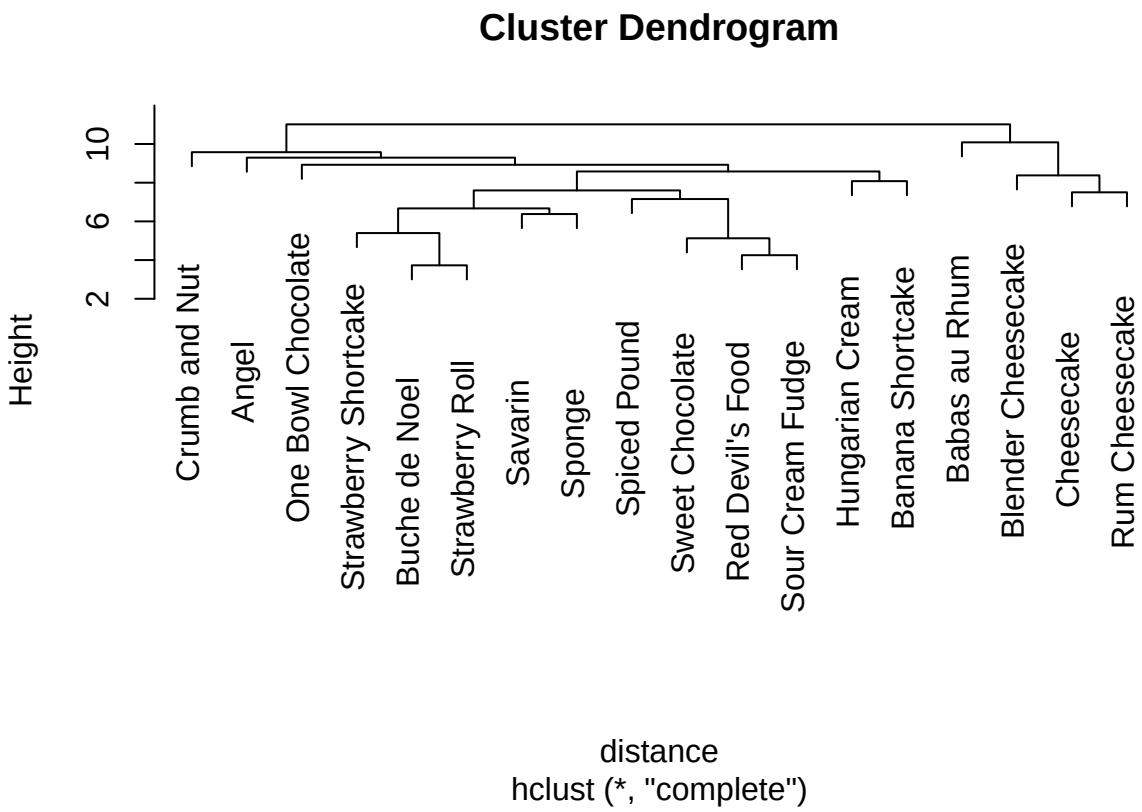
Complete-linkage: calculates the maximum distance between clusters before merging. Single-linkage: calculates the minimum . . . //. . . Average-linkage: calculates the average . . . //,..

```
distance <- dist(df)
```

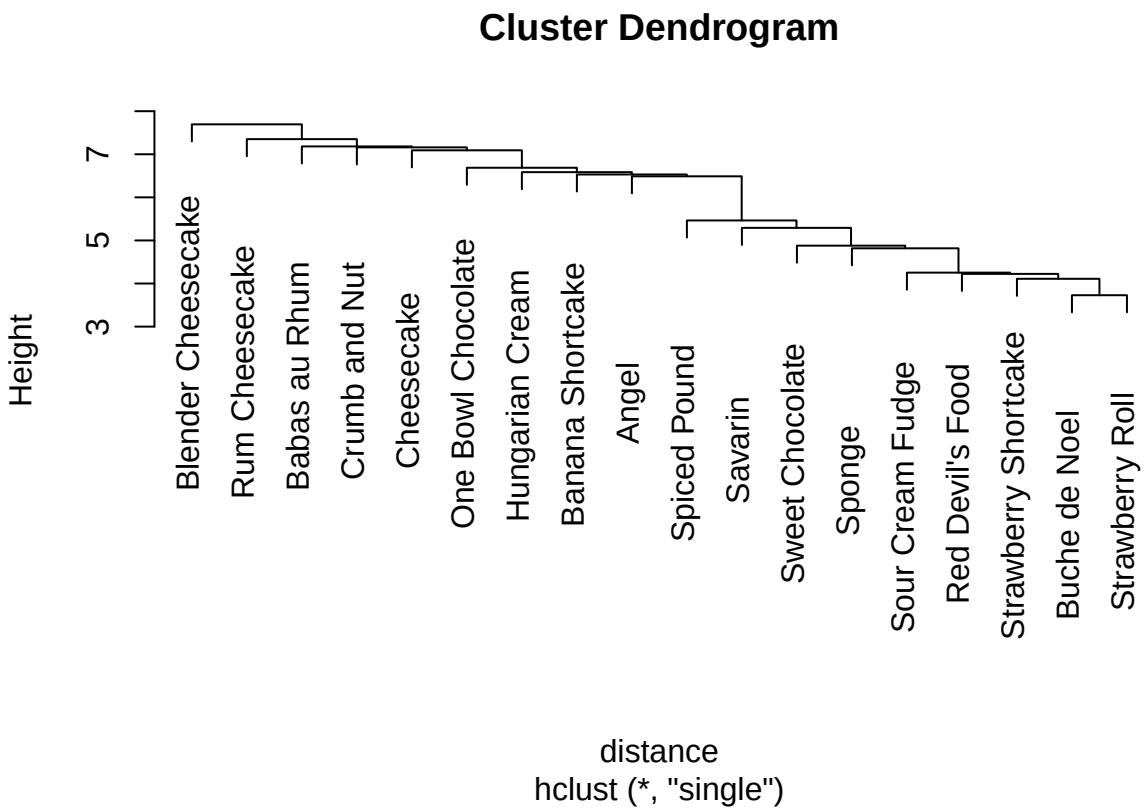
```
hclust_avg <- hclust(distance, method = 'average')
plot(hclust_avg)
```




```
hclust_comp <- hclust(distance, method = 'complete')
plot(hclust_comp)
```




```
hclust_sng <- hclust(distance, method = 'single')
plot(hclust_sng)
```

