# Class 7: Clustering and PCA

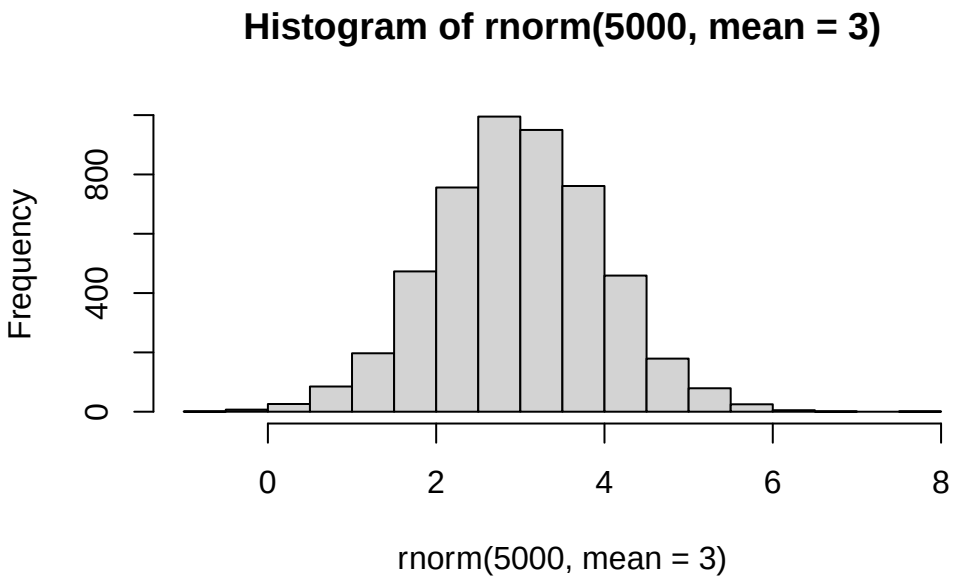
Juliane Kwong

## Clustering

First let's make up some data to cluster so we can get a feel for these methods and how to work with them.

We can use the `rnorm()`function to get random numbers from a normal distribution around a given `mean`.

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



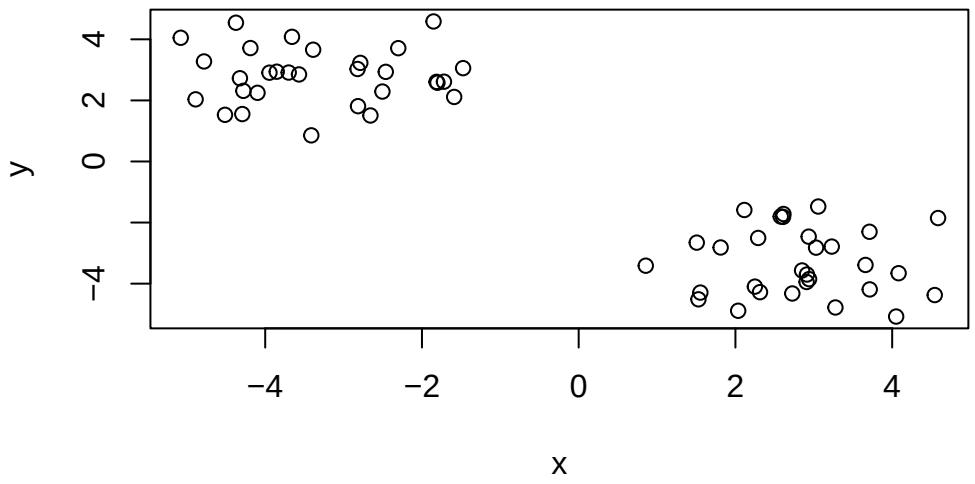

Let's get 30 points with a mean of 3 and another 30 with a mean of -3..

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

```
 [1]  3.0566104  2.5781116  3.7135582  1.5064755  3.2757198  1.5280125
 [7]  4.0507672  3.0287955  1.8110969  2.1135601  2.3132858  4.5856154
[13]  1.5525997  3.6595405  2.7254682  2.8504142  3.7107445  2.6077051
[19]  2.0348070  0.8557964  2.9398113  4.0819152  2.6123269  4.5430960
[25]  2.9081993  2.2484894  2.9124833  2.9346973  2.2897084  3.2288211
[31] -2.7866920 -2.5044566 -2.4617525 -3.7003606 -4.0982399 -3.9458644
[37] -4.3735618 -1.7193320 -3.6585212 -3.8527606 -3.4121382 -4.8880128
[43] -1.8138991 -2.3019296 -3.5687177 -4.3226044 -3.3882032 -4.2923951
[49] -1.8522271 -4.2788012 -1.5887119 -2.8157393 -2.8213901 -5.0779560
[55] -4.5132037 -4.7809367 -2.6571841 -4.1889714 -1.8024757 -1.4748767
```

Put two of these together:

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



### K-means clustering.

Very popular clustering method that we can use with the `kmeans()` function in base R.

