

# Class07

Linh Tran (PID: A16435846)

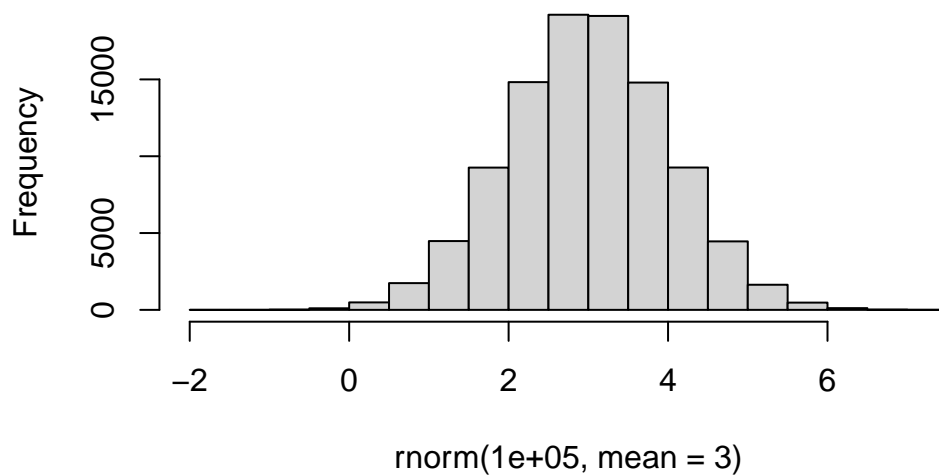
Today we will start our multi-part exploration of some key machine learning methods. We will begin with clustering - finding groupings in data, and then dimensionality reduction.

## Clustering

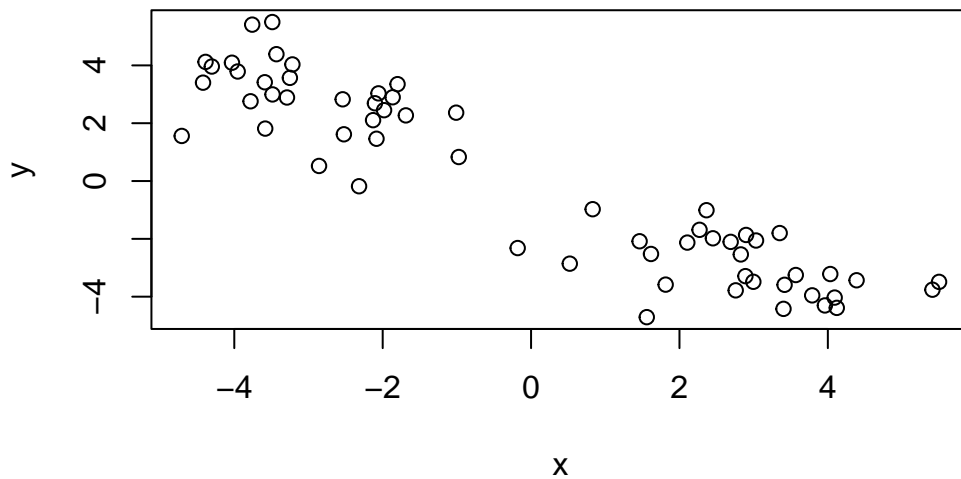
Let's start with "k-means" clustering. The main function in base R for this is `kmeans()`.

```
# Make up some data  
hist(rnorm(100000, mean=3))
```

**Histogram of `rnorm(1e+05, mean = 3)`**



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



Now let's try out `kmeans()`

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

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

Cluster means:

	x	y
1	2.864448	-2.953658
2	-2.953658	2.864448

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] 81.20248 81.20248
(between_SS / total_SS = 86.2 %)
```

Available components:

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

```
attributes(km)
```

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

```
$class
[1] "kmeans"
```

Q. How many points in cluster?

```
km$size
```

```
[1] 30 30
```

Q. What component of your result object details cluster assignment/membership?

