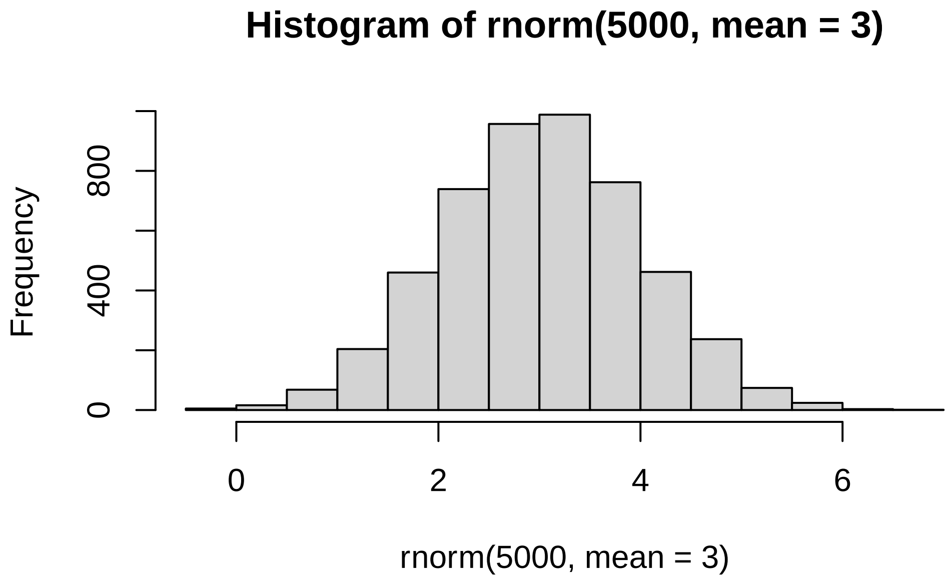
class_07 Machine Learning

Iulia Rusu

Main function for kmeans clustering in base R is called 'kmeans()'

```
hist(rnorm(5000, mean= 3))
```

