

# Class 7: Machine Learning 1

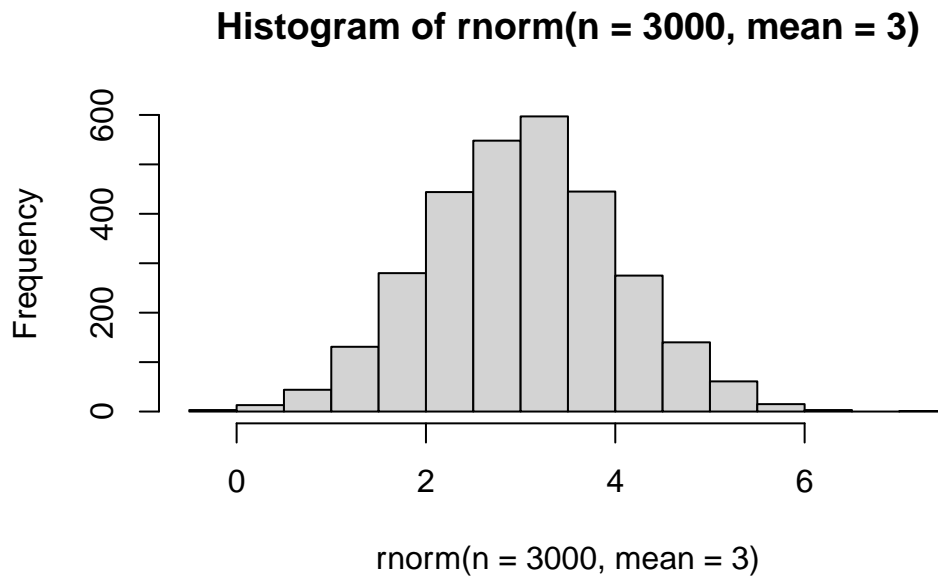
Jessica Gao (PID:A16939806)

Today we will explore unsupervised machine learning methods including clustering and dimensionality reduction methods.

Let's start by making up some data (where we know there are clear groups) that we can use to test out different clustering methods.

We can use the 'rnorm()' function to help us here

```
hist(rnorm(n=3000, mean=3))
```



make data with two “clusters”

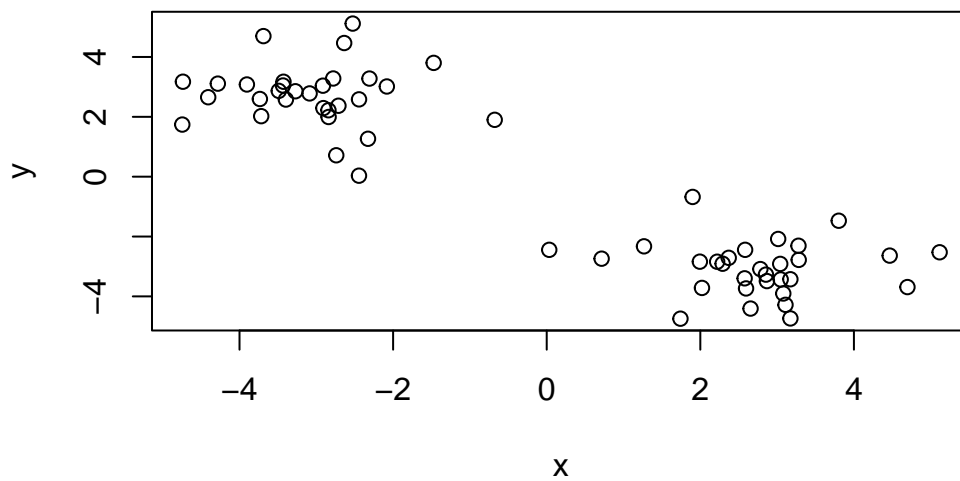
```
x <- c( rnorm(30, mean=-3),
        rnorm(30, mean=+3))

z <- cbind(x=x, y=rev(x))
head(z)
```

```
      x      y
[1,] -1.472038 3.80250435
[2,] -2.444566 0.03310359
[3,] -3.272625 2.85409446
[4,] -2.780015 3.28146196
[5,] -2.308026 3.27824801
[6,] -4.282349 3.10896174
```

```
#cbind = column binding
```

```
plot(z)
```