```
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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Q. What are centers/mean values of each cluster?

```
km$centers
```

```
      x      y
1  2.864448 -2.953658
2 -2.953658  2.864448
```

Q. Make a plot of your data showing your clustering results (groupings/clusters and cluster centers)

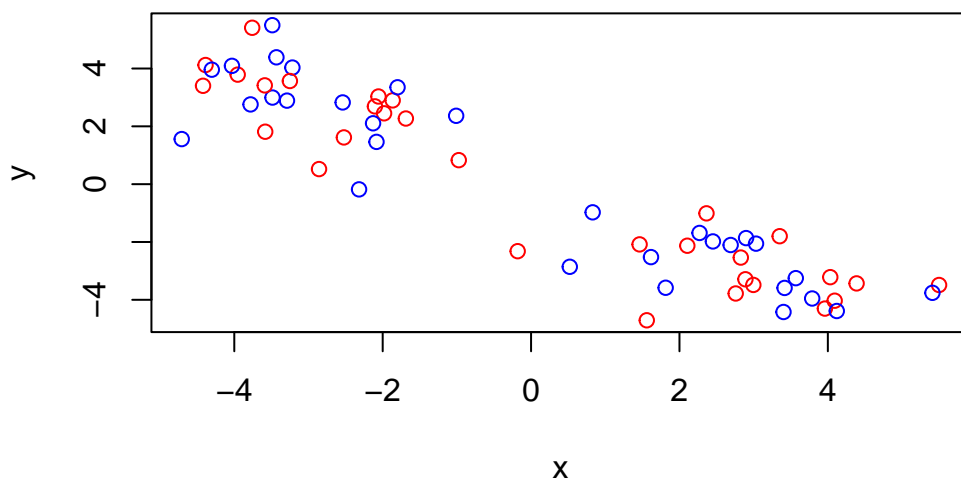
Recall

```
c(1:5) + c(1,100)
```

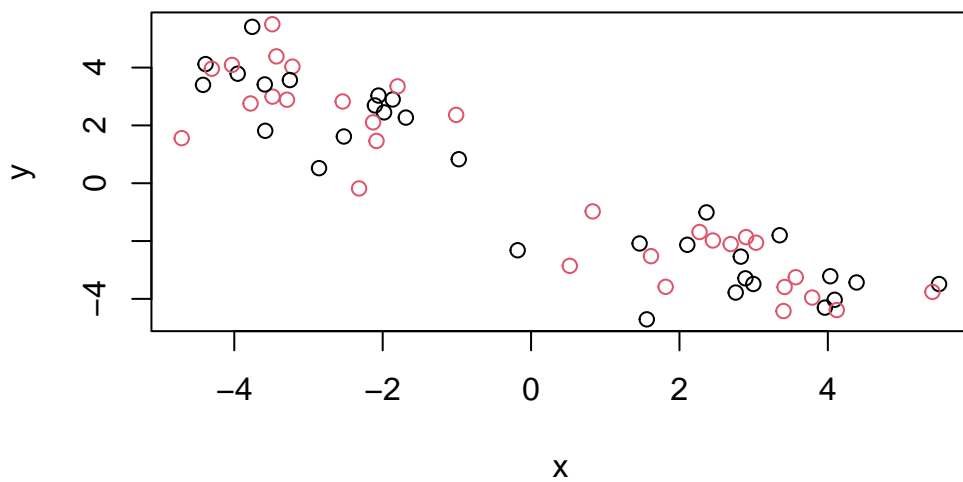
Warning in `c(1:5) + c(1, 100)`: longer object length is not a multiple of shorter object length

```
[1]  2 102  4 104  6
```

