Class 07 - Machine learning

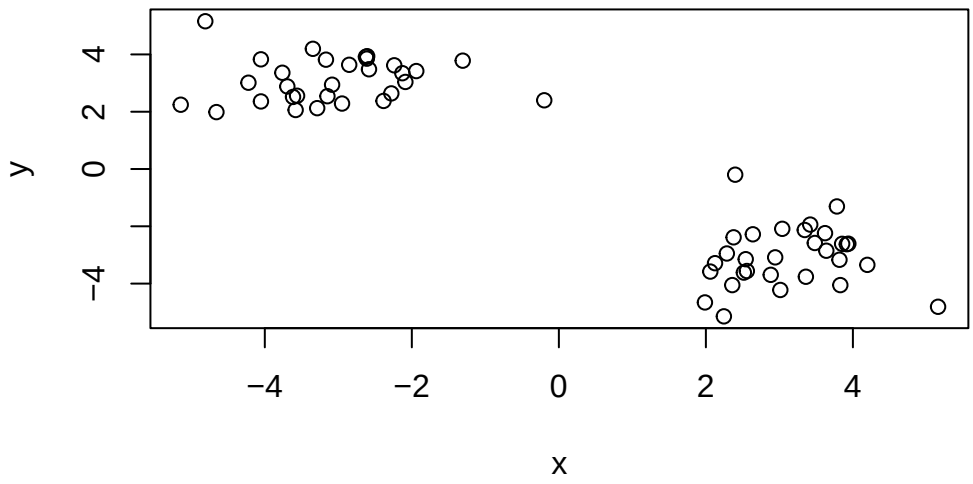
Julia Ainsworth

Table of contents

#K means clustering

Making up some data to cluster

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



Notes on funtions we're using today:

`c()` concatenates, makes vectors `cbind()` also makes vectors `rnorm()` makes a normal distribution centered around the argument given for mean

The function for k means clustering in base R is called `kmeans(x, centers = k)`. We give input data for the clustering and the number of clusters we want is "centers" (the k of kmeans)

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

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

Cluster means:
          x         y
1  3.110866 -3.066073
2 -3.066073  3.110866

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 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] 49.39441 49.39441
 (between_SS / total_SS =  92.1 %)

Available components:

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

To extract an element from km, see below. For clusterr size:

```
km$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 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
```

```
km
```

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

Cluster means:
          x         y
1  3.110866 -3.066073
2 -3.066073  3.110866

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 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] 49.39441 49.39441
 (between_SS / total_SS =  92.1 %)

Available components:

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

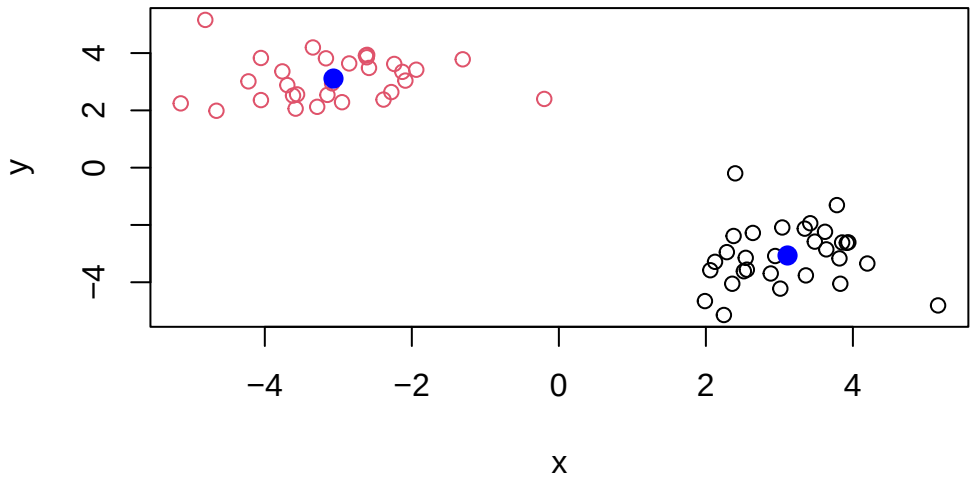
For cluster center:

```
km$centers
```

```
          x         y
1  3.110866 -3.066073
2 -3.066073  3.110866
```

Q: Plot x colored by the kmeans cluster assignment and add cluster centers as blue points. Answer: we have a vector of cluster membership, `km$cluster` which assigns colors based on whether it's in either cluster 1 or 2. To get the center, we use the `points` function which can also do `pch` = for different plotting characters. 16 is a closed circle, so it will stand out more against the open circles of our other data. `Cex` makes it stand out more; default is 1 and anything larger than 1 makes it stand out more. Note: can put NAMED arguments in any order.

```
plot(x, col = km$cluster)
points(km$centers, col ="blue", pch =16, cex = 1.4)
```