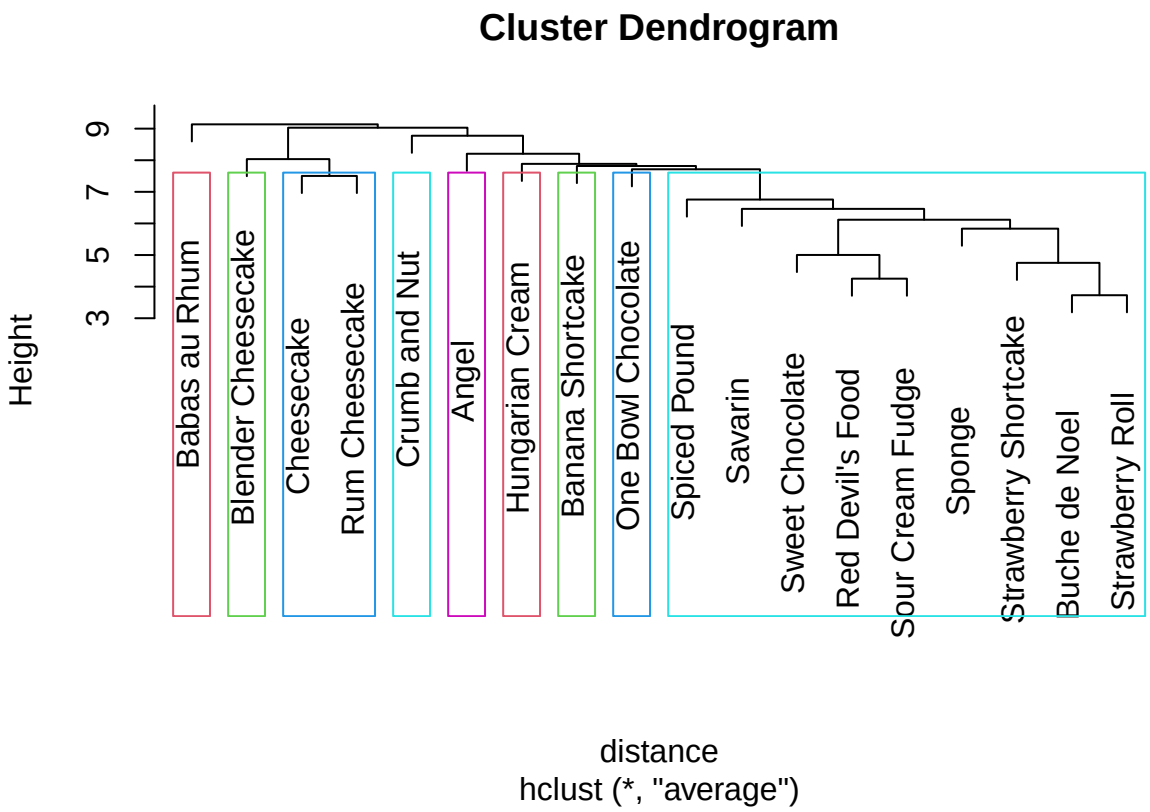


Next, you can cut the dendrogram in order to create the desired number of clusters with cutree function.

```
hc_clusters <- cutree(hclust_avg, k = 4)
hc_clusters
```

```
## Angel                 Babas au Rhum         Sweet Chocolate
##                     1                     2                     1
## Buche de Noel         Cheesecake            Rum Cheesecake
##                     1                     3                     3
## Blender Cheesecake    One Bowl Chocolate    Red Devil's Food
##                     3                     1                     1
## Sour Cream Fudge      Hungarian Cream       Crumb and Nut
##                     1                     1                     4
## Spiced Pound          Strawberry Roll       Savarin
##                     1                     1                     1
## Banana Shortcake      Strawberry Shortcake  Sponge
##                     1                     1                     1
```

```
plot(hclust_avg)
rect.hclust(hclust_avg , k = 9, border = 2:6)
```



How do hierarchical clustering results compare to k-means?

```
table(k3$cluster, hc_clusters )
```

```
##    hc_clusters
##      1  2  3  4
##   1 13  0  0  1
##   2  0  0  3  0
##   3  0  1  0  0
```

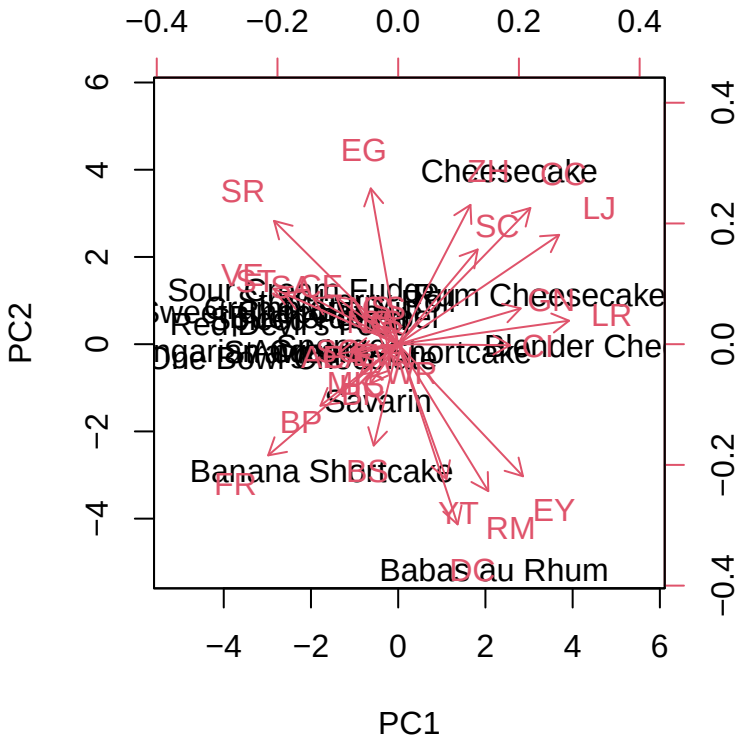
PCA - dimensionality reduction

```
prc <- prcomp(x=df, center = T, scale = F)
head(prc$x)
```