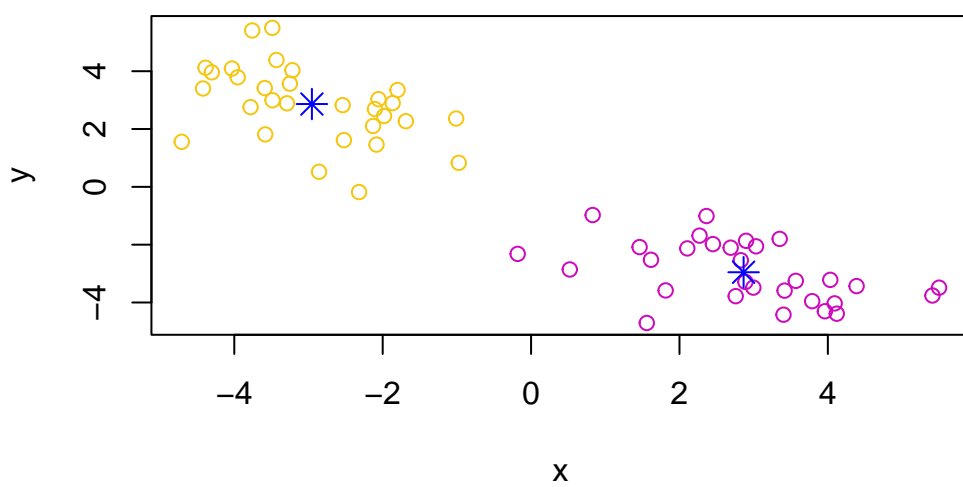
```
plot(x, col=c("red", "blue"))
```



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

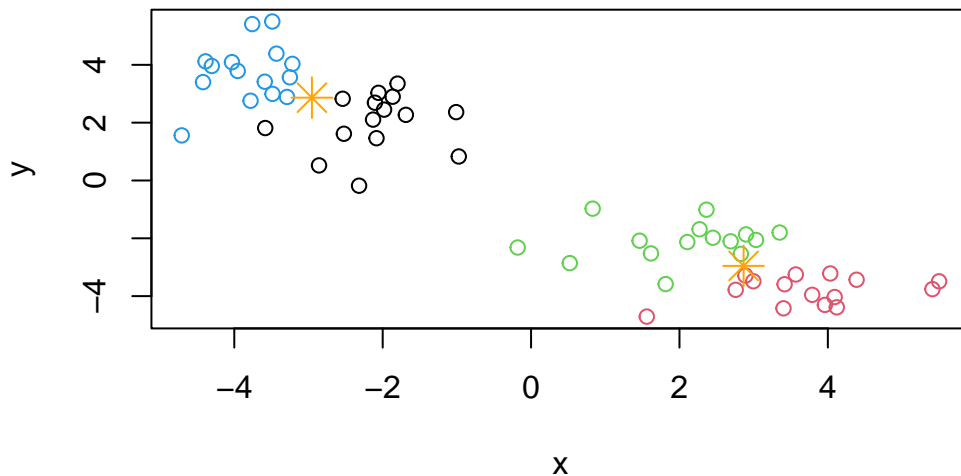


```
plot(x, col=km$cluster+5)
points(km$centers, col="blue", pch=8, cex=1.5)
```



Q. Run `kmeans()` again and cluster in 4 groups and plot the results.

```
km4 <- kmeans(x, centers=4)
plot(x, col=km4$cluster)
points(km4$centers, col="orange", pch=8, cex=2)
```



## Hierarchical Clustering

This form of clustering aims to reveal the structure in your data by progressively grouping points into an ever smaller number of clusters.

The main function in base R for this is called `hclust()`. This function does not take our input data directly but wants a “distance matrix” that details how (dis)similar all our input points are to each other.

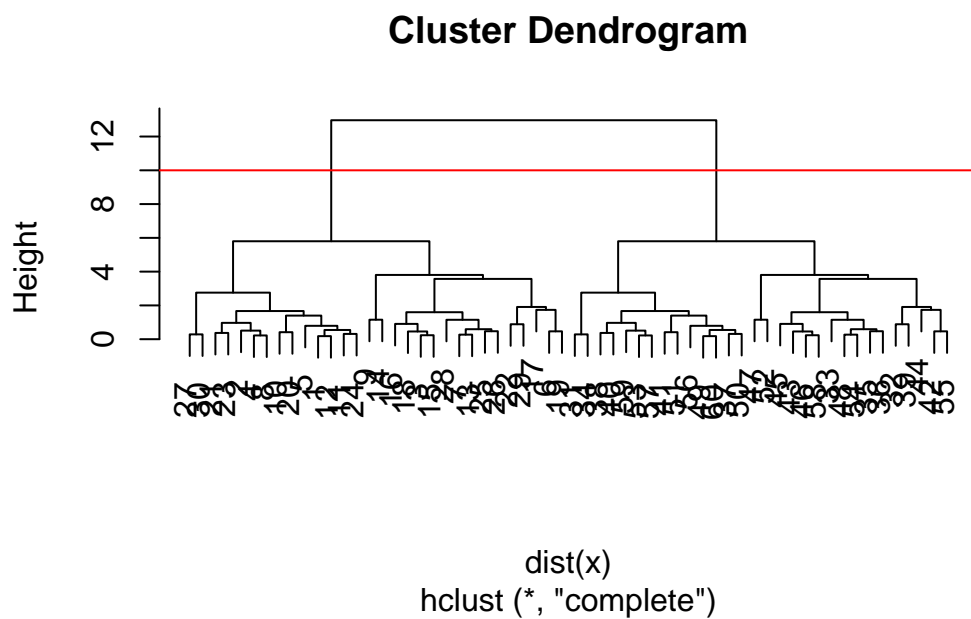
```
hc <- hclust(dist(x))
hc
```

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

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

The print out above is not very useful (unclick that from kmeans) but there is a useful `plot()` method.

