Class07

Qihao Liu (PID:U08901197)

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

K-means clustering

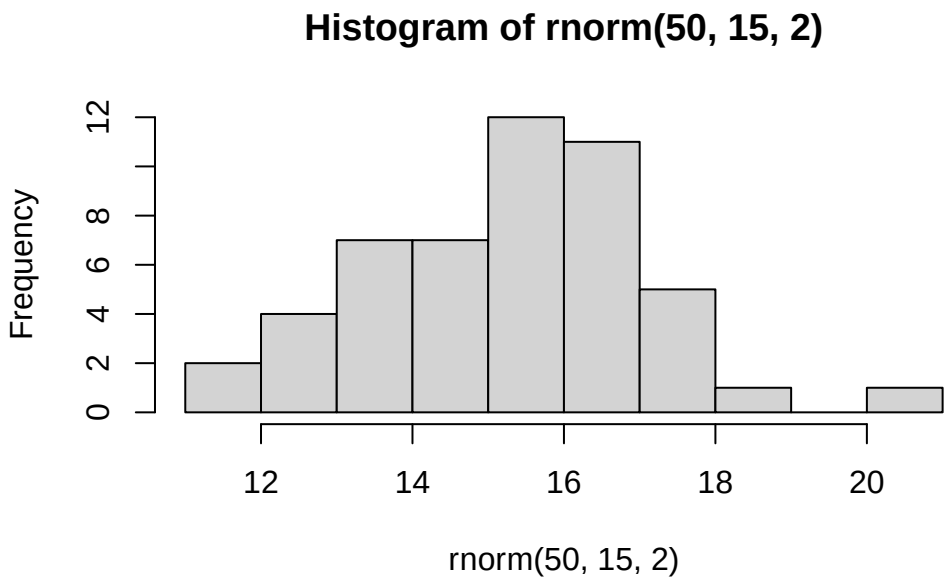
To see how this works, Let's first make up some data to cluster where we konw what the answer should be, we can use the `rnorm()` function to help here.

- this gives us n number of data drawn from a normal distribution with pre-determined mean and SD.

```
rnorm(50)
```

```
 [1] -0.314557740  0.293584366  0.437382985 -0.341177962  1.617358164
 [6]  0.285649551 -0.353732379 -1.707599965 -0.220765736  0.236525347
[11]  0.302301906  0.047119232  1.175289669  0.619331994  1.433629603
[16]  2.405872033  0.146716766 -0.770455906  0.540745219 -1.849265138
[21]  0.893409219  0.126023645  0.476127778 -0.005489706 -0.847520870
[26]  1.025420908  0.848141294 -0.339631106  1.086612360 -1.112626037
[31] -1.247014911 -0.567719273 -0.823729599  0.151221093 -0.551203148
[36]  1.176976313 -1.021067940 -1.617271045  0.036126101 -0.026246225
[41] -1.409964628 -0.847550774  1.224109650  0.201127474  0.695955435
[46]  0.425163377  0.724523271 -0.043011238  0.555065483  0.405408472
```

```
hist(rnorm(50,15,2))
```



