Machine Learning

Pham Vo

2/13/2022

# Clustering with kmeans and hclust()

We will begin by making up some data to cluster. shortcut to insert code chunk: option+command+i

tmp <- c(rnorm(30,3), rnorm(30,-3))  
x <- cbind(x=tmp,y=rev(tmp))  
x

## x y  
## [1,] 2.8743002 -2.8901669  
## [2,] 2.9363182 -2.9693210  
## [3,] 2.9073333 -2.4767405  
## [4,] 3.8818058 -3.7129839  
## [5,] 2.7414859 -3.3318831  
## [6,] 5.0719478 -3.9934698  
## [7,] 2.3802322 -2.3318479  
## [8,] 2.4884059 -3.5132228  
## [9,] 3.4515539 -4.4218790  
## [10,] 2.5529855 -4.0744393  
## [11,] 3.7925742 -3.6546072  
## [12,] 1.8461343 -2.6705285  
## [13,] 0.6354836 -4.5093889  
## [14,] 3.7067287 -1.2517350  
## [15,] 1.9147373 -2.0754128  
## [16,] 1.3035310 -1.9679169  
## [17,] 0.8779099 -3.3388093  
## [18,] 2.3805892 -3.1115670  
## [19,] 5.5112480 -3.6679013  
## [20,] 3.6975447 -3.4021133  
## [21,] 2.7999729 -2.0881202  
## [22,] 1.1944247 -1.4130999  
## [23,] 4.7876043 -4.3416825  
## [24,] 3.5380811 -1.1177078  
## [25,] 5.6655361 -3.1689216  
## [26,] 3.3221488 -2.1432095  
## [27,] 4.5109872 -1.8455444  
## [28,] 1.0453575 -4.2434291  
## [29,] 2.4474572 -3.2728480  
## [30,] 2.6725764 -2.1365590  
## [31,] -2.1365590 2.6725764  
## [32,] -3.2728480 2.4474572  
## [33,] -4.2434291 1.0453575  
## [34,] -1.8455444 4.5109872  
## [35,] -2.1432095 3.3221488  
## [36,] -3.1689216 5.6655361  
## [37,] -1.1177078 3.5380811  
## [38,] -4.3416825 4.7876043  
## [39,] -1.4130999 1.1944247  
## [40,] -2.0881202 2.7999729  
## [41,] -3.4021133 3.6975447  
## [42,] -3.6679013 5.5112480  
## [43,] -3.1115670 2.3805892  
## [44,] -3.3388093 0.8779099  
## [45,] -1.9679169 1.3035310  
## [46,] -2.0754128 1.9147373  
## [47,] -1.2517350 3.7067287  
## [48,] -4.5093889 0.6354836  
## [49,] -2.6705285 1.8461343  
## [50,] -3.6546072 3.7925742  
## [51,] -4.0744393 2.5529855  
## [52,] -4.4218790 3.4515539  
## [53,] -3.5132228 2.4884059  
## [54,] -2.3318479 2.3802322  
## [55,] -3.9934698 5.0719478  
## [56,] -3.3318831 2.7414859  
## [57,] -3.7129839 3.8818058  
## [58,] -2.4767405 2.9073333  
## [59,] -2.9693210 2.9363182  
## [60,] -2.8901669 2.8743002

## Run kmeans()

k <- kmeans(x, centers=2, nstart=20)  
k

## K-means clustering with 2 clusters of sizes 30, 30  
##   
## Cluster means:  
## x y  
## 1 -2.971235 2.964567  
## 2 2.964567 -2.971235  
##   
## Clustering vector:  
## [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1  
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
##   
## Within cluster sum of squares by cluster:  
## [1] 77.9083 77.9083  
## (between\_SS / total\_SS = 87.2 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

Q. what size is each cluster?

k$size

## [1] 30 30

Q. Cluster centers

k$centers

## x y  
## 1 -2.971235 2.964567  
## 2 2.964567 -2.971235

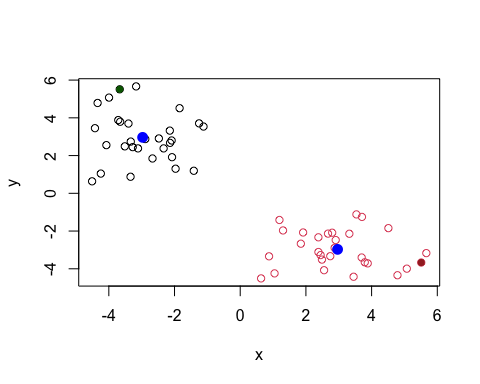
Q. Membership vector

k$cluster

## [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1  
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Plot our data with the clustering result

plot(x, col=k$cluster)  
points(k$centers, col="blue", pch=16, cex=1.5)  
points(x[42,1], x[42,2], col="darkgreen", pch = 16)  
points(x[19,1], x[19,2], col="brown", pch = 16)