How big is 'z'?

```
nrow(z)
```

```
[1] 60
```

```
ncol(z)
```

```
[1] 2
```

## K-mean clustering

The main function in “base” R for k-means clustering is called ‘kmeans()’

```
k <- kmeans(z, centers=2)
#output: clustering with 2 clusters, because centers is set equal to 2
#cluster means: +3 and -3, this was set for z in the cbind() function above
k
```

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

Cluster means:

	x	y
1	2.724728	-3.065895
2	-3.065895	2.724728

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

Within cluster sum of squares by cluster:

```
[1] 55.63578 55.63578
(between_SS / total_SS = 90.0 %)
```

Available components:

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

```
attributes(k)
```

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

Q. How many points lie in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What component of our results tell us about the cluster membership (i.e. which point likes in which cluster)

```
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
```

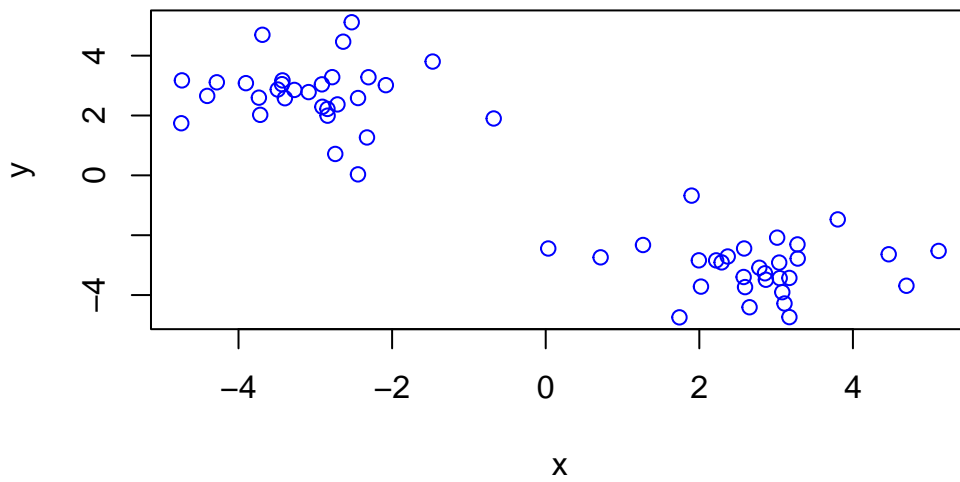
Q. Center of each cluster?

```
k$centers
```

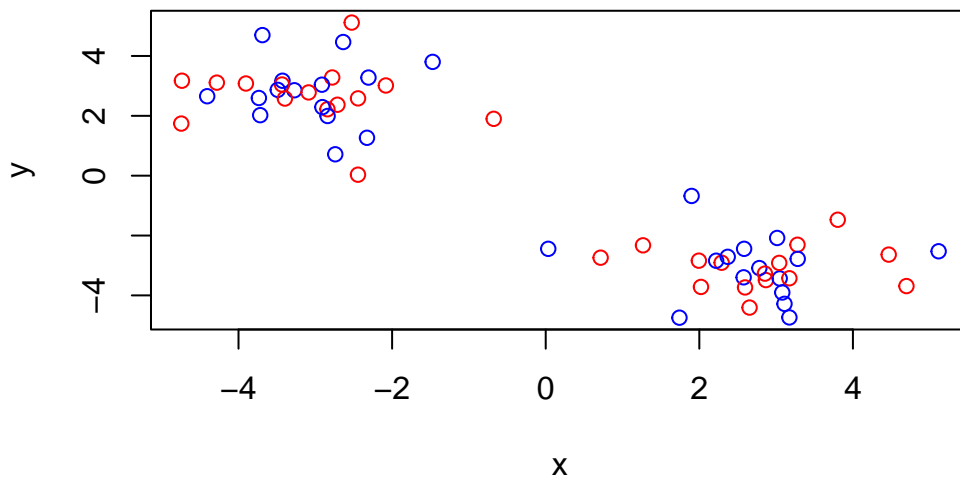
```
      x      y
1  2.724728 -3.065895
2 -3.065895  2.724728
```