Now let's make up a data in which there's clearly two distinct groups

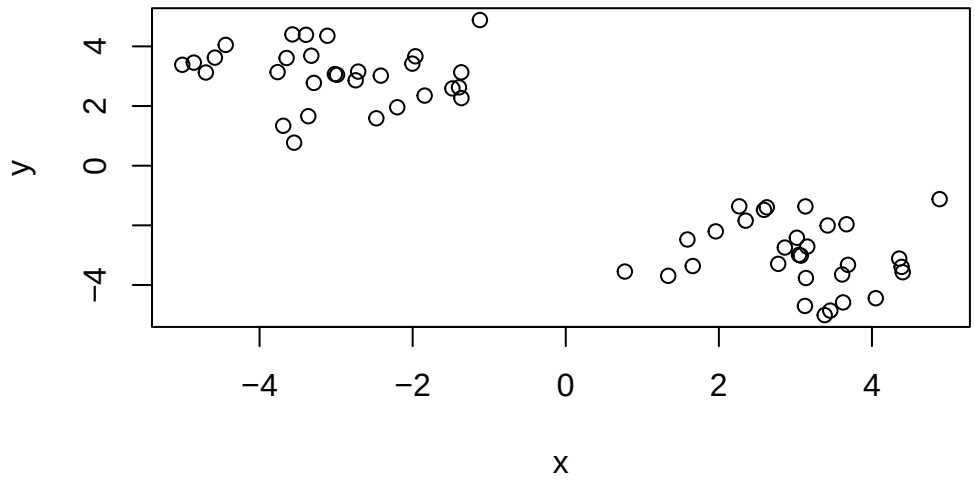
```
x <- c(rnorm(30, mean=-3), rnorm(30,mean=+3))
y <- rev(x) #reverse x

cbind(x,y) #column bind
```

```
                x          y
 [1,] -3.3633228  1.6587469
 [2,] -3.1139734  4.3562121
 [3,] -3.2904170  2.7755425
 [4,] -3.3237476  3.6866605
 [5,] -1.3626583  2.2660194
 [6,] -2.0039946  3.4205428
 [7,] -4.8589941  3.4553680
 [8,] -2.9881786  3.0470851
 [9,] -4.5835530  3.6224563
[10,] -2.4161763  3.0188452
[11,] -1.1215362  4.8836016
[12,] -1.9639578  3.6676564
[13,] -1.3651567  3.1307758
[14,] -3.3930503  4.3876657
[15,] -2.7443310  2.8617559
```

```
[16,] -3.5709327  4.4018028
[17,] -4.4421164  4.0495250
[18,] -1.4806899  2.5901605
[19,] -3.6912163  1.3383390
[20,] -2.2012189  1.9595087
[21,] -2.7116102  3.1547014
[22,] -3.5482427  0.7720112
[23,] -1.8432264  2.3500865
[24,] -3.0159769  3.0702528
[25,] -2.4743425  1.5892833
[26,] -3.7645592  3.1374938
[27,] -1.3952046  2.6258386
[28,] -3.6469766  3.6106285
[29,] -5.0095559  3.3824016
[30,] -4.7016793  3.1243123
[31,]  3.1243123 -4.7016793
[32,]  3.3824016 -5.0095559
[33,]  3.6106285 -3.6469766
[34,]  2.6258386 -1.3952046
[35,]  3.1374938 -3.7645592
[36,]  1.5892833 -2.4743425
[37,]  3.0702528 -3.0159769
[38,]  2.3500865 -1.8432264
[39,]  0.7720112 -3.5482427
[40,]  3.1547014 -2.7116102
[41,]  1.9595087 -2.2012189
[42,]  1.3383390 -3.6912163
[43,]  2.5901605 -1.4806899
[44,]  4.0495250 -4.4421164
[45,]  4.4018028 -3.5709327
[46,]  2.8617559 -2.7443310
[47,]  4.3876657 -3.3930503
[48,]  3.1307758 -1.3651567
[49,]  3.6676564 -1.9639578
[50,]  4.8836016 -1.1215362
[51,]  3.0188452 -2.4161763
[52,]  3.6224563 -4.5835530
[53,]  3.0470851 -2.9881786
[54,]  3.4553680 -4.8589941
[55,]  3.4205428 -2.0039946
[56,]  2.2660194 -1.3626583
[57,]  3.6866605 -3.3237476
[58,]  2.7755425 -3.2904170
```

```
[59,]  4.3562121 -3.1139734
[60,]  1.6587469 -3.3633228
```

```
z <- cbind(x,y)
plot(z)
```



- Q: What does cbind do?

The function for K-means clustering in "base" R is called `kmeans()` (k in kmeans means how many cluster you want)

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

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

Cluster means:
          x         y
1  3.046509 -2.979687
2 -2.979687  3.046509

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] 62.67531 62.67531
 (between_SS / total_SS =  89.7 %)

Available components:

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

To get the results of the returned list object, we can use the dollar $ syntax, for example, to get the size of each cluster:

- Note the components listed fro K-means are essentially different types of results

  Q: how many points are in each cluster

```
k$size
```

```
[1] 30 30
```

  Q: What 'component' gives - Membership/ Cluster Assignment - Cluster Center

```
k$cluster #Membership
```