#Hierarchical Clustering

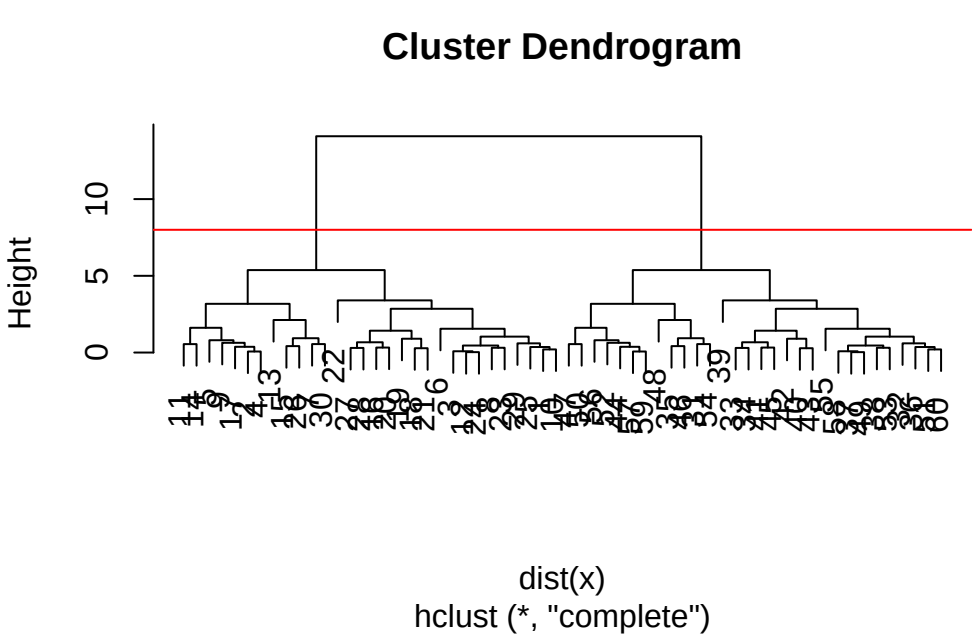
The `hclust()` function performs hierarchical clustering.The advantage over `kclust()` is you don't need to tell it "k"the number of clusters … It will take the distance matrix, d, as input, instead of the original data. This makes it more flexible.

```
hc <- hclust( dist(x) )
hc
```

```
Call:
hclust(d = dist(x))

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

```
plot(hc)
abline(h=8, col="red")
```



To get the "main" result (cluster membership), I want to "cut" this tree to yield "branches" whose leaves are the members of the cluster.The argument h specifies where to cut

```
cutree(hc, h = 8)
```

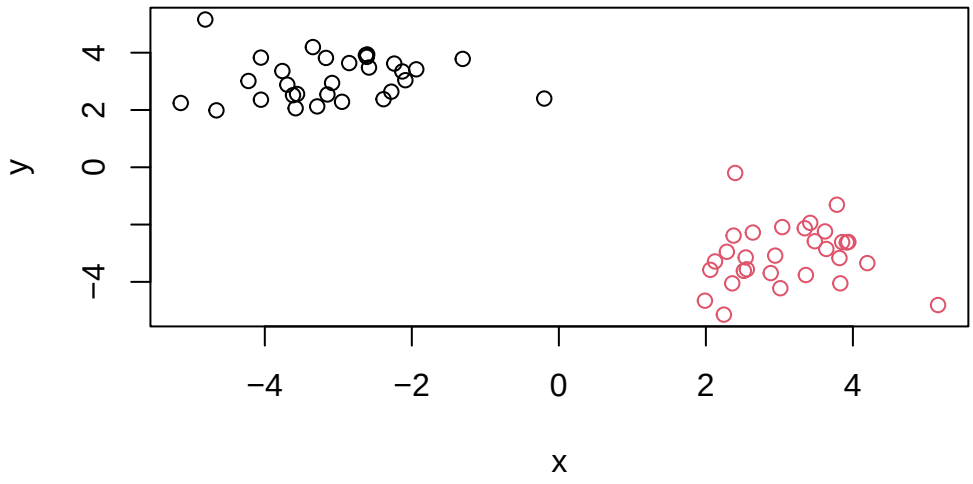
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

More often, we will use `cutree()` with k=2 for example. This gives us the same result; however, it specifies the number of groups rather than having us determine it by looking at the dendrogram.

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

Make a plot of `hclust()` results colored by the hierarchical clustering results

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



Principal component analysis (PCA)

Note: PC1 and PC2 are always orthogonal

UK food data, with amended row names:

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

Examining the data:

```
dim(x)
```

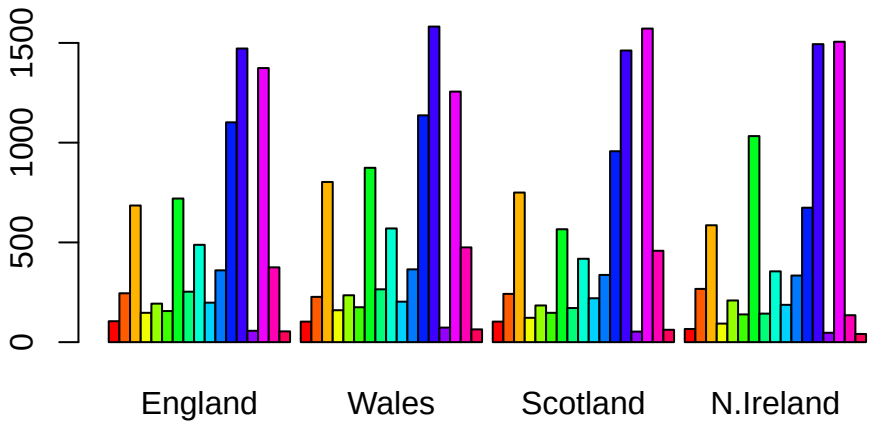
```
[1] 17  4
```

```
head(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
```

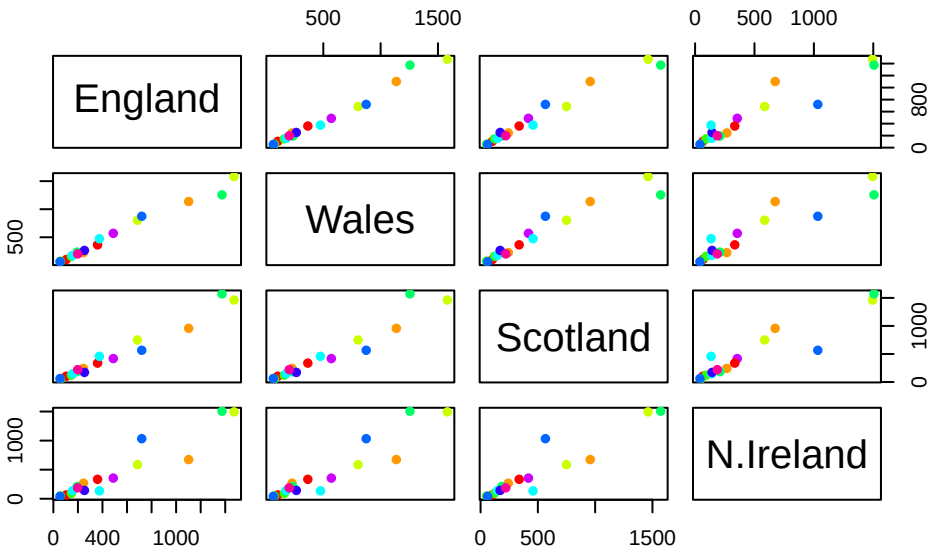
Examining the data using bar plots: Removing barplot(beside = FALSE) changes the bars to be on top of each other. Doesn't make much sense!

```
barplot(as.matrix(x), beside = T, col=rainbow(nrow(x)))
```



For Q5: A pairs plot can be a little useful; it does each possible pairwise comparison (Eng vs Wales, Eng vs Scotland, Eng vs N Ireland)

```
pairs(x, col=rainbow(10), pch=16)
```



This generates a very confusing series of plots!

PCA time!!

Main function in base R to do PCA is called `prcomp()` Annoying thing: it expects the transpose (it will want the countries as rows, and food/the differences to be columns) of our data as input

```
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
```

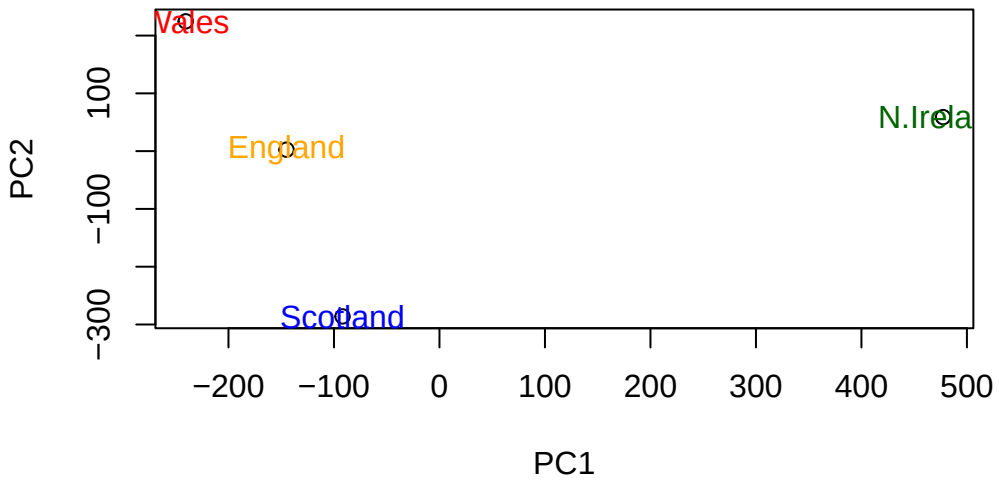
Q7: The object returned by `prcomp()` has our results that include a $x component. This is our "scores" along the PCs (i.e. the plot of our data along the new PC axis)

```
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
```

To get a plot of PC1 vs PC2, use the locations in the set:

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", )
text(pca$x[,1], pca$x[,2], colnames(x), col=c("orange", "red", "blue", "darkgreen"))
```



End of Wednesday's class 10/19/22

Lets focus on PC1 as it accounts for $> 90\%$ of variance

```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```