Q. Put this result info together and make a little “base R” plot of our clustering result. Alos add teh cluster center points to this plot.

```
plot(z, col="blue")
```

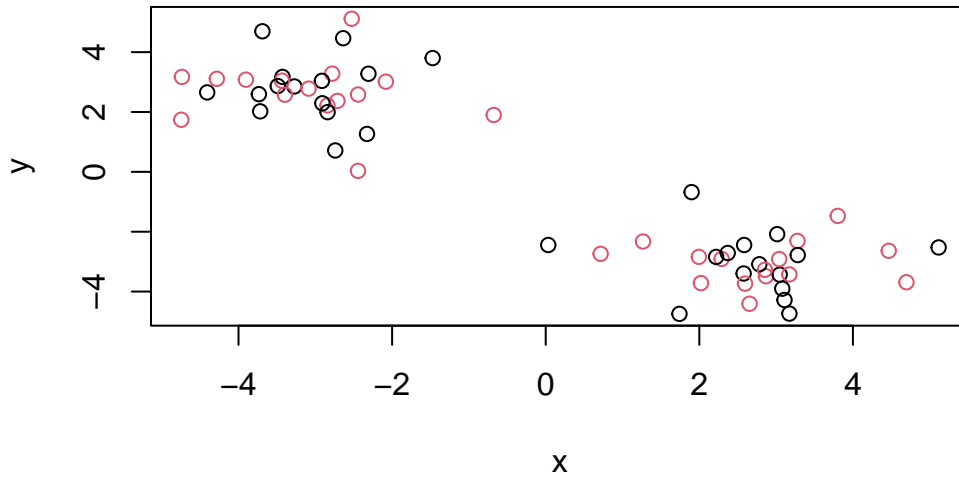


```
plot(z, col=c("blue","red"))
```



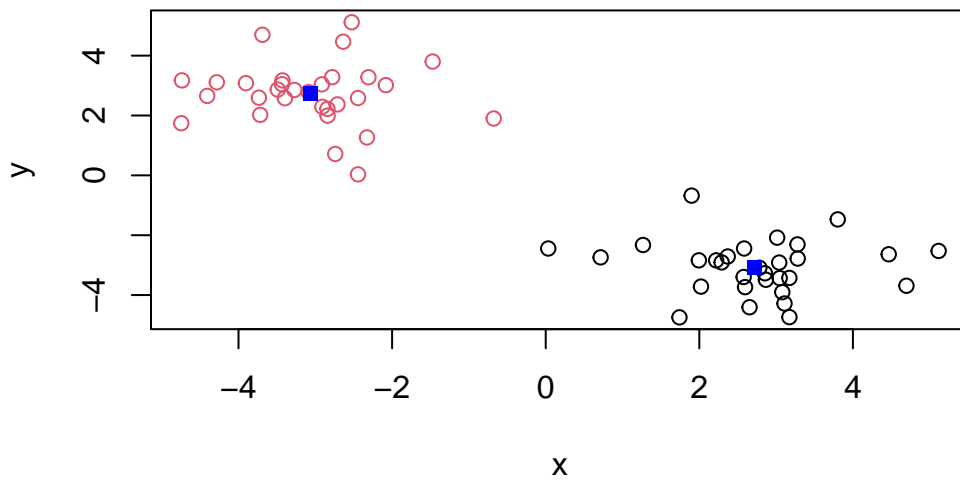
You can color by number

```
plot(z, col=c(1,2))
```



