Class07

Whitney Tran (PID:A16781338)

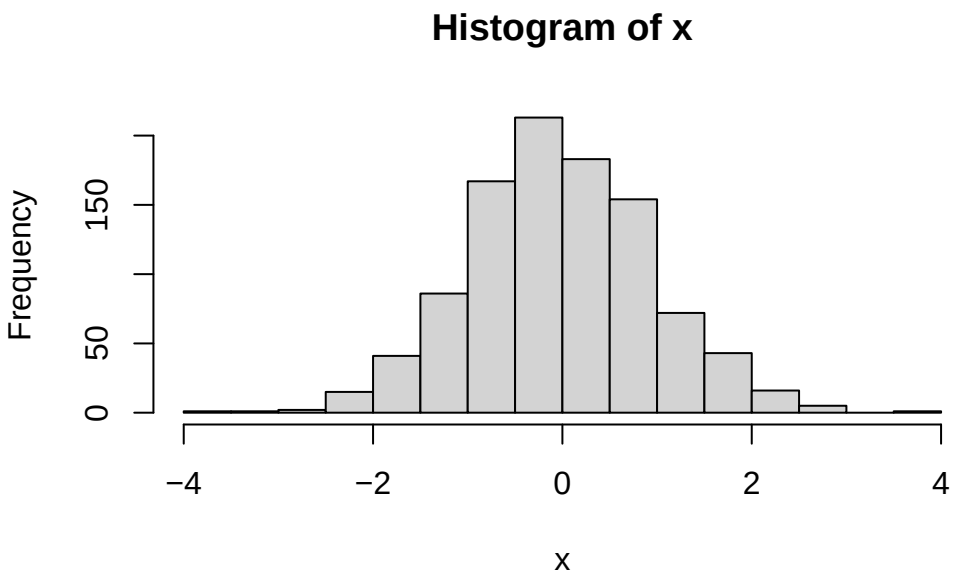
#Clustering Methods

The broad goal here is to find groupings (clusters) in your input data.

Kmeans

First, let's make up some data to cluster.

```
x<- rnorm(1000)
hist(x)
```



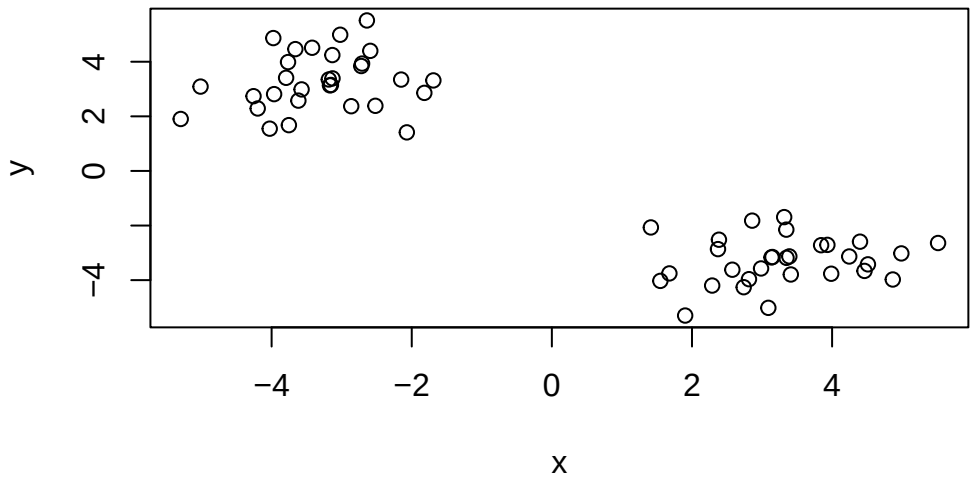
Make a vector of length 60 with 30 points centered at -3 and 30 points centered at +3

```
tmp<- c((rnorm(30, mean=-3)), rnorm(30, mean=3))
tmp
```

```
 [1] -2.519137 -1.691260 -3.020916 -3.130596 -5.014375 -3.661306 -2.069922
 [8] -2.863529 -3.166662 -3.964357 -5.297152 -3.765360 -2.151063 -3.133311
[15] -3.792748 -2.591588 -2.708606 -3.572162 -3.183422 -3.975972 -3.153007
[22] -3.617933 -3.753429 -4.027048 -4.258067 -4.198324 -3.421449 -2.640146
[29] -2.720184 -1.820102  2.859160  3.843241  5.512505  4.511597  2.288104
[36]  2.738314  1.548476  1.677711  2.575877  3.148045  4.866057  3.349552
[43]  2.985948  3.932499  4.398262  3.410736  4.244749  3.346391  3.987264
[50]  1.902701  2.813615  3.134671  2.371212  1.413074  4.462187  3.089716
[57]  3.389384  4.988267  3.315882  2.385038
```