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

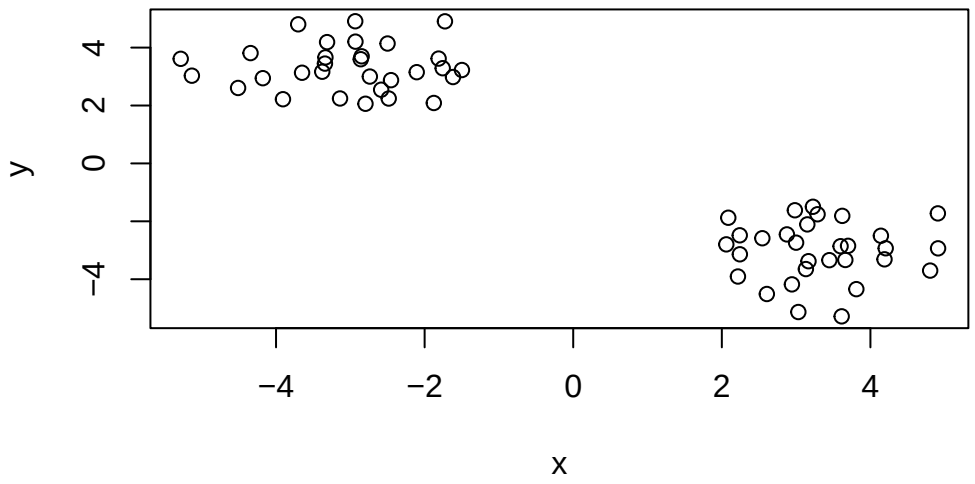
```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
          x         y
1 -3.298064  2.808608
2  2.808608 -3.298064

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

Within cluster sum of squares by cluster:
[1] 58.81603 58.81603
 (between_SS / total_SS =  90.5 %)

Available components:

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

Clustering vector helps to identify where a specific data point is in the plot.

```
km$size
```

```
[1] 30 30
```

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




```
km<-kmeans(x,center=2)
```

Q. How many points are in each cluster?

Q. What 'component' of your result object details - cluster size?

```
km$size
```

```
[1] 30 30
```

- cluster assignment?

```
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
```

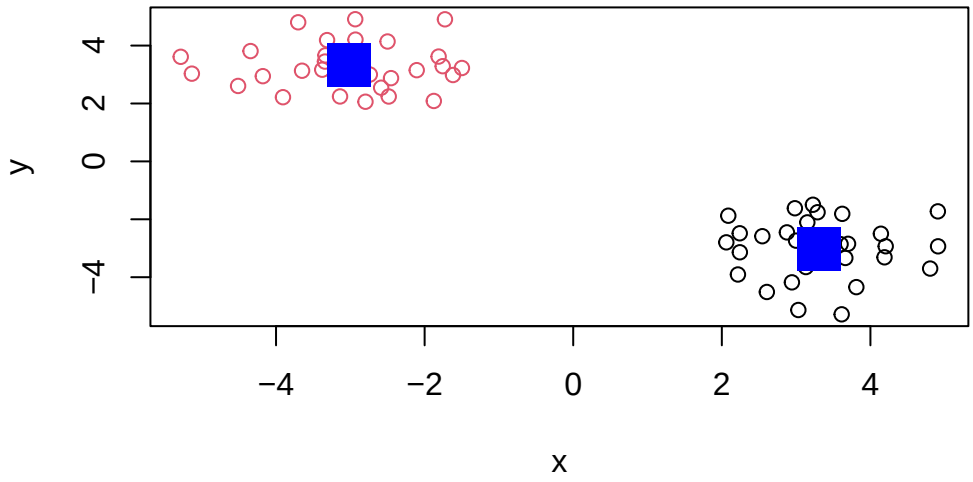
- cluster center?