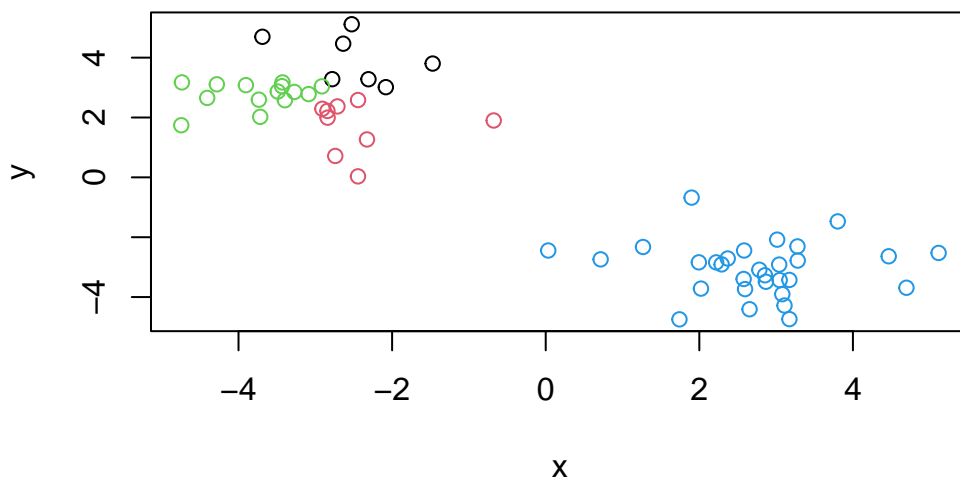
Plot colored by cluster membership:

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

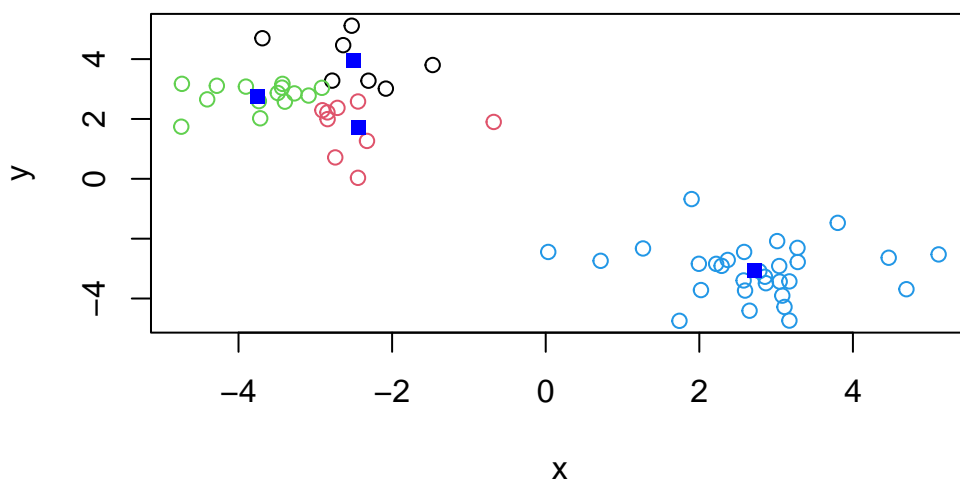


Q. Run kmeans on our input 'z' and define 4 clusters making the same results visualization plot as above (pot of z colored by cluster membership).

```
k4 <- kmeans(z, centers=4)
plot(z, col=k4$cluster)
```



```
plot(z, col=k4$cluster)
points(k4$centers, col="blue", pch=15)
```





Best clusters have the smallest ss value. More clusters = smaller ss value, you want the elbow point.