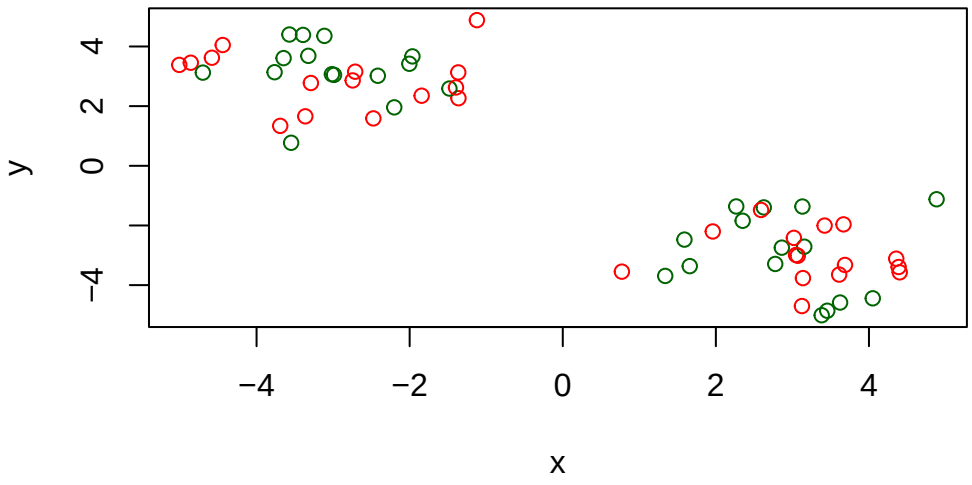
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
k$centers #Cluster Center
```

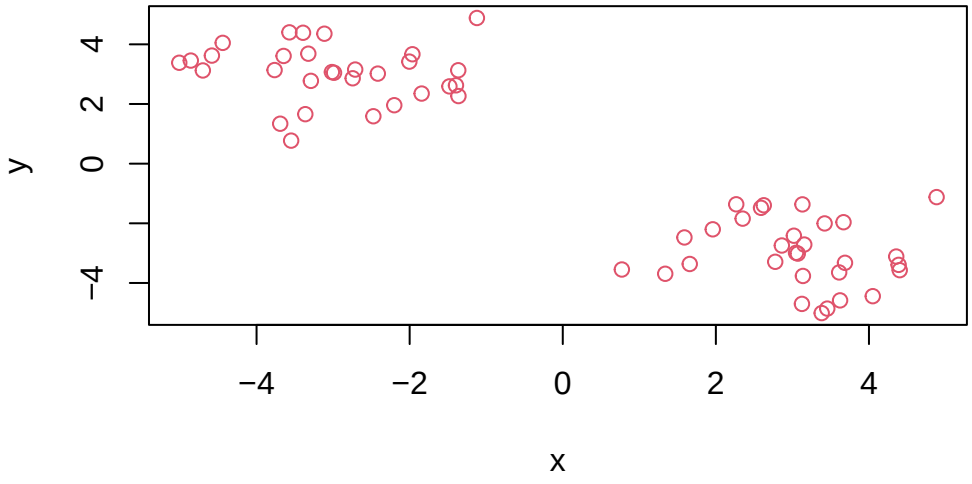
```
          x         y
1  3.046509 -2.979687
2 -2.979687  3.046509
```

  Q: Make a result figure of the data colored by cluster membership and show cluster centers

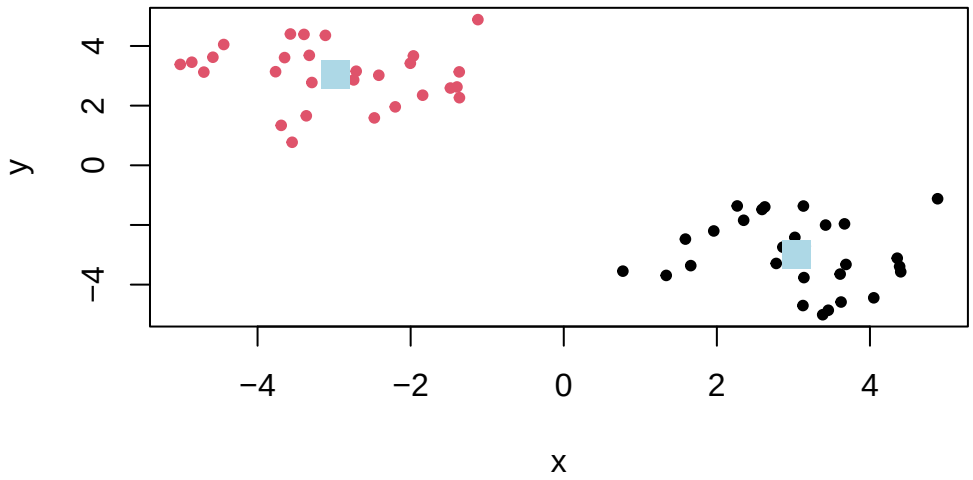
```r
plot(z, col = c("red","darkgreen")) #this colors the dots in alternating colors
```