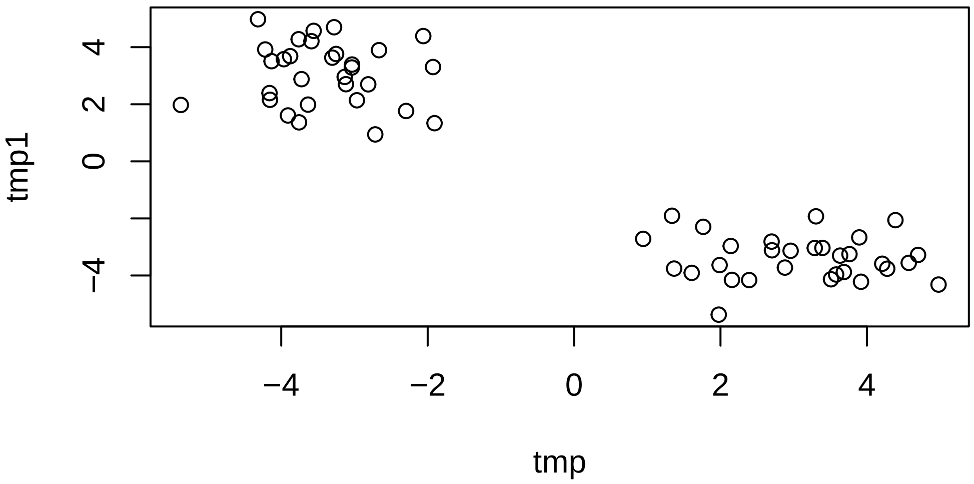


```
# Make a vector with 60 total points half centered at +3 and half centered at -3
tmp <- c(rnorm(30, mean =3), rnorm(30, mean= -3))
#reversed a vector to generate another vector with temp
tmp1 <- rev(tmp)
#cbind to bind to vectors as columns, generates matrix
temperature <- cbind(tmp, tmp1)
temperature
```

```
            tmp      tmp1
 [1,]  3.579611 -3.964540
 [2,]  2.140252 -2.966405
 [3,]  4.389818 -2.059953
 [4,]  3.509826 -4.132507
 [5,]  2.703410 -3.116287
 [6,]  1.366379 -3.757341
 [7,]  1.763849 -2.294419
 [8,]  4.277852 -3.762414
 [9,]  2.391975 -4.160339
[10,]  3.633930 -3.302875
[11,]  1.337579 -1.906407
[12,]  2.958263 -3.133425
[13,]  2.156720 -4.153809
[14,]  4.209870 -3.587902
[15,]  1.607313 -3.908292
[16,]  3.895723 -2.664850
[17,]  4.699163 -3.278570
[18,]  2.880225 -3.722995
[19,]  3.919782 -4.219083
[20,]  3.762063 -3.250798
[21,]  1.988400 -3.634352
[22,]  3.288278 -3.035885
[23,]  1.976397 -5.371975
[24,]  3.392614 -3.033089
[25,]  4.571973 -3.557470
[26,]  4.978979 -4.317606
[27,]  3.685530 -3.878485
[28,]  0.944076 -2.716172
[29,]  3.304277 -1.927053
[30,]  2.698474 -2.811844
[31,] -2.811844  2.698474
[32,] -1.927053  3.304277
[33,] -2.716172  0.944076
[34,] -3.878485  3.685530
[35,] -4.317606  4.978979
[36,] -3.557470  4.571973
[37,] -3.033089  3.392614
[38,] -5.371975  1.976397
[39,] -3.035885  3.288278
[40,] -3.634352  1.988400
[41,] -3.250798  3.762063
[42,] -4.219083  3.919782
```

```
[43,] -3.722995  2.880225
[44,] -3.278570  4.699163
[45,] -2.664850  3.895723
[46,] -3.908292  1.607313
[47,] -3.587902  4.209870
[48,] -4.153809  2.156720
[49,] -3.133425  2.958263
[50,] -1.906407  1.337579
[51,] -3.302875  3.633930
[52,] -4.160339  2.391975
[53,] -3.762414  4.277852
[54,] -2.294419  1.763849
[55,] -3.757341  1.366379
[56,] -3.116287  2.703410
[57,] -4.132507  3.509826
[58,] -2.059953  4.389818
[59,] -2.966405  2.140252
[60,] -3.964540  3.579611
```

```
plot(temperature)
```



```r
k <- kmeans(temperature, centers=2, nstart = 20)
k
```

```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
        tmp      tmp1
1  3.067087 -3.387572
2 -3.387572  3.067087

Clustering vector:
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 53.19723 53.19723
 (between_SS / total_SS =  92.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```r
attributes(k)
```

```
$names
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"

$class
[1] "kmeans"
```

```r
#What are the cluster centers
k$centers
```

```
        tmp      tmp1
1  3.067087 -3.387572
2 -3.387572  3.067087
```