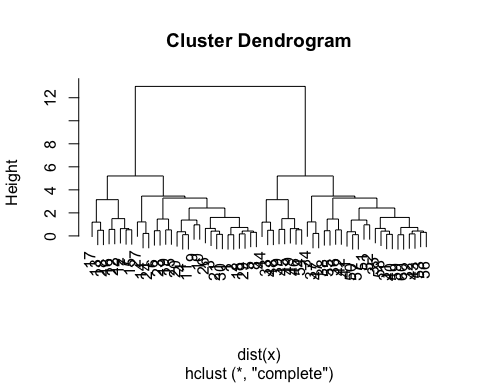
## hclust()

hc <- hclust(dist(x))  
hc

##   
## Call:  
## hclust(d = dist(x))  
##   
## Cluster method : complete   
## Distance : euclidean   
## Number of objects: 60

There is a cool and useful plot method for hclust()

plot(hc)



# Principal Component Analysis

Data import

url <- "https://tinyurl.com/UK-foods"  
x <- read.csv(url, row.names=1)  
x

## England Wales Scotland N.Ireland  
## Cheese 105 103 103 66  
## Carcass\_meat 245 227 242 267  
## Other\_meat 685 803 750 586  
## Fish 147 160 122 93  
## Fats\_and\_oils 193 235 184 209  
## Sugars 156 175 147 139  
## Fresh\_potatoes 720 874 566 1033  
## Fresh\_Veg 253 265 171 143  
## Other\_Veg 488 570 418 355  
## Processed\_potatoes 198 203 220 187  
## Processed\_Veg 360 365 337 334  
## Fresh\_fruit 1102 1137 957 674  
## Cereals 1472 1582 1462 1494  
## Beverages 57 73 53 47  
## Soft\_drinks 1374 1256 1572 1506  
## Alcoholic\_drinks 375 475 458 135  
## Confectionery 54 64 62 41

#Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

dim(x)

## [1] 17 4

ncol(x)

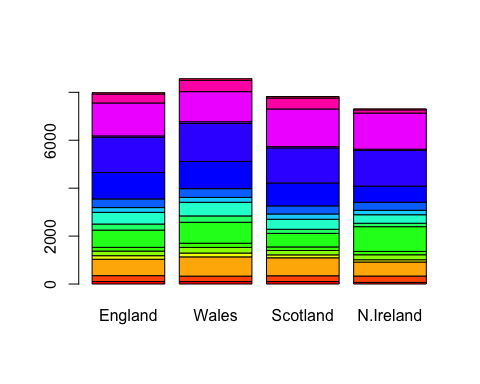
## [1] 4

nrow(x)

## [1] 17

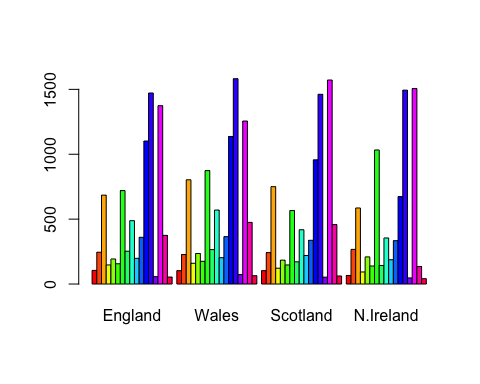
Let’s make some typical plots

mycols <- rainbow(nrow(x))  
barplot(as.matrix(x), col=mycols)



Make it side by side i.e. not stacked

barplot(as.matrix(x), col=mycols, beside=TRUE)

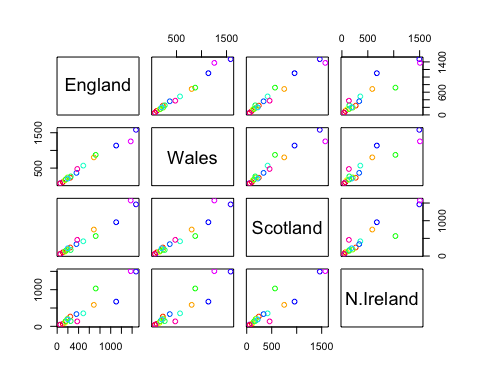


One plot that is helpful here.

#Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

No difference in values between x-axis and y-axis.

pairs(x, col=mycols)



#Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

# PCA to rescue

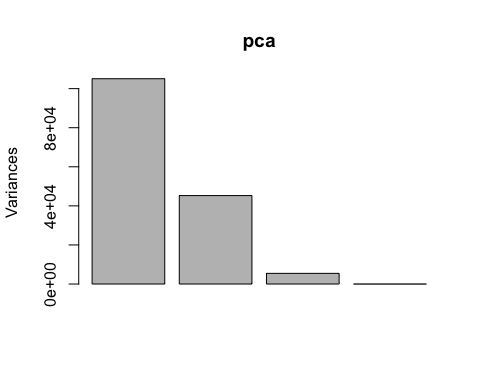
Do PCR of this 17D food data. The main function in base R is called “prcom()”. This function requires the transpose of our data in this case…

pca <- prcomp(t(x))  
summary(pca)

## Importance of components:  
## PC1 PC2 PC3 PC4  
## Standard deviation 324.1502 212.7478 73.87622 4.189e-14  
## Proportion of Variance 0.6744 0.2905 0.03503 0.000e+00  
## Cumulative Proportion 0.6744 0.9650 1.00000 1.000e+00

The prcom() function returns a list object.

plot(pca)



The PCA plot a.k.a a PCA score plot is a plot of PC1 vs PV2. Basically using the new PCA axis to view our data.

attributes(pca)

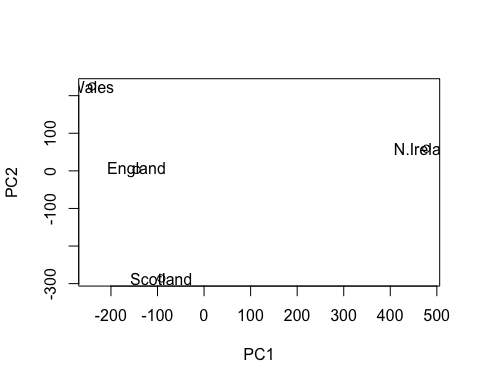
## $names  
## [1] "sdev" "rotation" "center" "scale" "x"   
##   
## $class  
## [1] "prcomp"

We will focus on “pca$x” for this plot

pca$x

## PC1 PC2 PC3 PC4  
## England -144.99315 2.532999 -105.768945 2.842865e-14  
## Wales -240.52915 224.646925 56.475555 7.804382e-13  
## Scotland -91.86934 -286.081786 44.415495 -9.614462e-13  
## N.Ireland 477.39164 58.901862 4.877895 1.448078e-13

plot(pca$x[,1], pca$x[,2], xlab= "PC1", ylab="PC2")  
text(pca$x[,1], pca$x[,2], labels = colnames(x))

 ## PCA of a RNA-Seq

url2 <- "https://tinyurl.com/expression-CSV"  
rna.data <- read.csv(url2, row.names=1)  
head(rna.data)

## wt1 wt2 wt3 wt4 wt5 ko1 ko2 ko3 ko4 ko5  
## gene1 439 458 408 429 420 90 88 86 90 93  
## gene2 219 200 204 210 187 427 423 434 433 426  
## gene3 1006 989 1030 1017 973 252 237 238 226 210  
## gene4 783 792 829 856 760 849 856 835 885 894  
## gene5 181 249 204 244 225 277 305 272 270 279  
## gene6 460 502 491 491 493 612 594 577 618 638

pca <- prcomp(t(rna.data))  
summary(pca)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516  
## Proportion of Variance 0.9917 0.0016 0.00144 0.00122 0.00098 0.00093  
## Cumulative Proportion 0.9917 0.9933 0.99471 0.99593 0.99691 0.99784  
## PC7 PC8 PC9 PC10  
## Standard deviation 65.29428 59.90981 53.20803 3.142e-13  
## Proportion of Variance 0.00086 0.00073 0.00057 0.000e+00  
## Cumulative Proportion 0.99870 0.99943 1.00000 1.000e+00

Make our PCA score plot

plot(pca$x[,1:2])  
text(pca$x[,1:2], labels = colnames(rna.data))