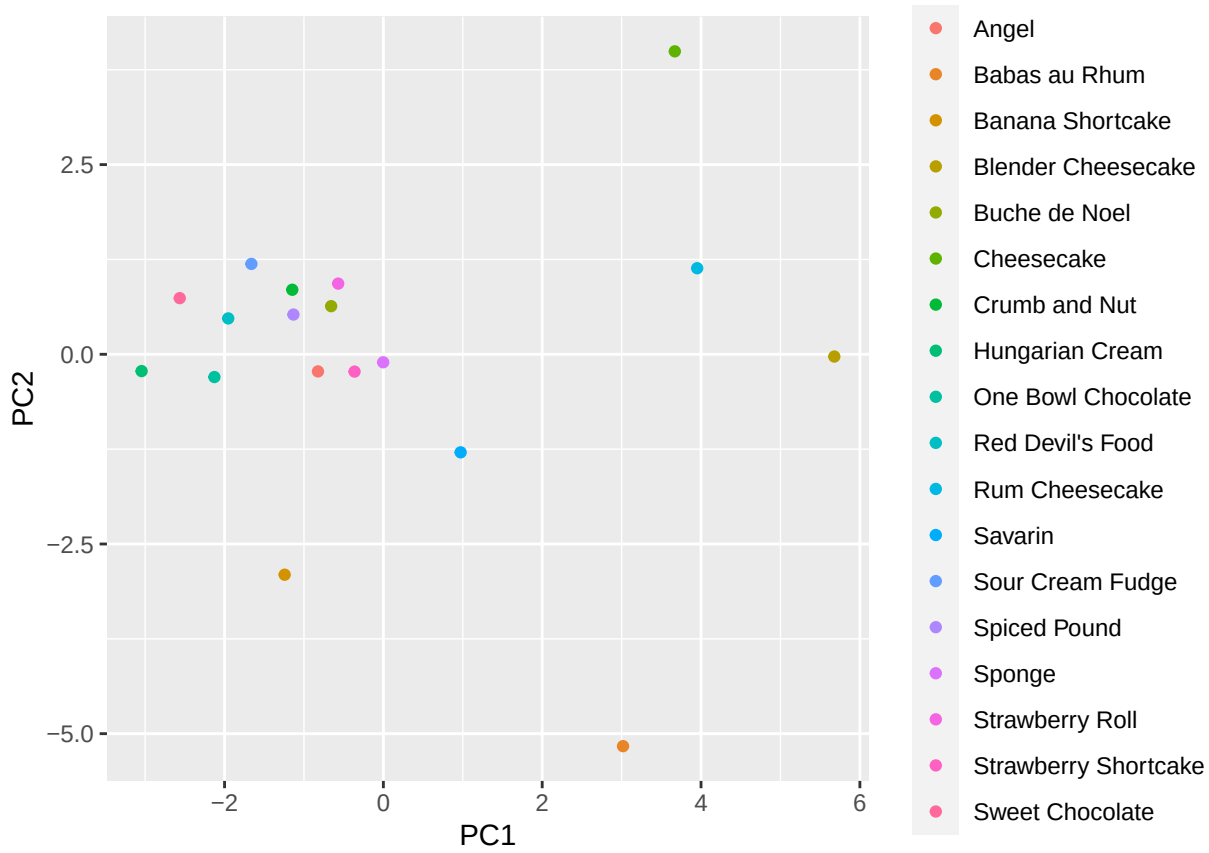
```
##                          PC1        PC2        PC3        PC4         PC5
## Angel              -0.8235218 -0.2256758 -0.4966174  0.7440914 -5.64701646
## Babas au Rhum       3.0176911 -5.1643542  1.0566059  1.8184945  0.03027562
## Sweet Chocolate    -2.5642064  0.7401729  2.3605377  1.5997688  1.57038376
## Buche de Noel      -0.6576046  0.6342797  0.1997411  0.3889312 -0.64700924
## Cheesecake          3.6698127  3.9925296  0.9518445 -0.5209152  0.70128793
## Rum Cheesecake      3.9515945  1.1343036 -1.0201069 -1.1747908  0.93079181
##                          PC6        PC7        PC8        PC9        PC10
## Angel              -0.06193174 -1.0969865  1.1453034 -1.9691781  0.3111457
## Babas au Rhum       0.91414339 -0.4795439 -0.4627946  1.6124560  1.4644421
```

```
## Sweet Chocolate       -0.65511452 -0.9536505  2.1255182  0.5892451 -0.8699609
## Buche de Noel          0.68874930  0.2207774 -0.4637475  0.3368643  0.1525638
## Cheesecake             1.06130001 -0.2432917 -1.2450803 -0.8140686  0.9758170
## Rum Cheesecake        -0.86635860 -3.5324519  0.8127749  0.1147226  0.9226391
##                             PC11        PC12        PC13        PC14
## Angel                 -0.5508124  0.03701621  0.06100216 -0.28356037
## Babas au Rhum         -0.5902464 -0.83083671  1.03150650  0.03439265
## Sweet Chocolate       -0.6314049  0.54541762  0.44540371 -1.29414622
## Buche de Noel          0.4826963  0.15700511 -0.06453949  1.13202980
## Cheesecake            -2.6523373 -0.61322461  0.47636889  0.22087322
## Rum Cheesecake         1.8789074  0.97942279  0.31322422 -0.39300934
##                              PC15        PC16         PC17          PC18
## Angel                 0.005206188 -0.09335977  0.068761642  5.143501e-16
## Babas au Rhum         0.048318284 -0.04503813  0.034264868 -9.020104e-17
## Sweet Chocolate       1.093029422 -0.35725199 -0.189478143 -2.246421e-16
## Buche de Noel         0.539825418  0.86776485 -1.017636620  9.107756e-17
## Cheesecake            0.091372778 -0.25299304  0.034043293  2.938858e-16
## Rum Cheesecake       -0.417089758  0.30450979 -0.008335133 -2.281116e-16
```

```
biplot(prc, scale = 0)
```




```
prc_adj <- data.frame(prc$x)
ggplot(data = prc_adj, aes(x=PC1, y=PC2, color= row.names(prc_adj)))+
  geom_point()
```



Percent of variance explained by components

```
prc_var <- prc$sdev^2
prc_var
```

```
##  [1] 6.119756e+00 3.453910e+00 3.303585e+00 2.847598e+00 2.697385e+00
##  [6] 2.416901e+00 2.067171e+00 1.936912e+00 1.884374e+00 1.612418e+00
## [11] 1.272094e+00 1.134353e+00 9.169980e-01 5.615575e-01 3.759828e-01
## [16] 2.884269e-01 1.105782e-01 2.575097e-32
```

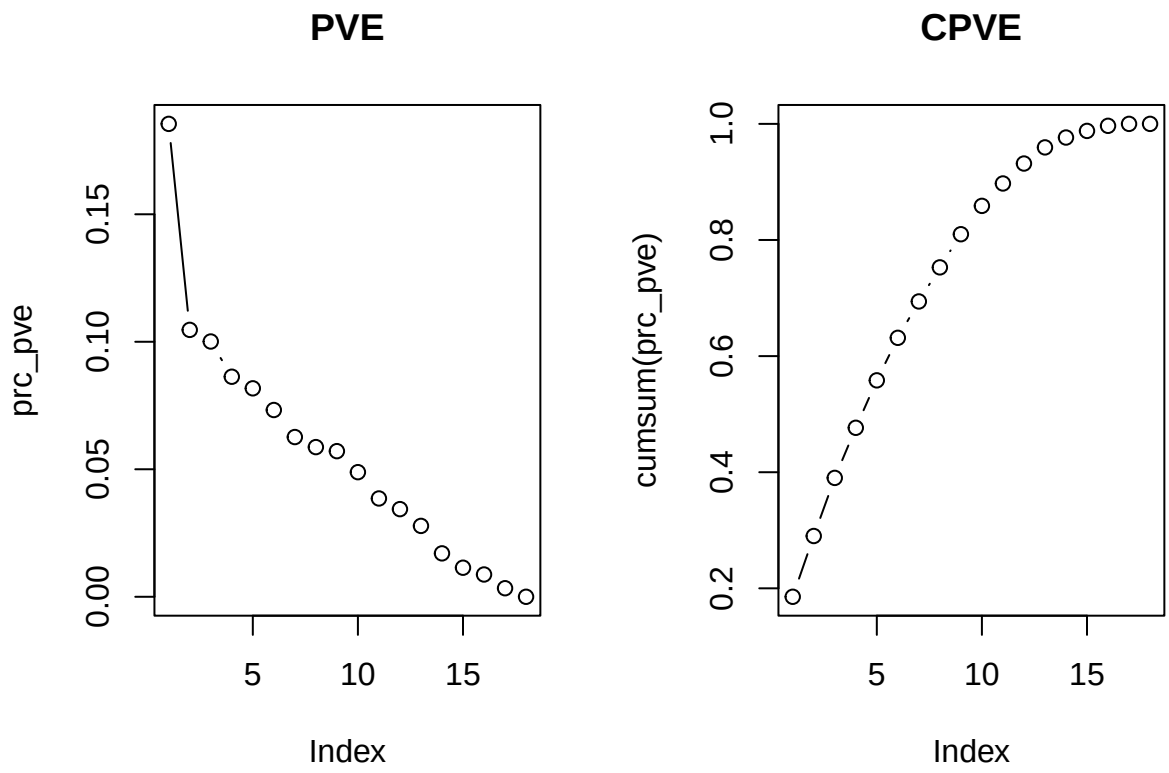
```
prc_pve <- prc_var/sum(prc_var)
prc_pve
```

```
##  [1] 1.854471e-01 1.046639e-01 1.001086e-01 8.629084e-02 8.173895e-02
##  [6] 7.323942e-02 6.264156e-02 5.869432e-02 5.710224e-02 4.886114e-02
## [11] 3.854831e-02 3.437433e-02 2.778782e-02 1.701689e-02 1.139342e-02
## [16] 8.740208e-03 3.350853e-03 7.803325e-34
```

```
cumsum(prc_pve)
```

```
##  [1] 0.1854471 0.2901111 0.3902197 0.4765106 0.5582495 0.6314889 0.6941305
##  [8] 0.7528248 0.8099270 0.8587882 0.8973365 0.9317108 0.9594986 0.9765155
## [15] 0.9879089 0.9966491 1.0000000 1.0000000
```