I will now make a wee x and y dataset with 2 groups of points.

```
x<- cbind(x=tmp,y=rev(tmp))
plot(x)
```



```
k <- kmeans(x, centers=2)
k
```

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

Cluster means:
          x         y
1  3.283008 -3.296104
2 -3.296104  3.283008

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] 52.98615 52.98615
 (between_SS / total_SS =  92.5 %)

Available components:

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

Q. From your result object `k`, how many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What "component" of your result object details the cluster membership?

```
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 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
```

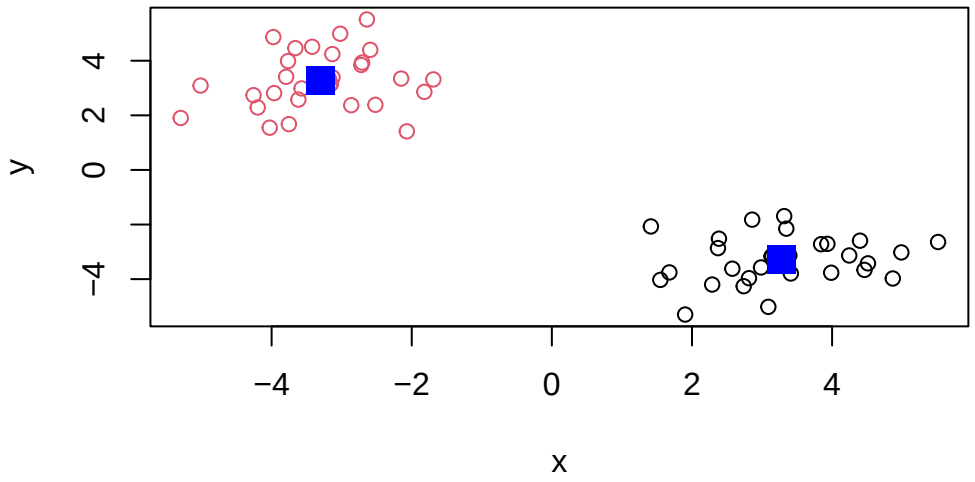
Q. cluster centers?

```
k$centers
```

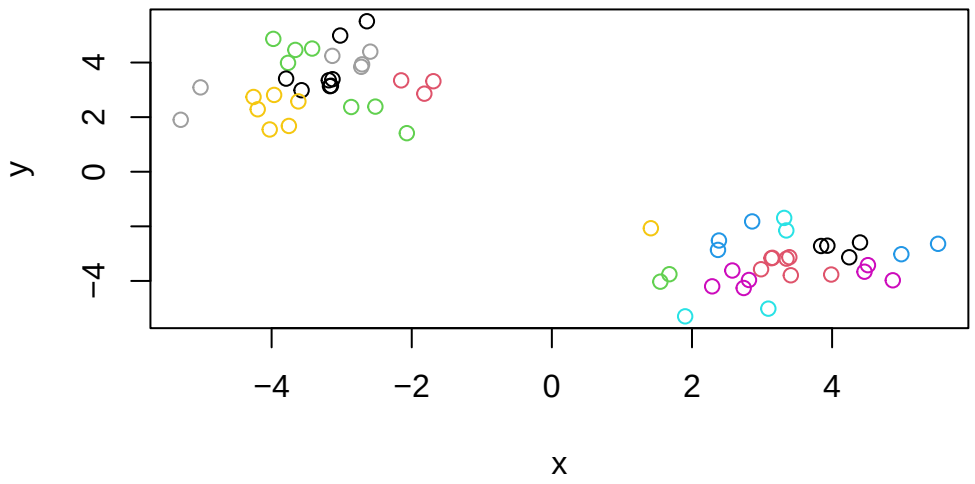
```
          x         y
1  3.283008 -3.296104
2 -3.296104  3.283008
```

Q. Plot of our clustering results.

```
plot(x, col=k$cluster)
points(k$centers, col="blue", pch=15, cex=2)
```




```
# kmeans
k4 <- kmeans(x, center=20)
#plot
plot(x, col=k4$cluster)
```



A big limitation of kmeans is that it does what you ask even if you ask for silly clusters.