```r
plot(z, col = 2) #color by bumber
```



```
plot(z, col = k$cluster, pch=20) # this basically tells R, use the number indicating each pt
points(k$centers, col = "lightblue", cex=2, pch=15)
```



K-means clustering is very popular as it is very fast and relatively straight forward: it takes numeric data as input and returns the cluster membership vector etc. But the "issue/feature" of it is that you need to tell `kmeans()` how many cluster we want!

> Q: Run Kmeans() again and cluster into 4 groups, and plot the results So the result is not ideal...

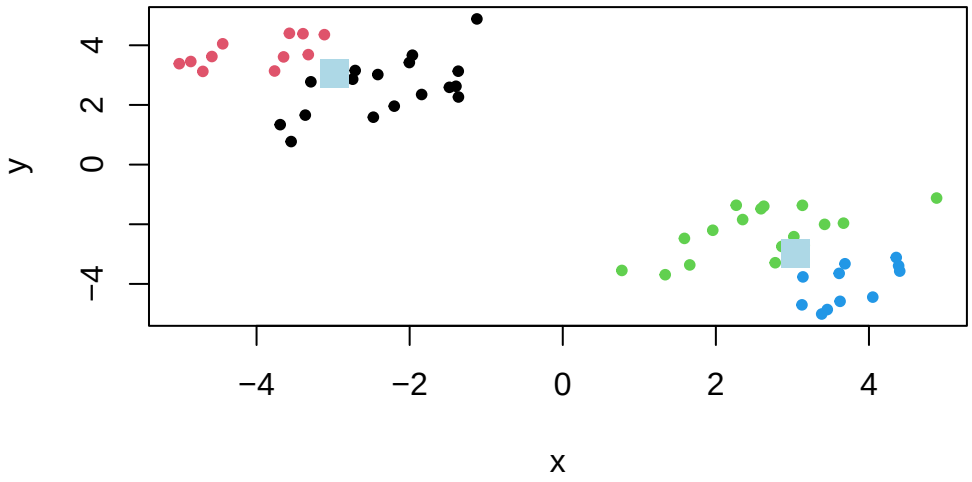
```
k4 <- kmeans(z, centers=4)
k4$cluster
```

```
 [1] 1 2 1 2 1 1 2 1 2 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 2 1 2 2 2 4 4 4 3 4 3 3 3
[39] 3 3 3 3 3 4 4 3 4 3 3 3 3 4 3 4 3 3 4 3 4 3
```

```
k4$centers
```

```
          x         y
1 -2.367445  2.641092
2 -4.037194  3.746775
3  2.641092 -2.367445
4  3.746775 -4.037194
```

```r
plot(z, col = k4$cluster, pch=20)
points(k$centers, col = "lightblue", cex=2, pch=15)
```



An alternative way is to make a **Scree Plot**, graphing total variation within a cluster (measured as Total Within-Cluster Sum of Squares)against the number of centers, and we can take the "elbow" center (the centers n that has the greatest decrease in total variation within a cluster compare to n-1) to be the optimal one.

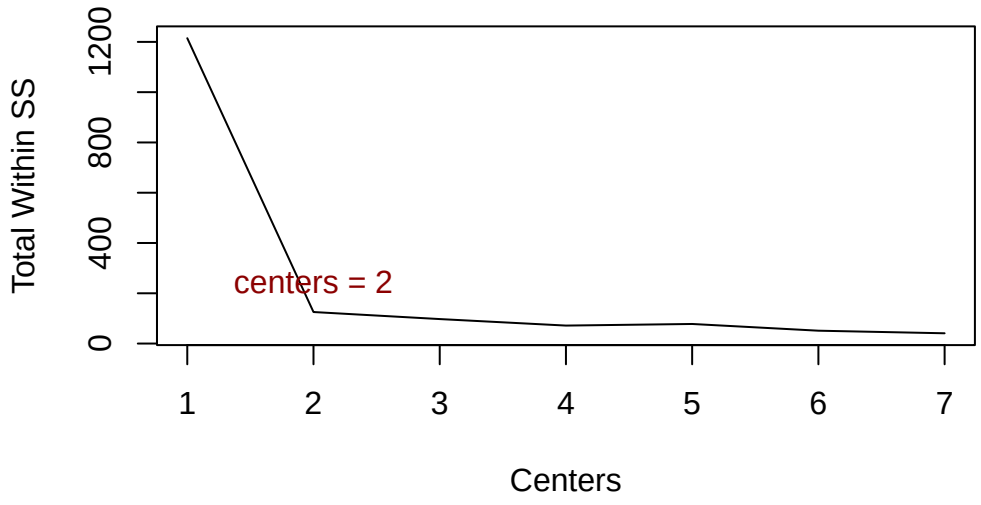
- Note scree is the place underneath a cliff that the rubbles collects