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

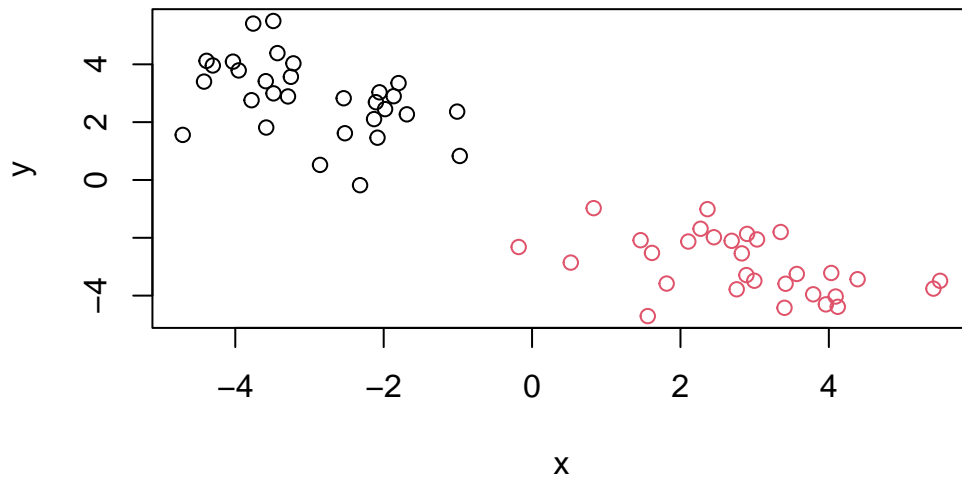


To get my main result (my cluster membership vector) I need to “cut” my tree using the function `cutree()`

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

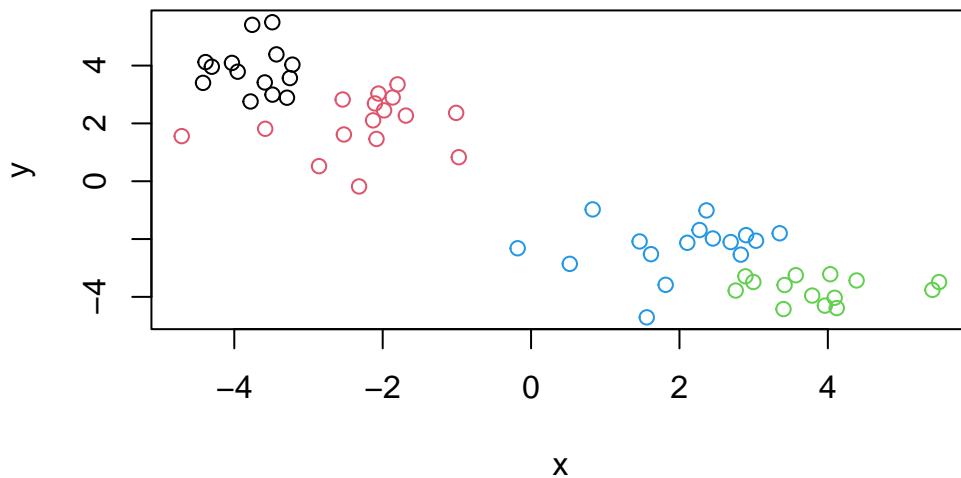
```
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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



```
plot(x,col= cutree(hc, h=4))
```





## Principal Component Analysis

The goal of PCA is to reduce the dimensionality of a dataset down to some smaller subset of new variables (called PCs) that are a useful bases for further analysis, like visualization, clustering, etc.

Read data about crazy eating

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

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
x
```

	X	England	Wales	Scotland	N.Ireland
1	Cheese	105	103	103	66
2	Carcass_meat	245	227	242	267
3	Other_meat	685	803	750	586
4	Fish	147	160	122	93
5	Fats_and_oils	193	235	184	209
6	Sugars	156	175	147	139

7	Fresh_potatoes	720	874	566	1033
8	Fresh_Veg	253	265	171	143
9	Other_Veg	488	570	418	355
10	Processed_potatoes	198	203	220	187
11	Processed_Veg	360	365	337	334
12	Fresh_fruit	1102	1137	957	674
13	Cereals	1472	1582	1462	1494
14	Beverages	57	73	53	47
15	Soft_drinks	1374	1256	1572	1506
16	Alcoholic_drinks	375	475	458	135
17	Confectionery	54	64	62	41

```
dim(x)
```

```
[1] 17  5
```

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?