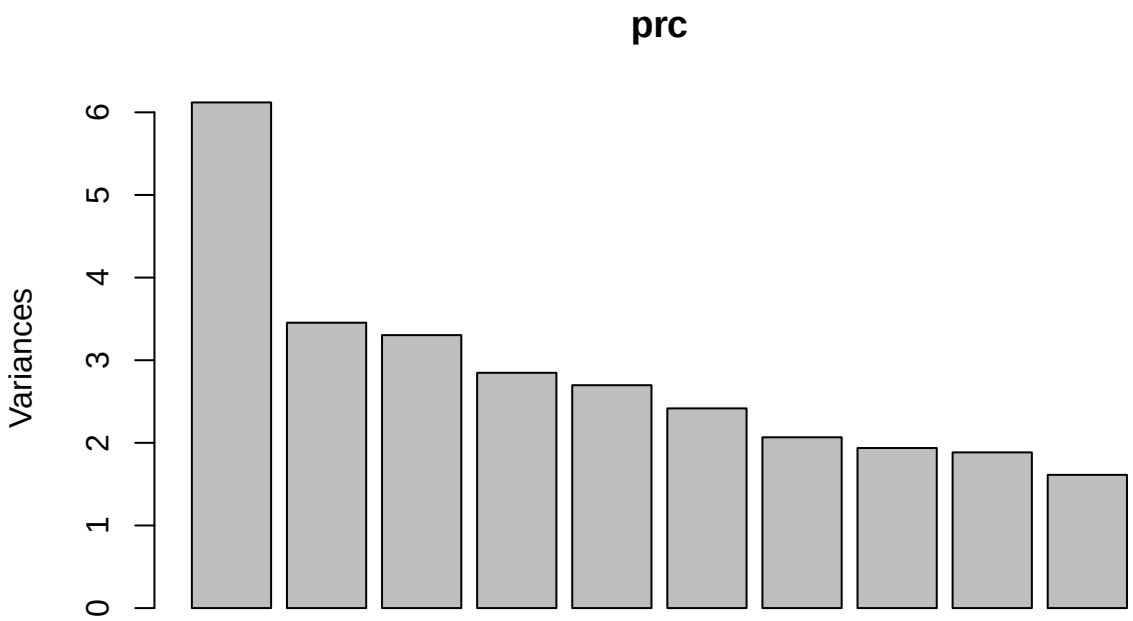
```r
par(mfrow=c(1,2))
plot(prc_pve, type = 'b', main = 'PVE')
plot(cumsum(prc_pve), type = 'b', main = 'CPVE')
```




```r
par(mfrow=c(1,1))
```

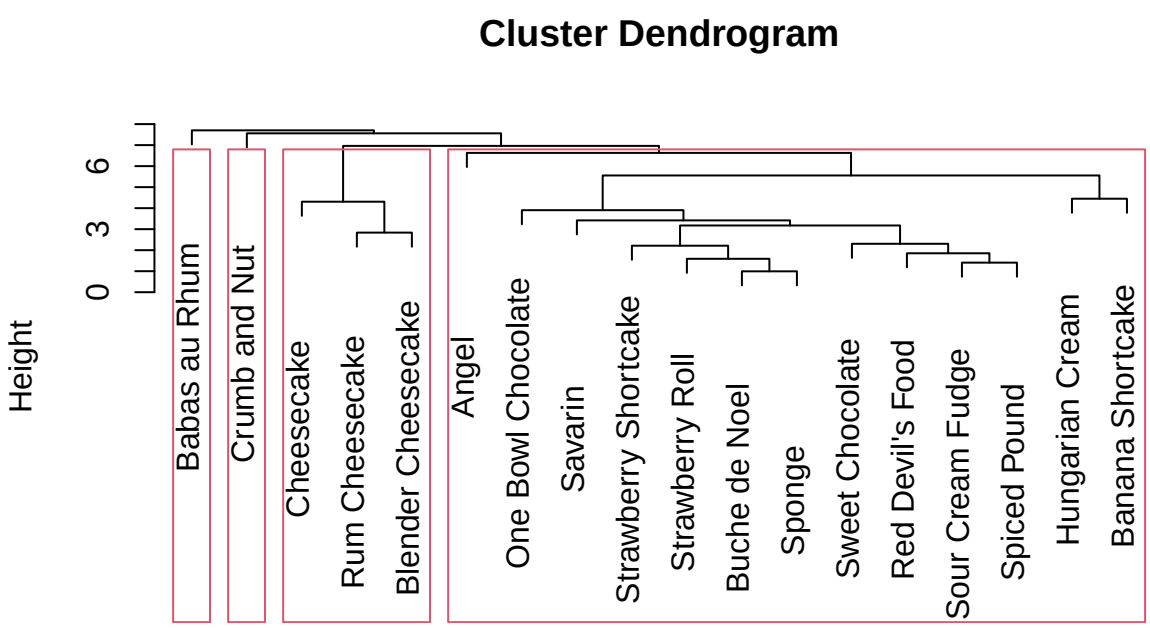
Variance explainet by the first few components (height = prc$sdevˆ2)

```r
plot(prc)
```



We can perform h-clustering on the first few (5 in our case) PC score vectors

```
hc_prc <- hclust(dist(prc$x[,1:5]), method = "average")
plot(hc_prc)
rect.hclust(hc_prc, k=4)
```




```
hc_pca_res <- cutree(hc_prc, 4)
hc_pca_res
```