## Hierarchical clustering

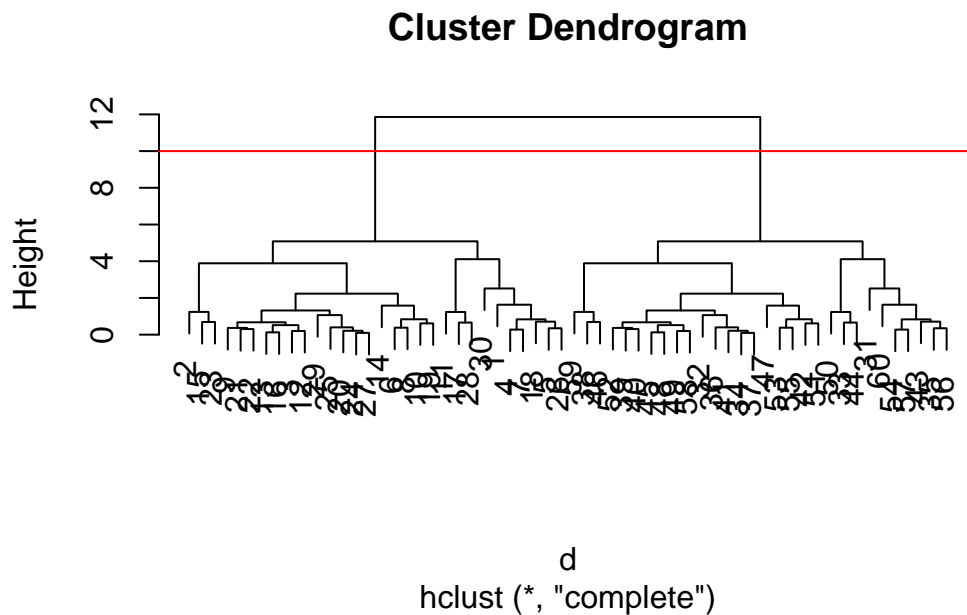
The main function in base R for this called 'hclust()' it will take as input a distance matrix (key point is that you can't just give your "raw" data as input - you have to first calculate a distance matrix from your data).

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

Call:  
hclust(d = d)

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

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

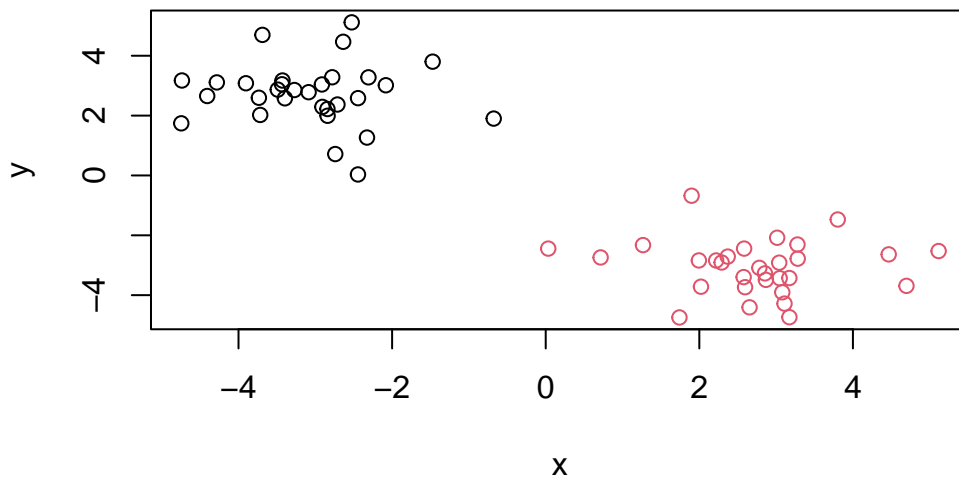


all the number below 30 is on one side, and above 30 on the other side Tells you how far apart each value is from each other, the closer ones to each other are clustered together.

Once I inspect the “tree” I can “cut” the tree to yield my groupings or clusters. The function to this is called ‘cutree()’

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

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



## Hands on with Principal Component Analysis (PCA)

Let's examine some silly 17-dimensional data detailing food consumption in the UK (England, Scotland, Wales, and N. Ireland). Are these countries eating habits different or similar and if so how?

###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

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?

```
row.names=1
```

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

```
nrow(x)
```

```
[1] 17
```