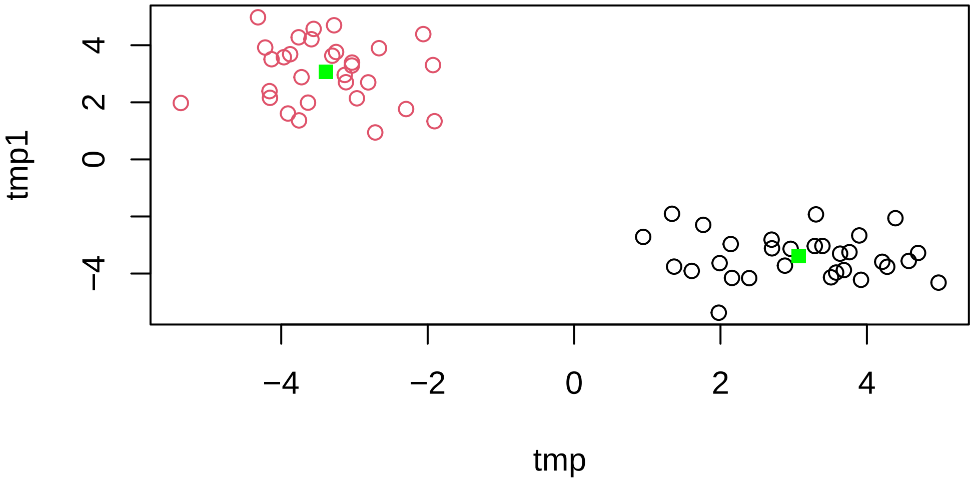
```
#Whats my clustering result

k$cluster
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Plot data as ‘x’ showing your clustering result and the center point for each cluster?

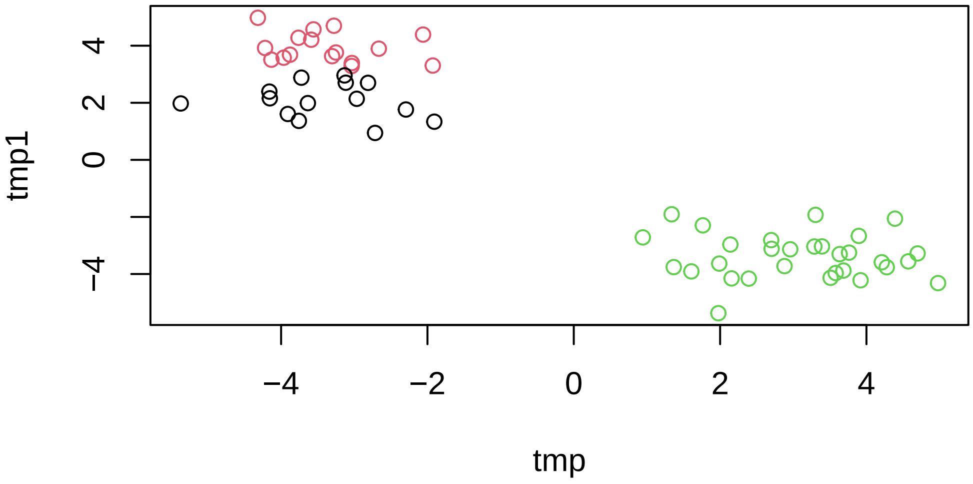
```
plot( temperature, col=k$cluster)
points(k$centers, pch=15, col="green")
```



```
#recycle property, allows you to make new data
```

```
#Run kmeans and cluster into 3 groups, plot

k3 <- kmeans(temperature, centers = 3, nstart = 20)
plot(temperature, col=k3$cluster)
```



```
k$tot.withinss
```

```
[1] 106.3945
```

```
k3$tot.withinss
```

```
[1] 80.04027
```

Big limitation of kmeans is that it imposes structure on your data that you ask for in the first place.

#Heirarchical Clustering

The main function in “base” R for this is called ‘hclust()’ It wants a distance matrix as input not the data itself

We can calculate a distance matrix in lots of different ways bit here we will use the ‘dist()’ function.

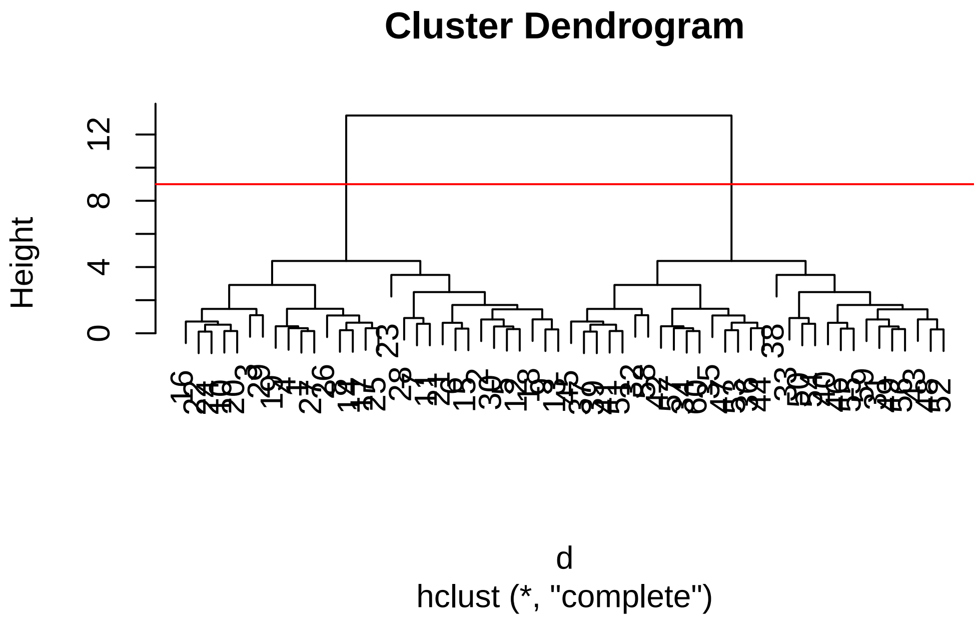
```
d <- dist(temperature)
hc <- hclust(d)
```