```
km$centers
```

```
          x         y
1  3.313479 -3.024258
2 -3.024258  3.313479
```

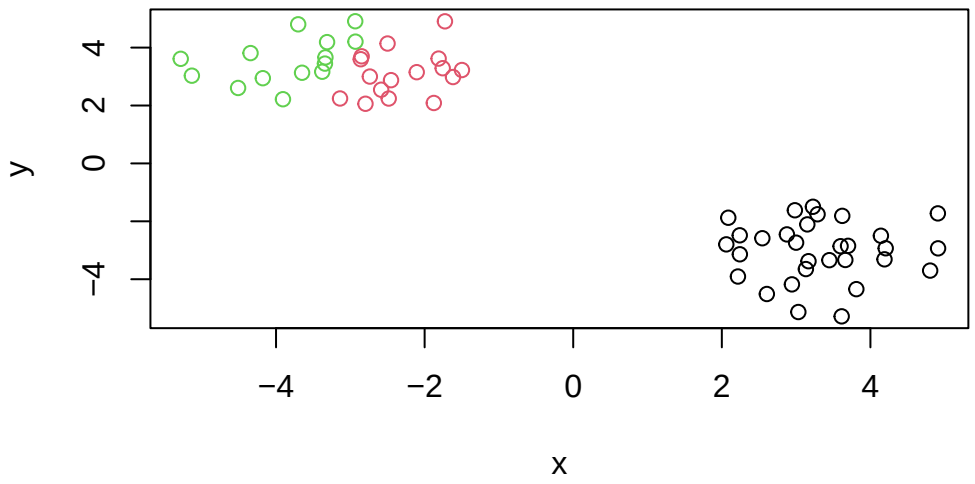
Q. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points.

```
plot(x,col=km$cluster)
points(km$centers, col="blue",pch=15,cex=3)
```




Q. Let's cluster into 3 groups or same `x` data and make a plot.

```
km<-kmeans(x,center=3)
plot(x,col=km$cluster)
```




## Hierarchical CLustering

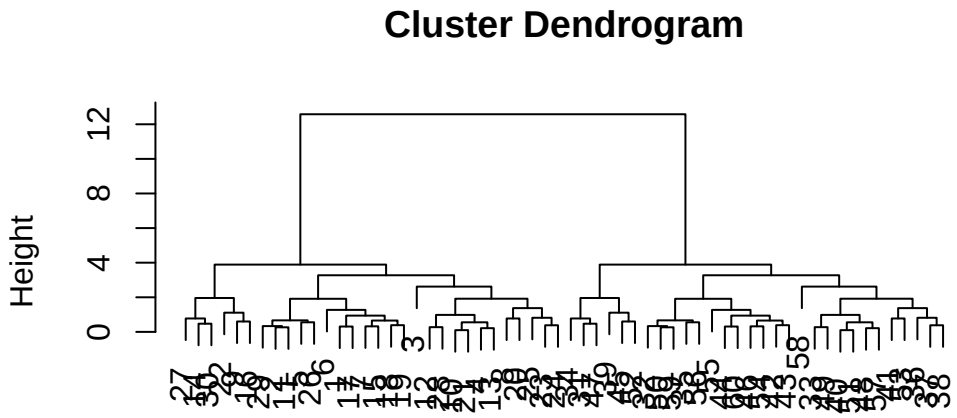
We can use the `hclust()` function for Hierarchical Clustering. Unlike `kmeans()`, where we could just pass in our data as input, we need to give `hclust()` a "distance matrix".

We will use the `dist()` function to start with.

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

```
Call:
hclust(d = d)

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

```
plot(hc)
```




I can now "cut" my tree with the `cutree()` to yield a cluster membership vector.

```
grps<-cutree(hc,h=8)
grps
```

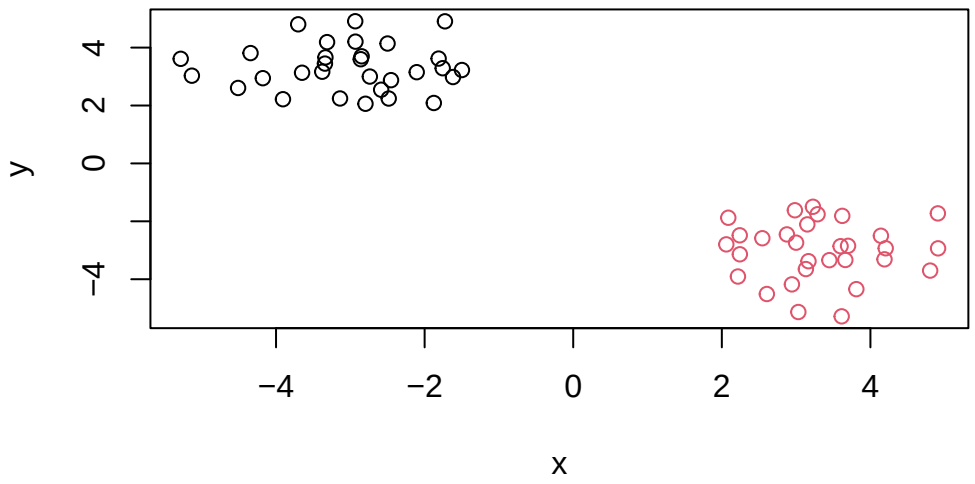
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also tell `cutree()` to cut where it yields "k" groups.

```
cutree(hc,k=2)
```

```
 [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 2
```