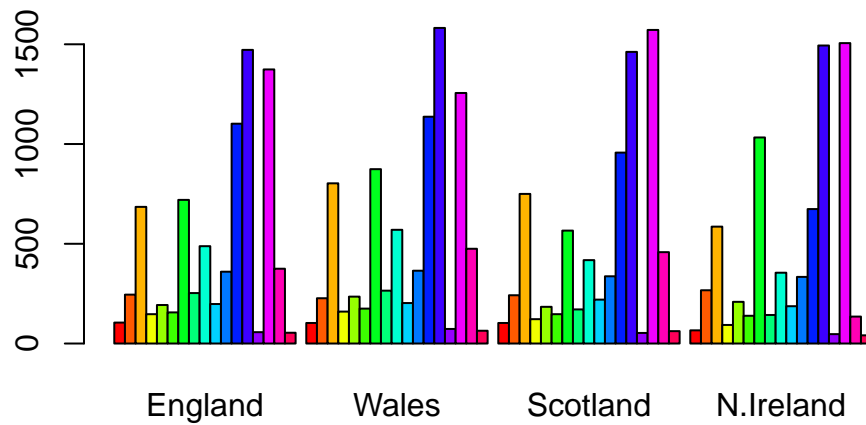
```
ncol(x)
```

```
[1] 4
```

```
dim(x) #gives row, column
```

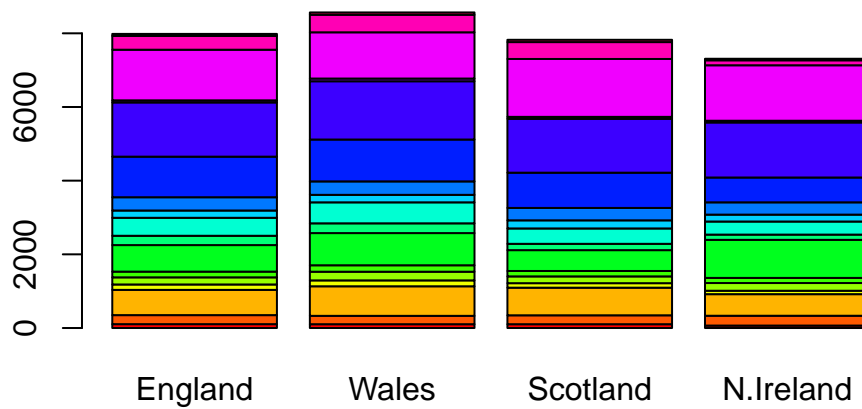
```
[1] 17 4
```

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



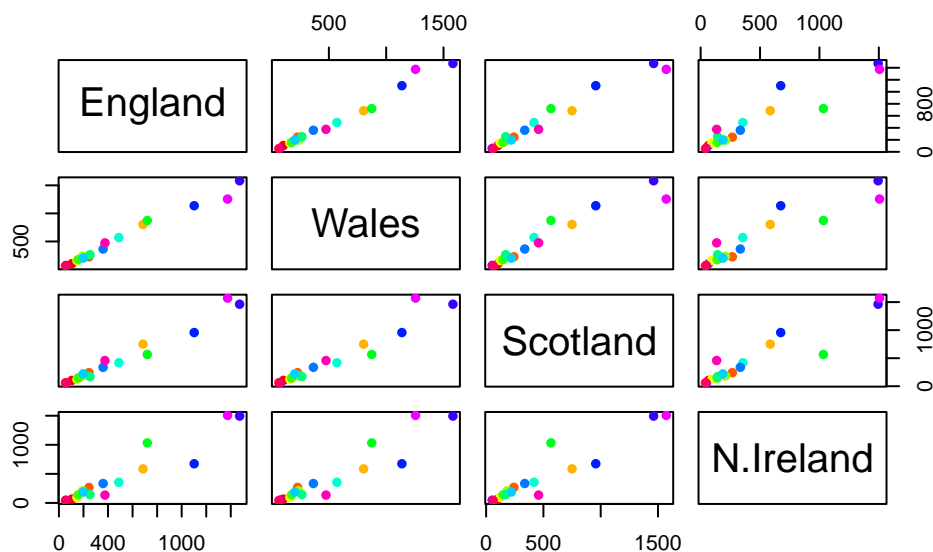
Q3. Changing what optional argument in the above `barplot()` function results in the following plot?

```
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(nrow(x)), pch=16)
```



If the point is in a straight line, then similar amount of similar amounts of that food consumed in both countries.

Looking at these types of “pairwise plots” can be helpful but it does not scale well and kind of sucks! There must be a better way...

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

In the upper right graph, there is a dark blue dot that is weighing more in England than in N. Ireland, there is a lighter teal colored dot that is weighing more in N. Ireland than in England. Overall, N.Ireland has more differences when compared to the other countries.