```
hc
```

```
Call:
hclust(d = d)

Cluster method   : complete
Distance         : euclidean
Number of objects: 60
```

```
plot(hc)
abline(h=9, col="red")
```



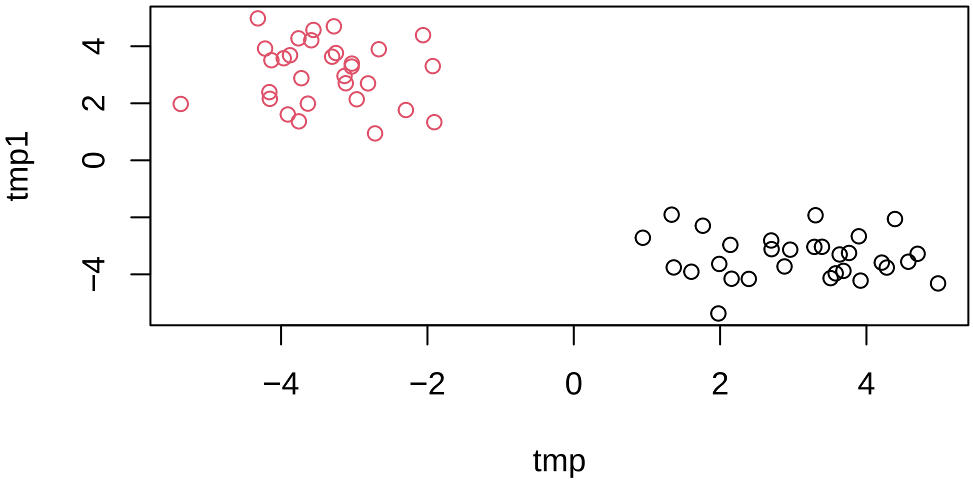
There is a special plot method for hclust obkects. Let’s see it.

To get the cluster memership vector we need to “cut” the tree at a given height that we [ick]

```
grps <- cutree(hc, k=2)
grps
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
plot(temperature, col=grps)
```



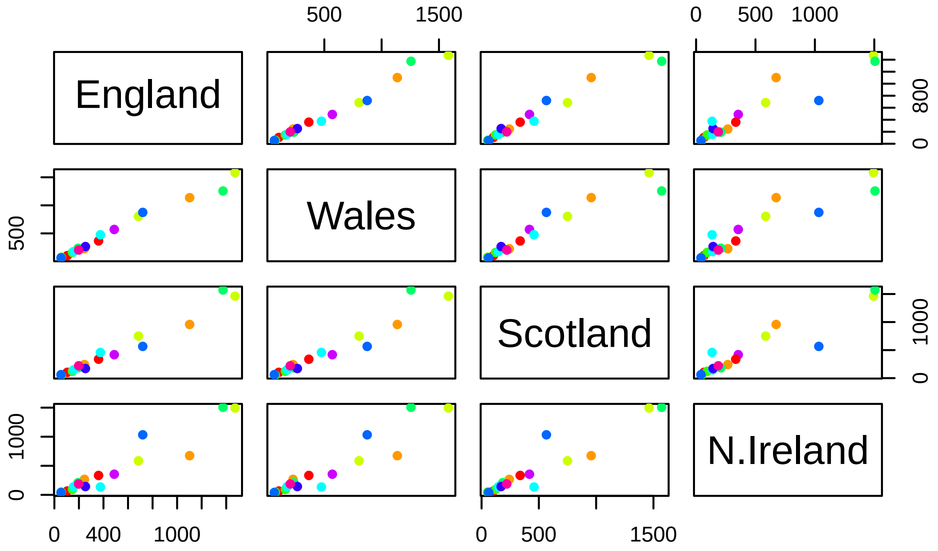
Principle Component Analysis

```
url <- "https://tinyurl.com/UK-foods"
x1 <- read.csv(url, row.names = 1)
```

```
head(x1)
```

```
               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
```

```
#One useful plot in this case (beause we only have 4 countries to look accross) is a "pair
pairs(x1, col=rainbow(10), pch=16)
```



##Enter PCA

The main function to do a PCA in “base” R is called ‘prcomp’ It wants our foods as the columns and the countries as the rows. It basically wants the transpose of the table.