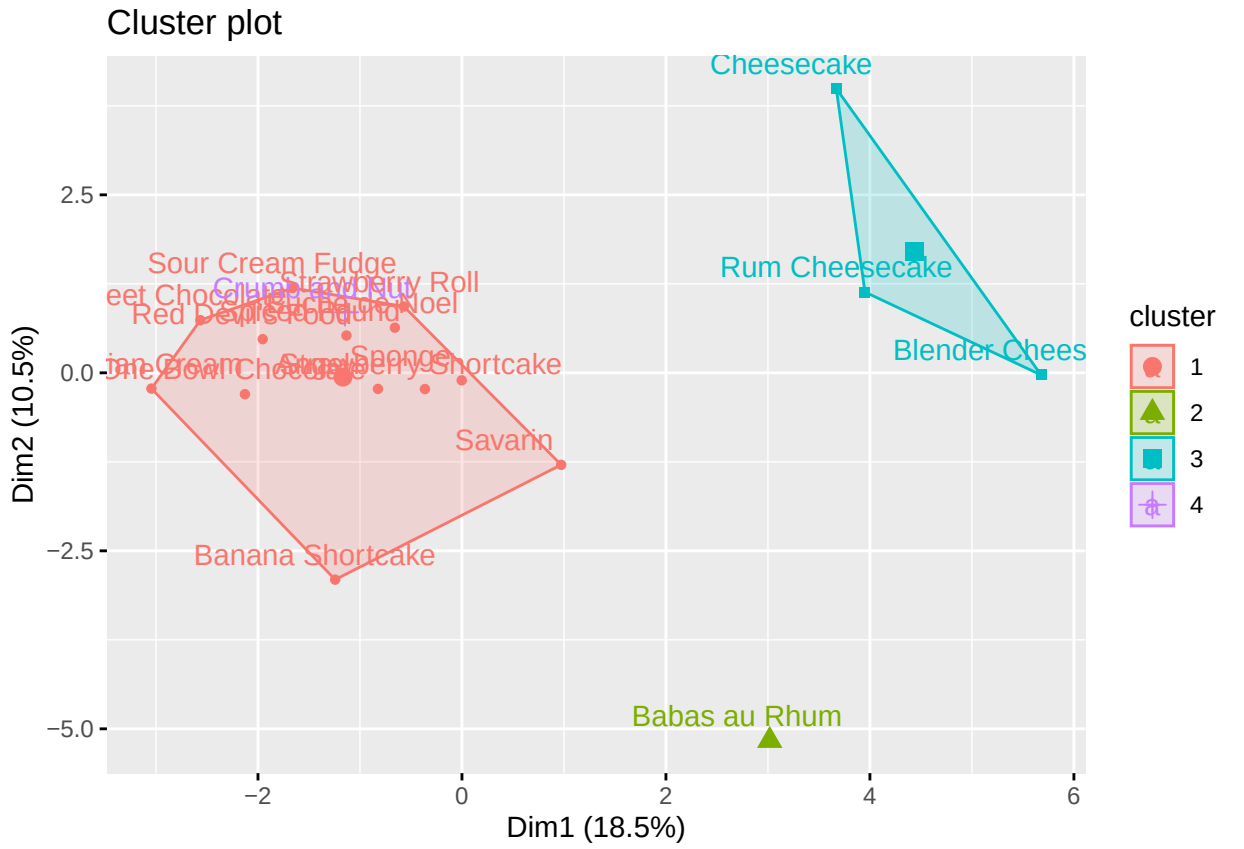
```
## Angel                 Babas au Rhum         Sweet Chocolate
##                     1                     2                     1
## Buche de Noel         Cheesecake            Rum Cheesecake
##                     1                     3                     3
## Blender Cheesecake    One Bowl Chocolate    Red Devil’s Food
##                     3                     1                     1
## Sour Cream Fudge      Hungarian Cream       Crumb and Nut
##                     1                     1                     4
## Spiced Pound          Strawberry Roll       Savarin
##                     1                     1                     1
## Banana Shortcake      Strawberry Shortcake  Sponge
##                     1                     1                     1
```

```
table(hc_clusters, hc_pca_res)
```

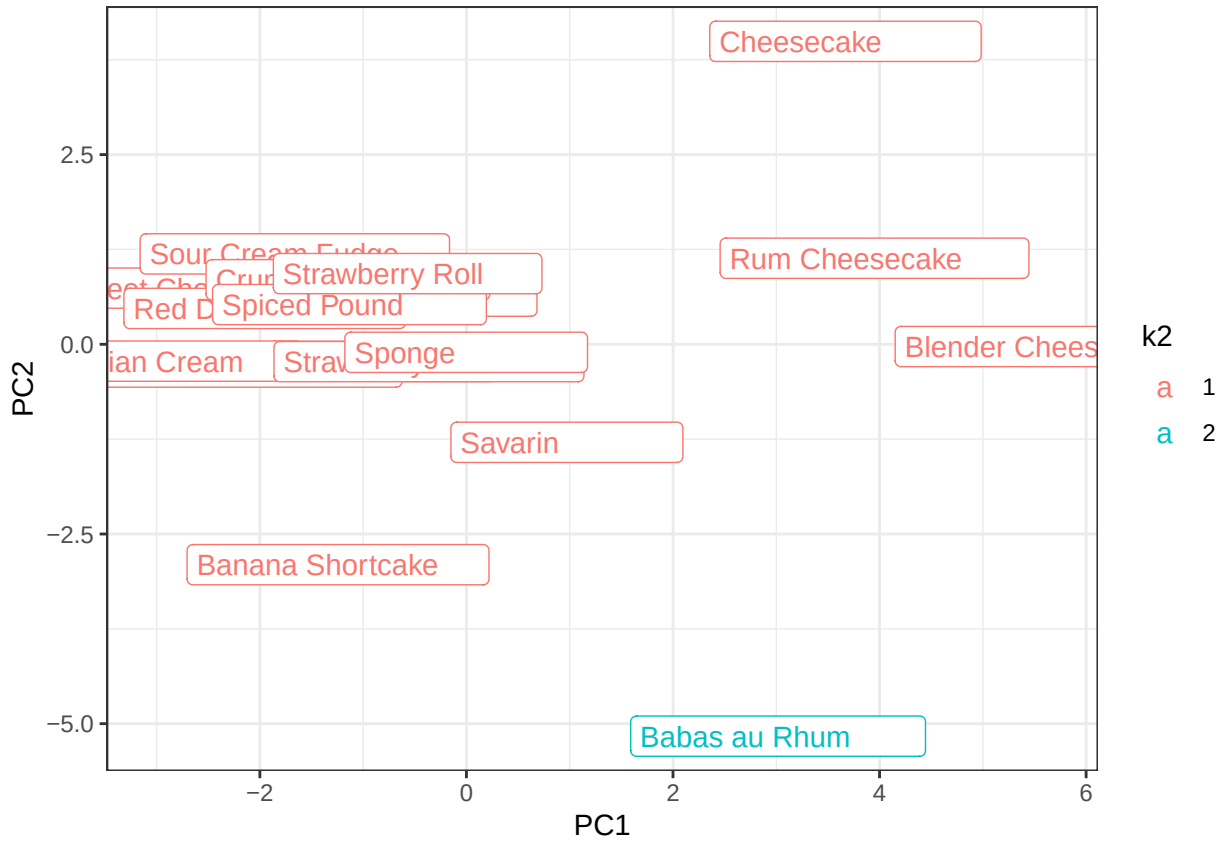
```
##             hc_pca_res
## hc_clusters  1  2  3  4
##           1 13  0  0  0
##           2  0  1  0  0
##           3  0  0  3  0
##           4  0  0  0  1
```