Hierarchical Clustering

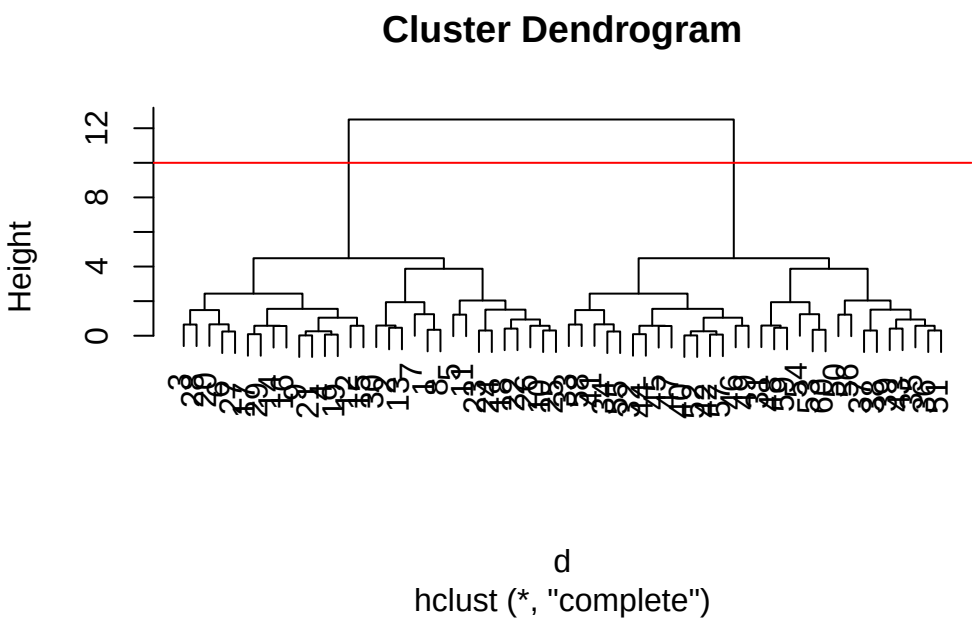
The main base R function for Hierarchical Clustering is `hclust()`. Unlike `kmeans()`, you can not just pass it your data as input. You first need to calculate a distance matrix.

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

```
Call:
hclust(d = d)

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

Use `plot()`to view results.

```
plot(hc)
abline(h=10, col="red")
```



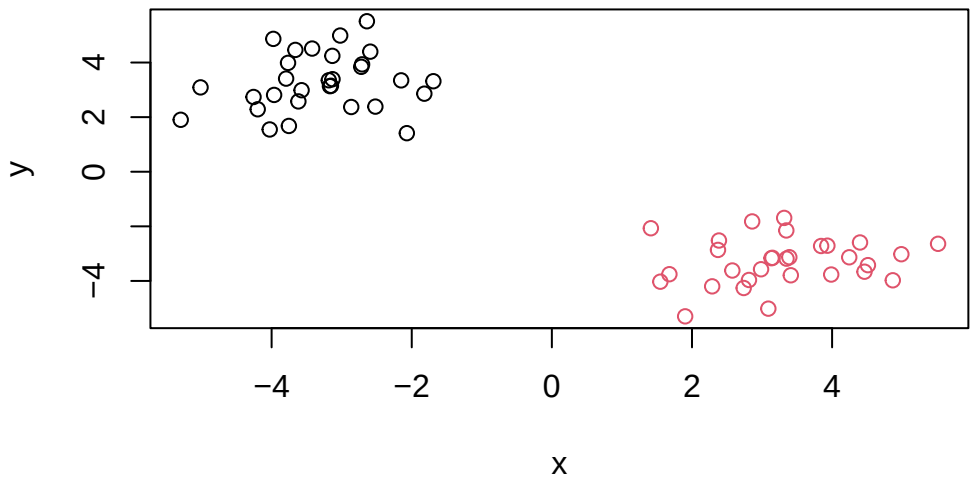
To make the "cut" and get our cluster membership vector we can use the `cutree()` function.