```
#transpose with t(x)

pca <- prcomp(t(x1))
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 2.921e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

```
attributes(pca)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

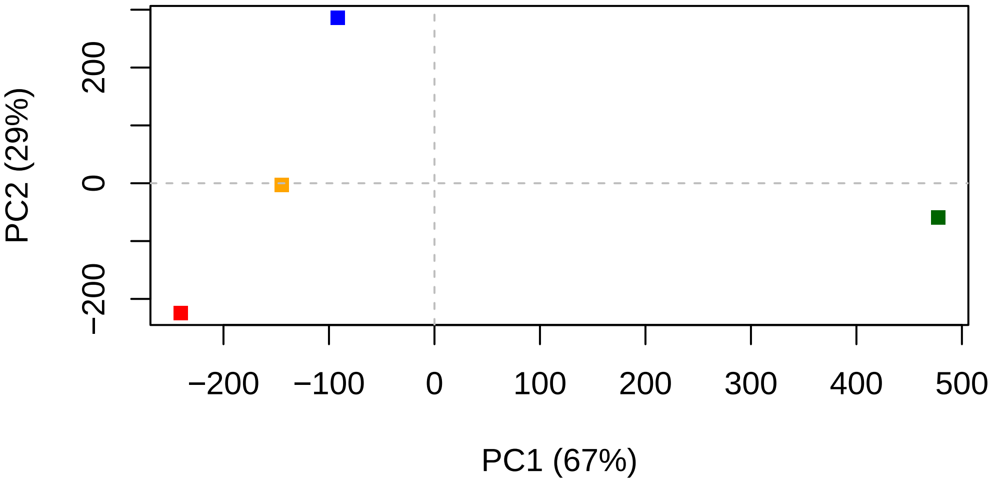
```
pca$x1
```

```
NULL
```

```
pca$rotation
```

```
                           PC1          PC2         PC3          PC4
Cheese             -0.056955380  0.016012850  0.02394295 -0.409382587
Carcass_meat        0.047927628  0.013915823  0.06367111  0.729481922
Other_meat         -0.258916658 -0.015331138 -0.55384854  0.331001134
Fish               -0.084414983 -0.050754947  0.03906481  0.022375878
Fats_and_oils      -0.005193623 -0.095388656 -0.12522257  0.034512161
Sugars             -0.037620983 -0.043021699 -0.03605745  0.024943337
Fresh_potatoes      0.401402060 -0.715017078 -0.20668248  0.021396007
Fresh_Veg          -0.151849942 -0.144900268  0.21382237  0.001606882
Other_Veg          -0.243593729 -0.225450923 -0.05332841  0.031153231
Processed_potatoes -0.026886233  0.042850761 -0.07364902 -0.017379680
Processed_Veg      -0.036488269 -0.045451802  0.05289191  0.021250980
Fresh_fruit        -0.632640898 -0.177740743  0.40012865  0.227657348
Cereals            -0.047702858 -0.212599678 -0.35884921  0.100043319
Beverages          -0.026187756 -0.030560542 -0.04135860 -0.018382072
Soft_drinks         0.232244140  0.555124311 -0.16942648  0.222319484
Alcoholic_drinks   -0.463968168  0.113536523 -0.49858320 -0.273126013
Confectionery      -0.029650201  0.005949921 -0.05232164  0.001890737
```

```
plot(pca$x[, 1], pca$x[,2],  xlab = "PC1 (67%)", ylab="PC2 (29%)",
     col=c("orange", "red", "blue", "darkgreen"), pch=15)
abline(h=0, col="gray", lty=2)
abline(v=0, col="gray", lty=2)
```


PC2 (29%)
200
0
−200
−200
−100
0
100
200
300
400
500
PC1 (67%)