Results are concordant with k-means with k=4

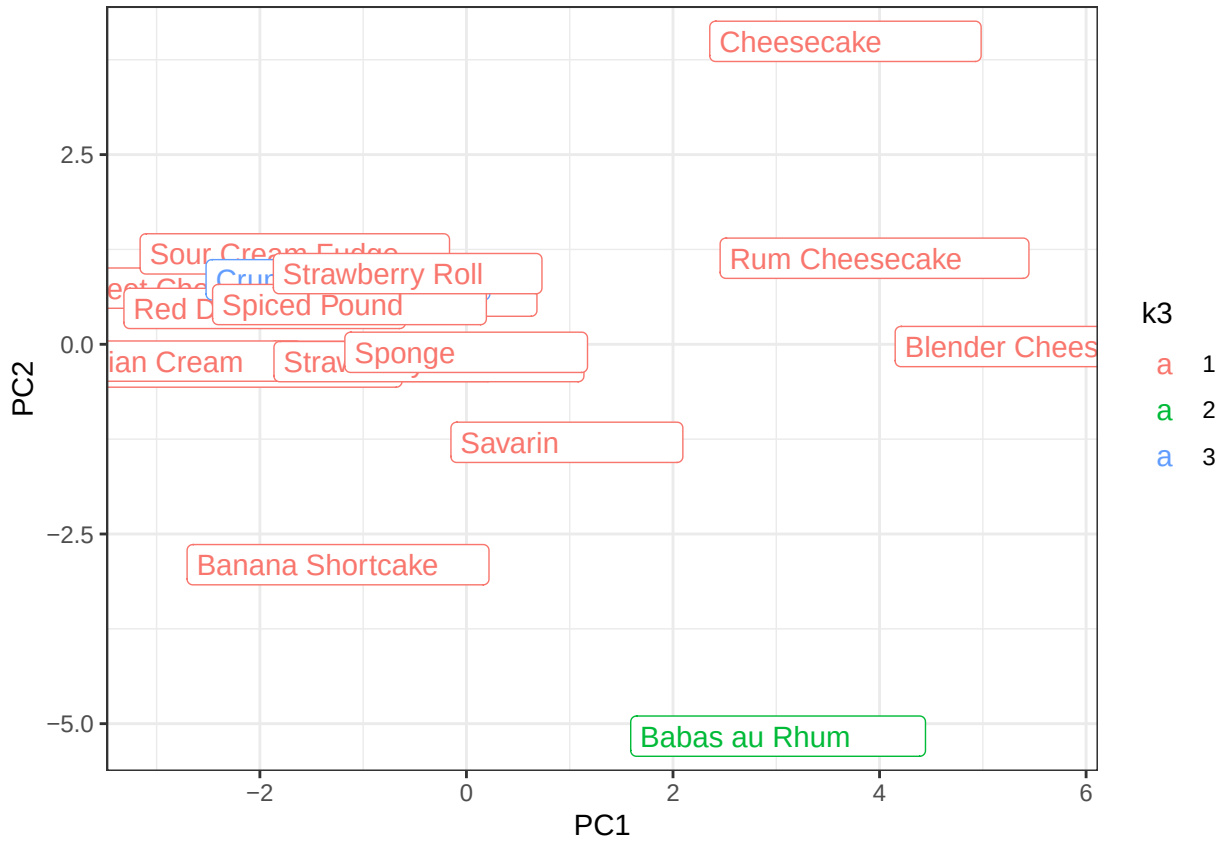
```
fviz_cluster(k4,  data = df)
```



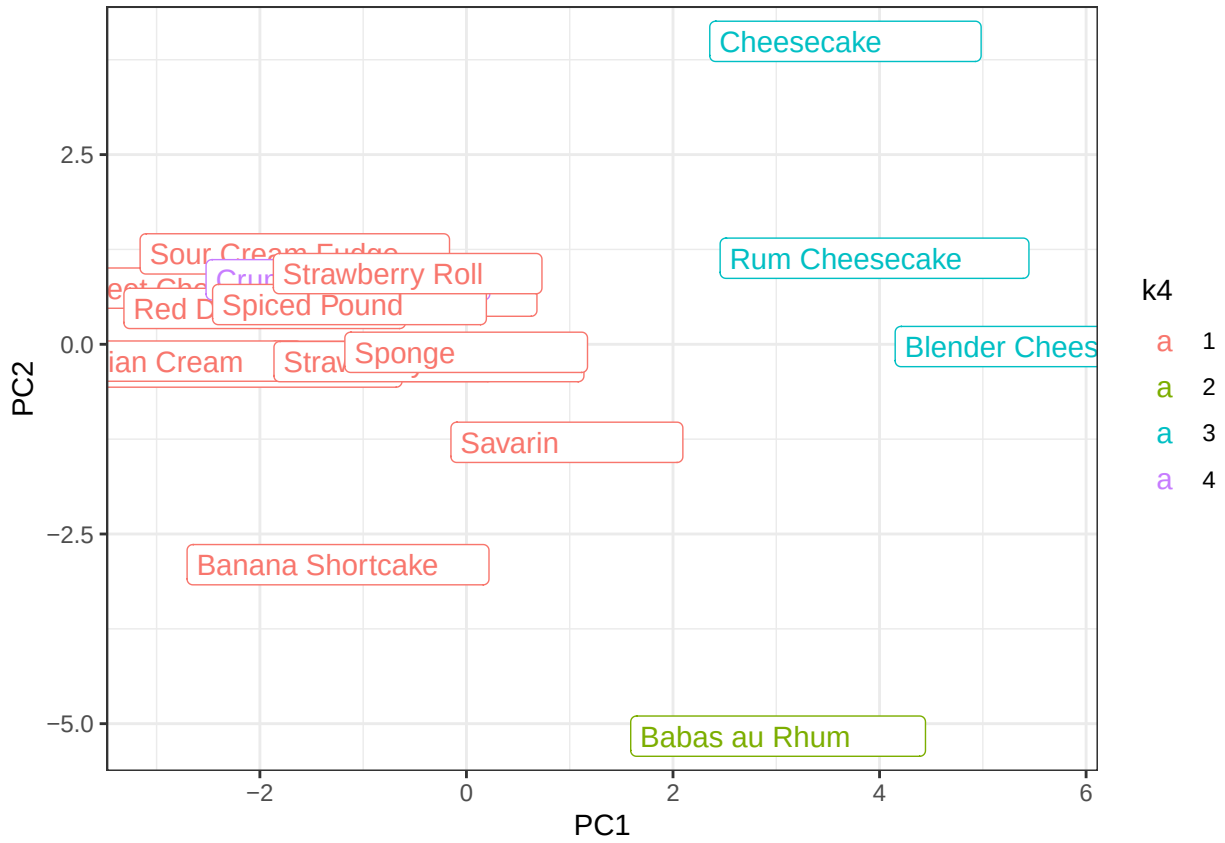

```
data_pca_clust <- data.frame(prc$x[, 1:2],
                             k2 = factor(cutree(hc_prc, 2)),
                             k3 = factor(cutree(hc_prc, 3)),
                             k4 = factor(cutree(hc_prc, 4)),
                             k5 = factor(cutree(hc_prc, 5)))
ggplot(data_pca_clust,  aes(x = PC1, y = PC2, color = k2, label = row.names(data_pca_clust))) + geom_po
```