```
grps<- cutree(hc, h=10)
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 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
```

Make a plot of our data colored by hclust results.

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



#Principal Component Analysis (PCA)

Here we will do Principal Component Analysis (PCA) on some food data from the UK.

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

```
#rownames(x) <-x[,1]
#x<-x[,-1]
#x
```

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

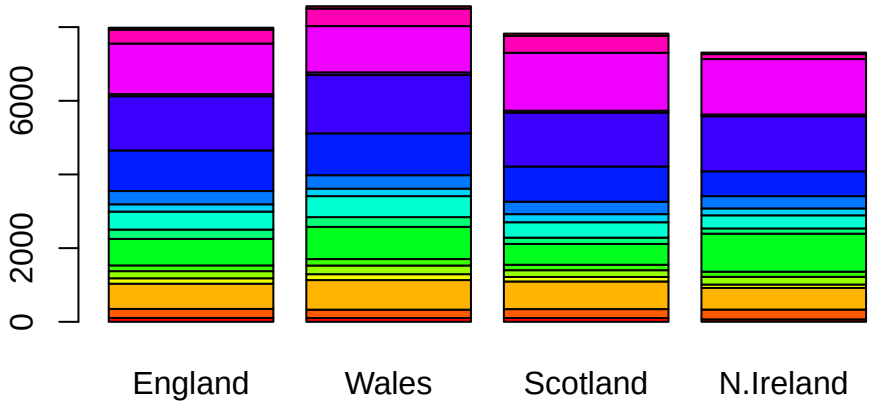
- There are 5 rows and 17 columns in the data frame. `ncol()` and `nrow()` can be used to find this.

Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

- I personally prefer adding `row.names=1` to the `read.csv()` function. This is better because running x <- x[,-1] would cause some data to disappear everytime you run it.

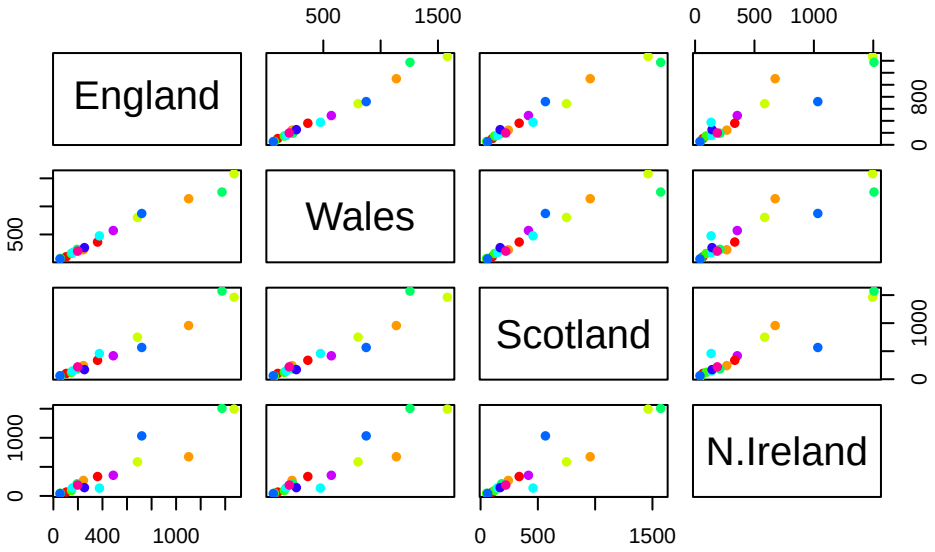
Q3: Changing what optional argument in the above barplot() function results in the following plot?

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



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?

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



- If a point lies on the diagonal,

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

- N.Ireland has more potatoes and less alcohol.

##PCA to the rescue

The main "base" R function for PCA is called `prcomp()`.

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

```
Importance of components:
                            PC1      PC2      PC3        PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

How much variange is captured in 2 PCs?