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




## Principal Component Analysis (PCA)

### PCA of UK food data

```
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 |

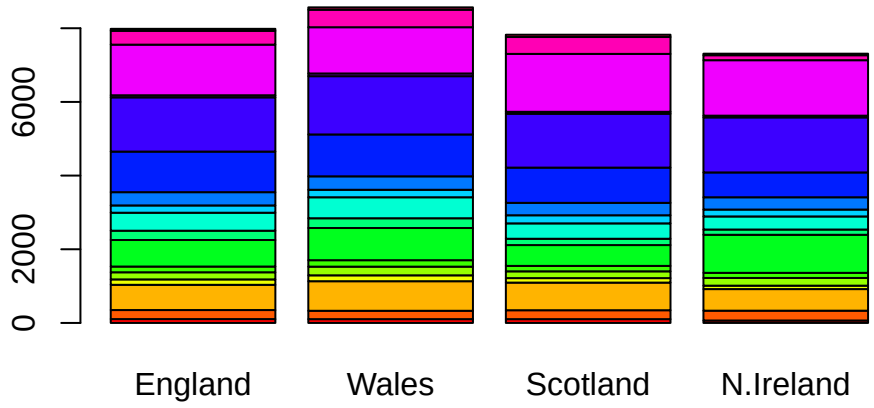
```
Fresh_fruit             1102  1137      957       674
Cereals                 1472  1582     1462      1494
Beverages                 57    73       53        47
Soft_drinks             1374  1256     1572      1506
Alcoholic_drinks         375   475      458       135
Confectionery             54    64       62        41
```

Exploratory analysis

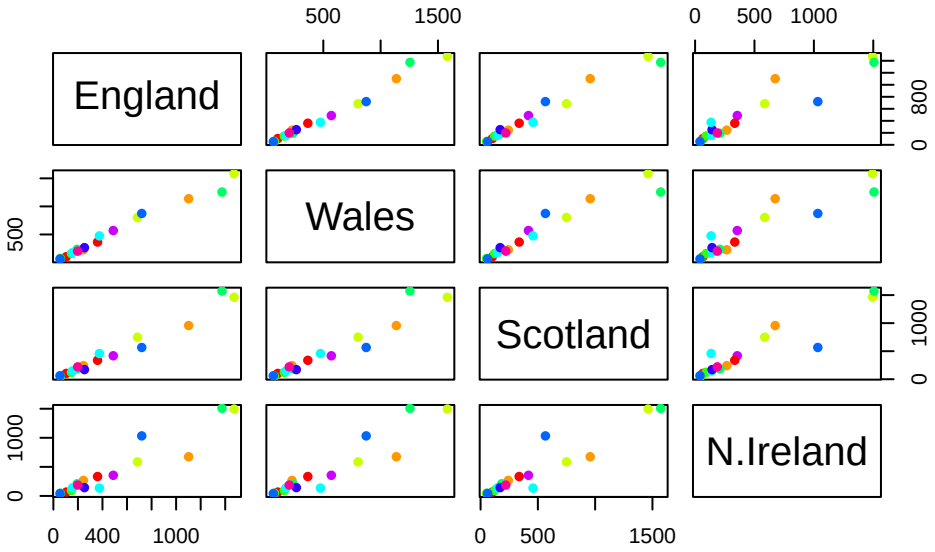
```
dim(x)
```

```
[1] 17  4
```

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




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




The main PCA function in base R is called `prcomp()` it expects the transpose (location) of our data.

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 5.552e-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$x
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  1.042460e-14
Wales     -240.52915  224.646925   56.475555  9.556806e-13
Scotland   -91.86934 -286.081786   44.415495 -1.257152e-12
N.Ireland  477.39164   58.901862    4.877895  2.872787e-13
```

```
plot(pca$x[,1], pca$x[,2],
     col=c("orange","red","blue","darkgreen"),
     pch=16)
```