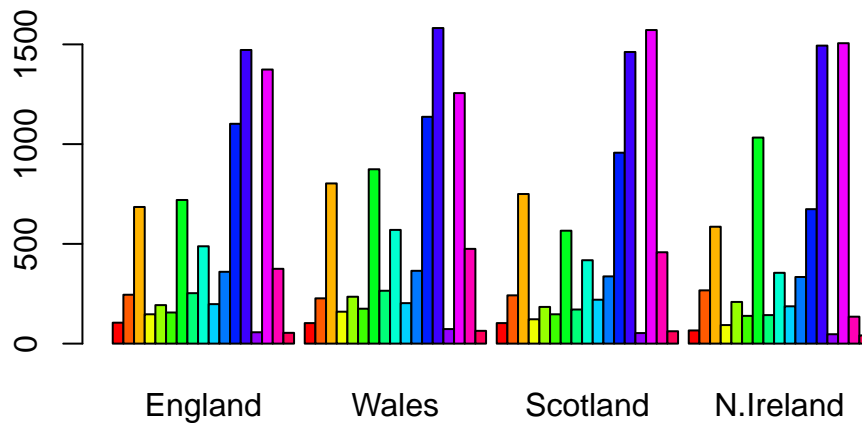
The ‘row-names problem’ can be solved by adding `row.names=1` to the code.

```
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

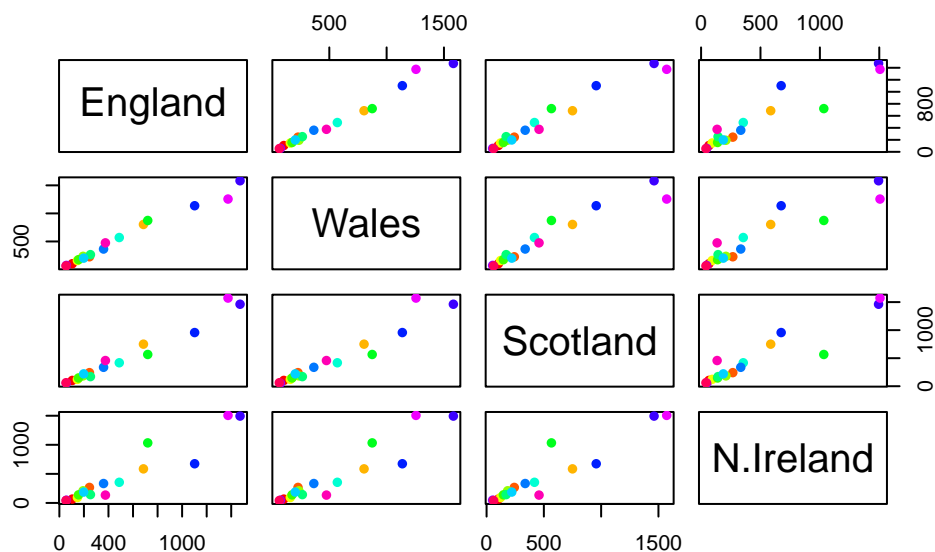
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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?

The so-called “pairs” plot can be useful for small datasets:

```
#rainbow(2000)
pairs(x, col=rainbow(nrow(x)), pch=16)
```



So the paris plot is useful for small datasets but it can be lots of work to interpret and gets untractable for larger datasets.

So PCA to the rescue... The mean function to do PCA in base R is call `prcomp()`

`t(x)`

	Cheese	Carcass_meat	Other_meat	Fish	Fats_and_oils	Sugars
England	105	245	685	147	193	156
Wales	103	227	803	160	235	175
Scotland	103	242	750	122	184	147
N.Ireland	66	267	586	93	209	139
	Fresh_potatoes	Fresh_Veg	Other_Veg	Processed_potatoes		
England	720	253	488		198	
Wales	874	265	570		203	
Scotland	566	171	418		220	
N.Ireland	1033	143	355		187	
	Processed_Veg	Fresh_fruit	Cereals	Beverages	Soft_drinks	
England	360	1102	1472	57	1374	
Wales	365	1137	1582	73	1256	
Scotland	337	957	1462	53	1572	
N.Ireland	334	674	1494	47	1506	

	Alcoholic_drinks	Confectionery
England	375	54
Wales	475	64
Scotland	458	62
N.Ireland	135	41