- 96.5%

Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

To make our main "PC score plot" (a.k.a "PC1 vs PC2 plot", or "PC plot" or "ordination plot").

```
attributes(pca)
```

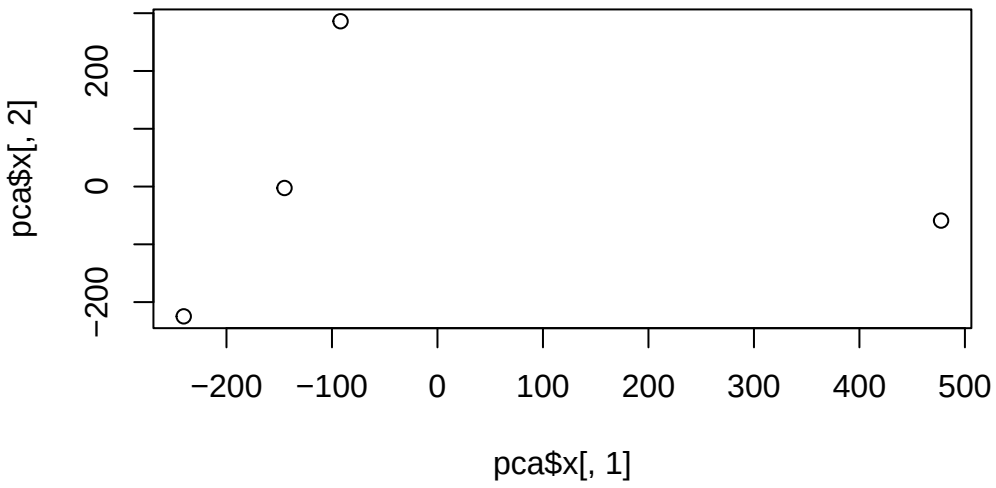
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

We are after the `pca$x` component to make our main PCA plot.

```
pca$x
```

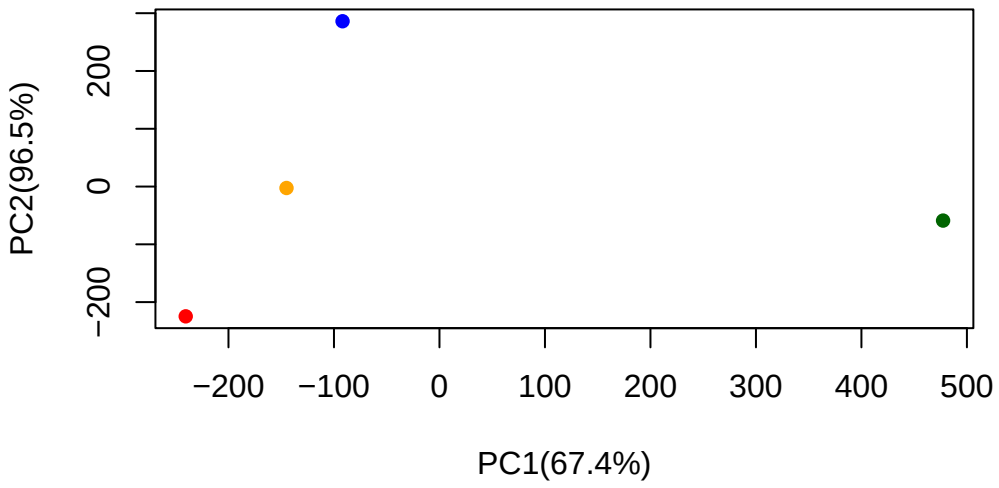
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
plot(pca$x[,1], pca$x[,2])
```



Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
mycols<- c("orange","red", "blue","darkgreen")

plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, xlab="PC1(67.4%)", ylab="PC2(96.5%)")
```



Another important result from PCA is how the original variables (in this case, the foods)contribute to the PCs. This is contained in the `pca$rotation` object - folks often call this the "loadings" or "contributions" to the PCs.

```
pca$rotation[,1]
```

```
          Cheese       Carcass_meat         Other_meat               Fish
-0.056955380         0.047927628        -0.258916658        -0.084414983
Fats_and_oils              Sugars     Fresh_potatoes          Fresh_Veg
-0.005193623        -0.037620983         0.401402060        -0.151849942
   Other_Veg  Processed_potatoes       Processed_Veg         Fresh_fruit
-0.243593729        -0.026886233        -0.036488269        -0.632640898
      Cereals            Beverages        Soft_drinks    Alcoholic_drinks
-0.047702858        -0.026187756         0.232244140        -0.463968168
Confectionery
-0.029650201
```

We can make a plot along PC1.

```
library(ggplot2)
```

```r
contrib <- as.data.frame(pca$rotation)

ggplot(contrib)+ aes(PC1, rownames(contrib))+ geom_col()
```