## PCA to the rescue

The main function for PCA in base R is called ‘prcomp()’. This function wants the transpose of our input data - i.e. the important foods in as columns and the countries as rows.

```
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

Let's see what is in our PCA result object 'pca'

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

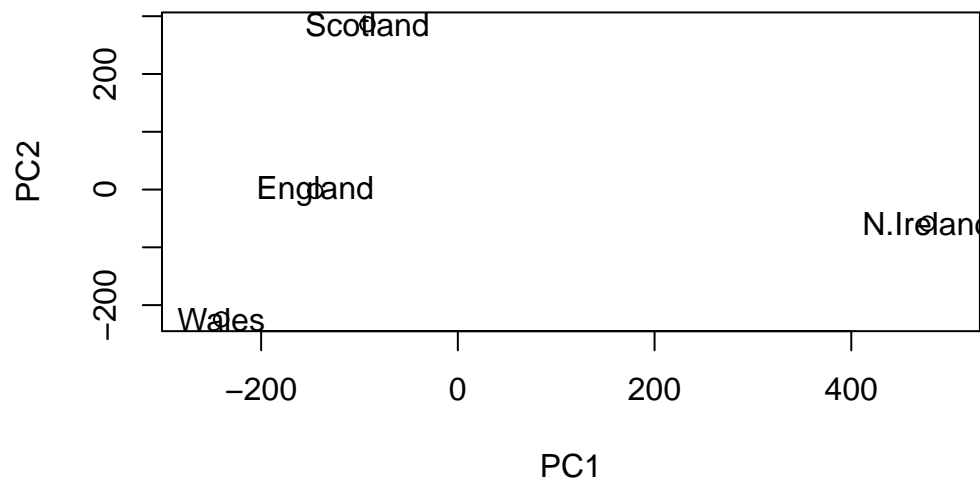
The 'pca\$x' result object is where we will focus first as this details how the countries are related to each other in terms of our new "axis" (a.k.a. "PCs", "eigenvectors", etc.)

```
head(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

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

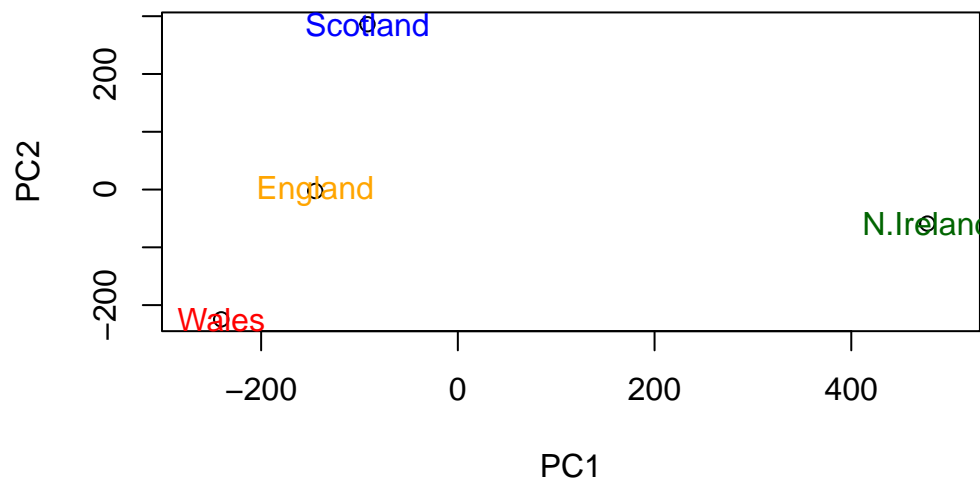
```
# Plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



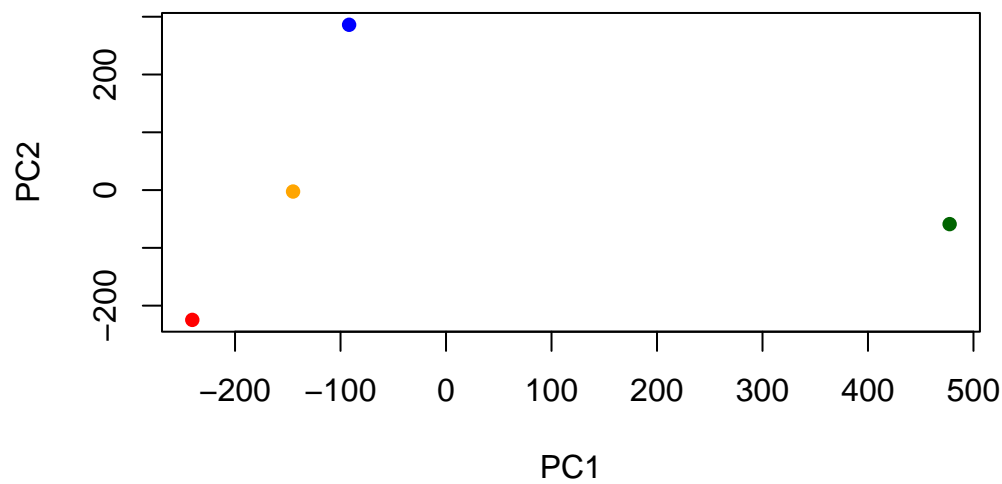
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x), col=c("orange", "red", "blue", "darkgreen"))
```





```
plot(pca$x[,1], pca$x[,2], pch=16, col=c("orange", "red", "blue", "darkgreen"), xlab="PC1", ylab="PC2")
```