```
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

```
attributes(pca)
```

\$names

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

\$class

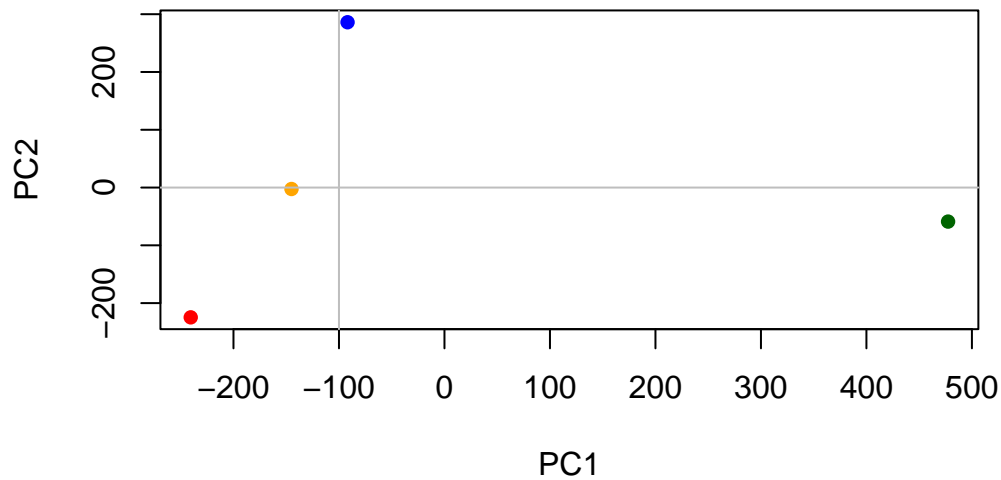
```
[1] "prcomp"
```

```
pca$x
```

	PC1	PC2	PC3	PC4
England	-144.99315	-2.532999	105.768945	-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

A major PCA result viz is called a “PCA plot” (a.k.a. a score plot, biplot, PC1 vs PC2 plot, ordination plot)

```
mycols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, xlab="PC1", ylab="PC2")
abline(h=0, col="gray")
abline(v=-100, col="gray")
```



Another important output from PCA is called the “loadings” vector or the “rotation” component - this tells us how much the original variables (the foods in this case) contribute to the new PCs

```
pca$rotation
```

	PC1	PC2	PC3	PC4
Cheese	-0.056955380	0.016012850	0.02394295	-0.694538519
Carcass_meat	0.047927628	0.013915823	0.06367111	0.489884628
Other_meat	-0.258916658	-0.015331138	-0.55384854	0.279023718
Fish	-0.084414983	-0.050754947	0.03906481	-0.008483145
Fats_and_oils	-0.005193623	-0.095388656	-0.12522257	0.076097502
Sugars	-0.037620983	-0.043021699	-0.03605745	0.034101334
Fresh_potatoes	0.401402060	-0.715017078	-0.20668248	-0.090972715
Fresh_Veg	-0.151849942	-0.144900268	0.21382237	-0.039901917
Other_Veg	-0.243593729	-0.225450923	-0.05332841	0.016719075
Processed_potatoes	-0.026886233	0.042850761	-0.07364902	0.030125166
Processed_Veg	-0.036488269	-0.045451802	0.05289191	-0.013969507
Fresh_fruit	-0.632640898	-0.177740743	0.40012865	0.184072217
Cereals	-0.047702858	-0.212599678	-0.35884921	0.191926714
Beverages	-0.026187756	-0.030560542	-0.04135860	0.004831876
Soft_drinks	0.232244140	0.555124311	-0.16942648	0.103508492

Alcoholic_drinks	-0.463968168	0.113536523	-0.49858320	-0.316290619
Confectionery	-0.029650201	0.005949921	-0.05232164	0.001847469

PCA looks to be a super useful method for gaining some insight into high dimensional data that is difficult to examine in other ways.

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

Changing the “`beside=`” option would result in the following plot

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