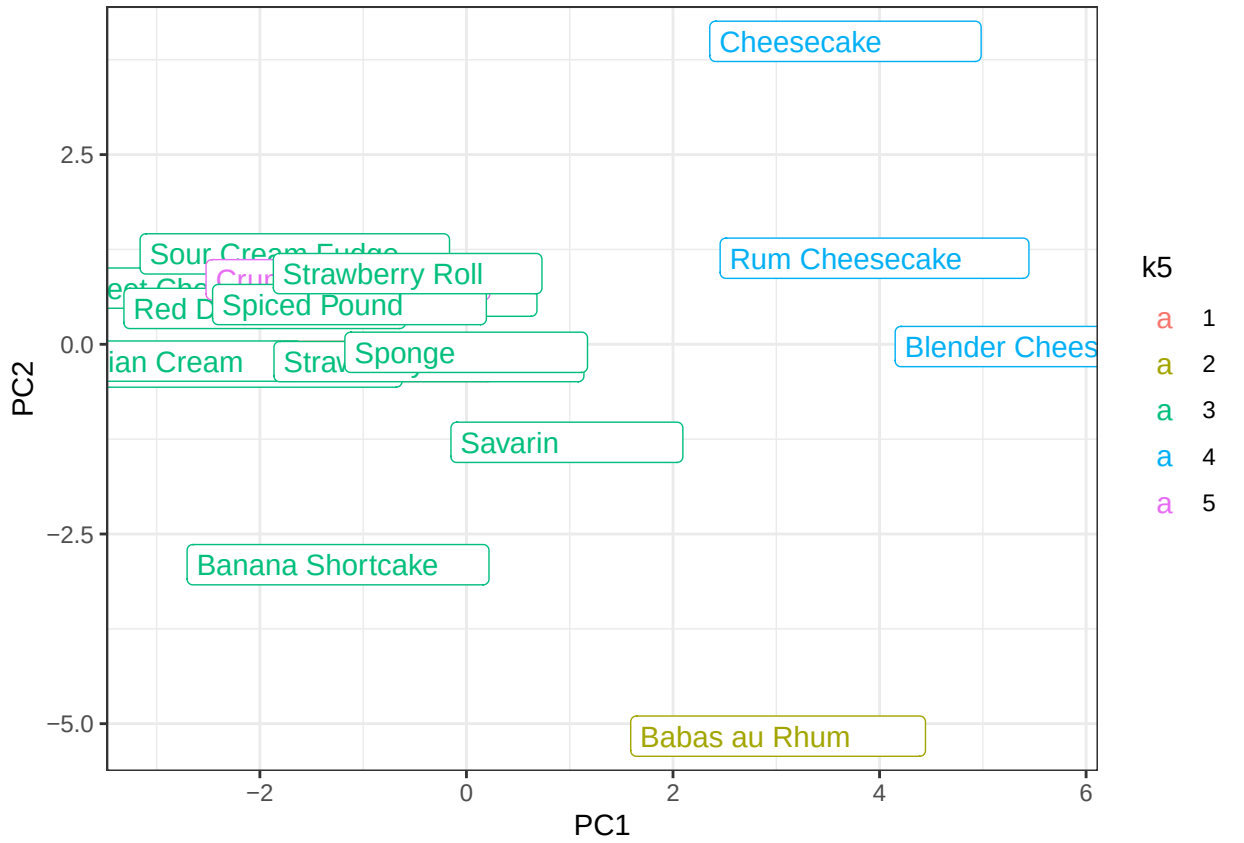

```
ggplot(data_pca_clust,  aes(x = PC1, y = PC2, color = k3, label = row.names(data_pca_clust))) + geom_po
```




```
ggplot(data_pca_clust,  aes(x = PC1, y = PC2, color = k4, label= row.names(data_pca_clust))) + geom_poi
```



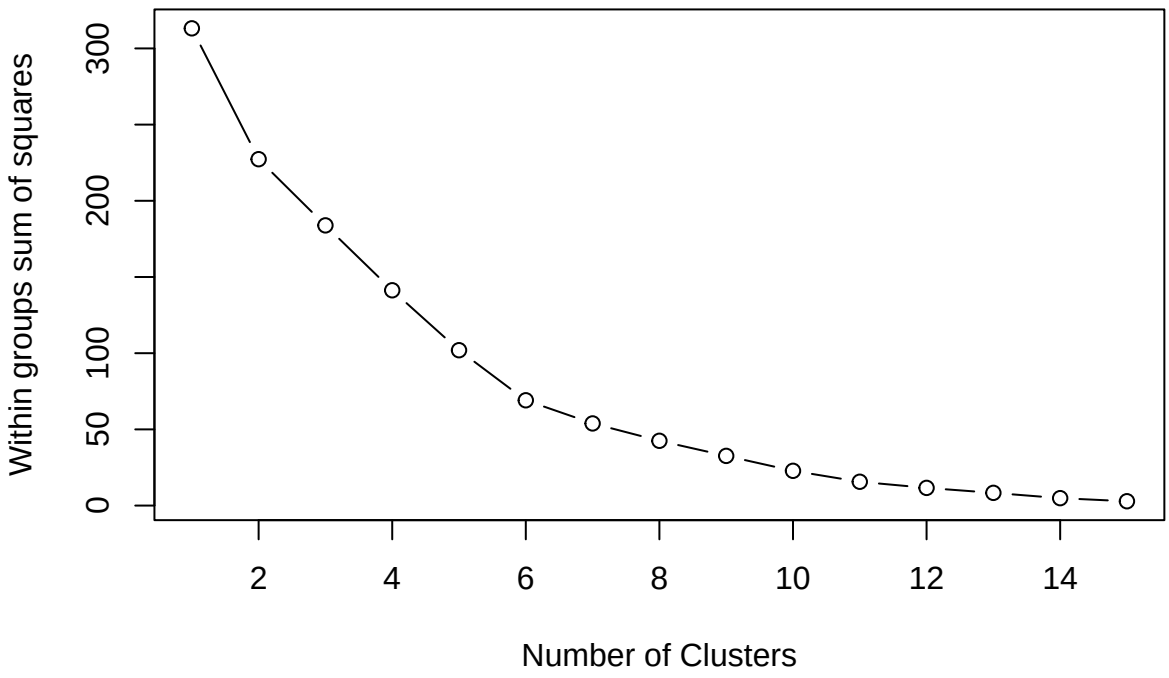

```
ggplot(data_pca_clust,  aes(x = PC1, y = PC2, color = k5, label = row.names(data_pca_clust))) + geom_po
```



We can provide h-clustering on pca results

```
pca <- prc$x[,1:5]

wss <- (nrow(pca)-1)*sum(apply(pca,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(pca,
   centers=i, nstart = 20)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
  ylab="Within groups sum of squares")
```