We can look at the so-called PC “loadings” result object to see how the original food contribute

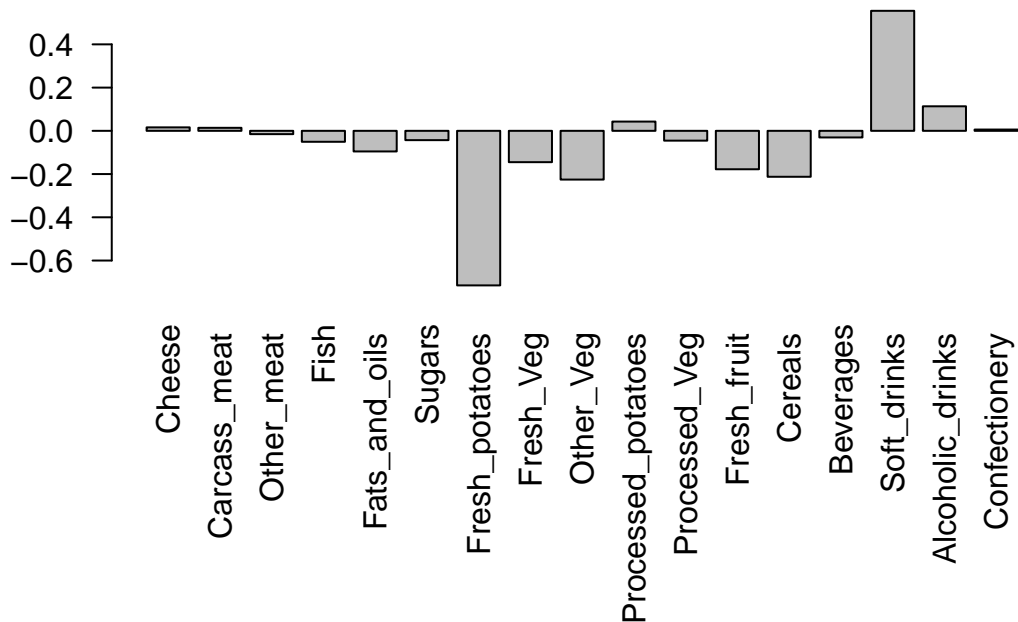
to our new PC )i.e. how the original variables contribute to our new better PC variables). Better because they capture more spread.

```
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			

Q9. Generate a similar 'loadings plot' for PC2. What two food groups feature prominently and what does PC2 mainly tell us about?

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



This plot from PC2 tells us that food with the largest positive loading score is only soft drinks. The food that has the highest negative loading score is only fresh\_potatoes.