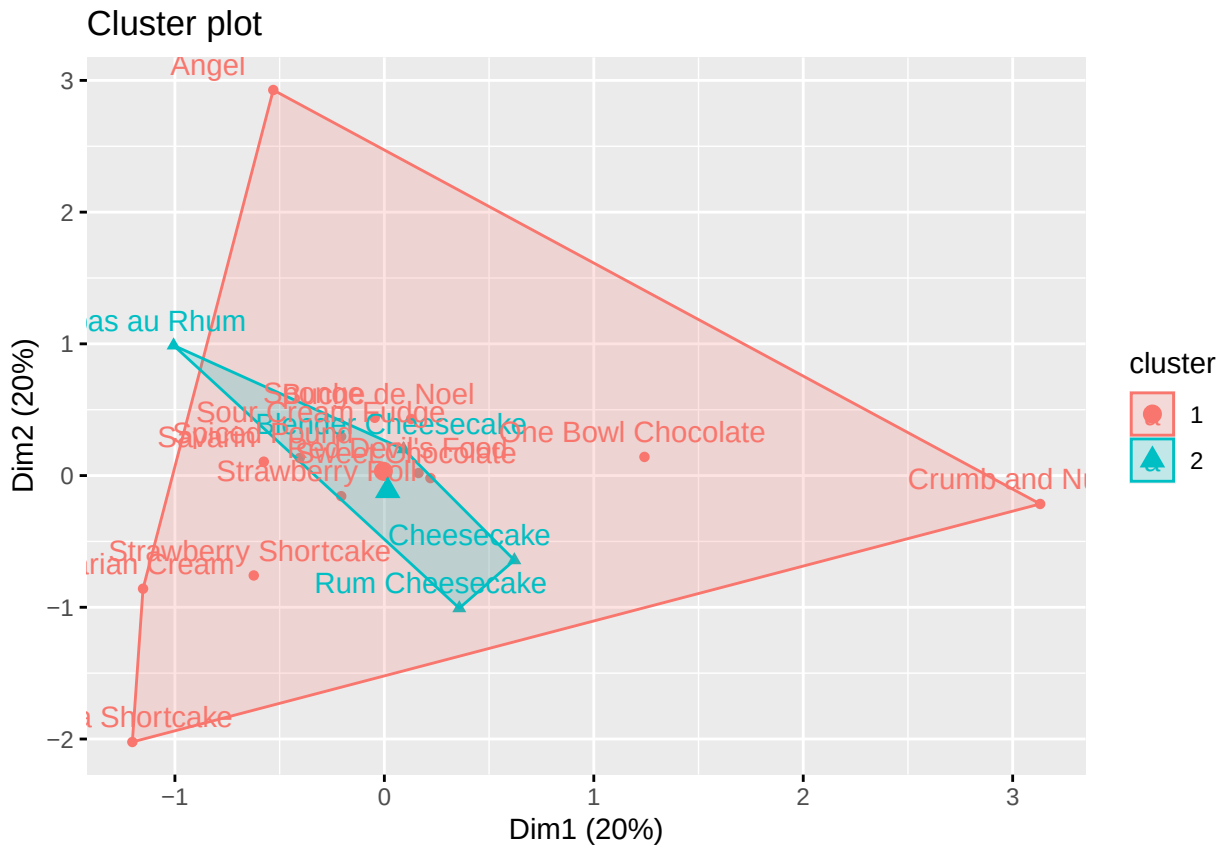



```
km_pca <- kmeans(pca, 2, nstart = 20)
km_pca
```

```
## K-means clustering with 2 clusters of sizes 14, 4
##
## Cluster means:
##         PC1         PC2          PC3          PC4         PC5
## 1 -1.165516  0.00479409 -0.009307346  0.005286202 -0.06723736
## 2  4.079307 -0.01677932  0.032575713 -0.018501705  0.23533077
##
## Clustering vector:
## Angel                 Babas au Rhum         Sweet Chocolate
##                     1                     2                     1
## Buche de Noel         Cheesecake            Rum Cheesecake
##                     1                     2                     2
## Blender Cheesecake    One Bowl Chocolate    Red Devil's Food
##                     2                     1                     1
## Sour Cream Fudge      Hungarian Cream       Crumb and Nut
##                     1                     1                     1
## Spiced Pound          Strawberry Roll       Savarin
##                     1                     1                     1
## Banana Shortcake      Strawberry Shortcake  Sponge
##                     1                     1                     1
##
## Within cluster sum of squares by cluster:
```

```
## [1] 169.09073  58.21276
##  (between_SS / total_SS =  27.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
fviz_cluster(km_pca, data=pca)
```




```
table(k2$cluster, km_pca$cluster)
```

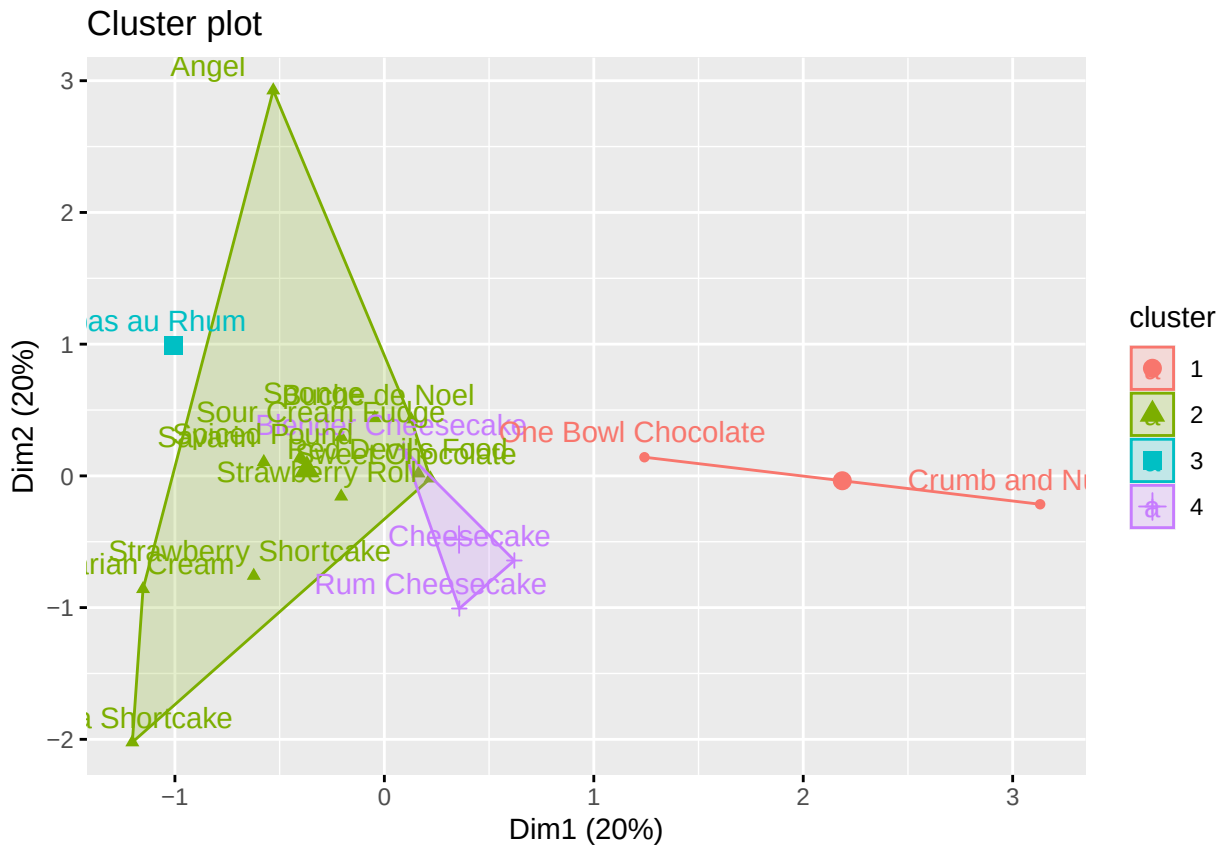
```
##
##      1  2
##   1  0  4
##   2 14  0
```

```
km_pca4 <- kmeans(pca, 4, nstart = 20)
km_pca4
```

```
## K-means clustering with 4 clusters of sizes 2, 12, 1, 3
##
## Cluster means:
##         PC1        PC2        PC3        PC4         PC5
```

```
## 1 -1.637537  0.2753172 -3.3862329  2.3368209  1.11512241
## 2 -1.086846 -0.0402931  0.5535136 -0.3833029 -0.26429733
## 3  3.017691 -5.1643542  1.0566059  1.8184945  0.03027562
## 4  4.433180  1.6990790 -0.3087677 -0.6308338  0.30368249
##
## Clustering vector:
## Angel                 Babas au Rhum         Sweet Chocolate
##                     2                     3                     2
## Buche de Noel         Cheesecake            Rum Cheesecake
##                     2                     4                     4
## Blender Cheesecake    One Bowl Chocolate    Red Devil's Food
##                     4                     1                     2
## Sour Cream Fudge      Hungarian Cream       Crumb and Nut
##                     2                     2                     1
## Spiced Pound          Strawberry Roll       Savarin
##                     2                     2                     2
## Banana Shortcake      Strawberry Shortcake  Sponge
##                     2                     2                     2
##
## Within cluster sum of squares by cluster:
## [1]   9.799286 116.046282   0.000000  15.426368
##  (between_SS / total_SS =  54.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
fviz_cluster(km_pca4, data=pca)
```




```
table(k4$cluster, km_pca4$cluster)
```

```
##
##      1  2  3  4
##   1  1 12  0  0
##   2  0  0  1  0
##   3  0  0  0  3
##   4  1  0  0  0
```