```r
k1 <- kmeans(z, centers=1)
k2 <- kmeans(z, centers=2)
k3 <- kmeans(z, centers=3)
k4 <- kmeans(z, centers=4)
k4$tot.withinss
```

```
[1] 71.52013
```

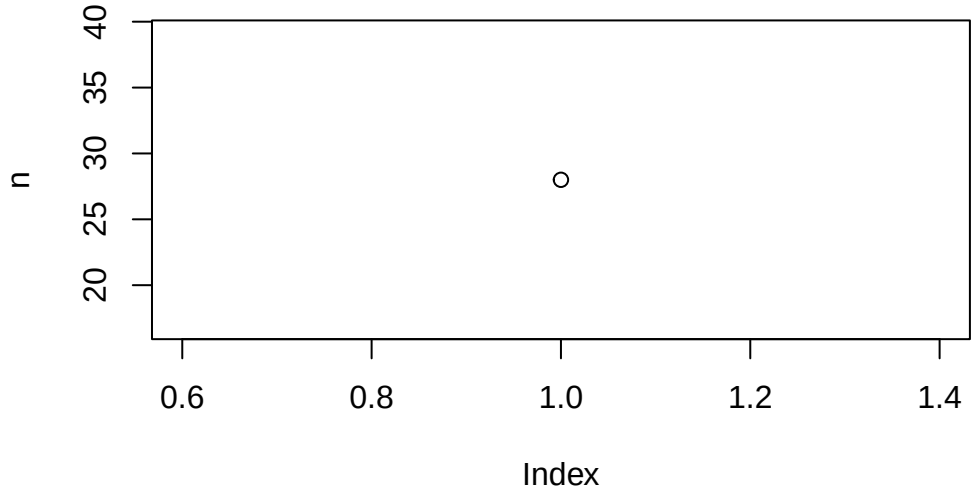
```r
k5 <- kmeans(z, centers=5)
k6 <- kmeans(z, centers=6)
k7 <- kmeans(z, centers=7)
x <- c(1,2,3,4,5,6,7)
```

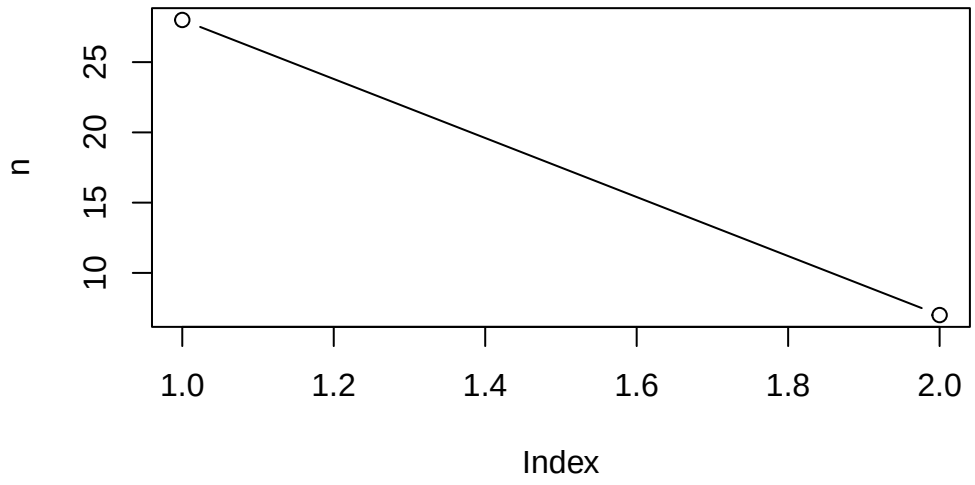
```r
y <- c(k1$tot.withinss, k2$tot.withinss, k3$tot.withinss, k4$tot.withinss, k5$tot.withinss, k
plot(x,y, type="l", xlab = "Centers", ylab = "Total Within SS")
text(x[2], y[2], labels = "centers = 2", pos = 3, col = "darkred")
```

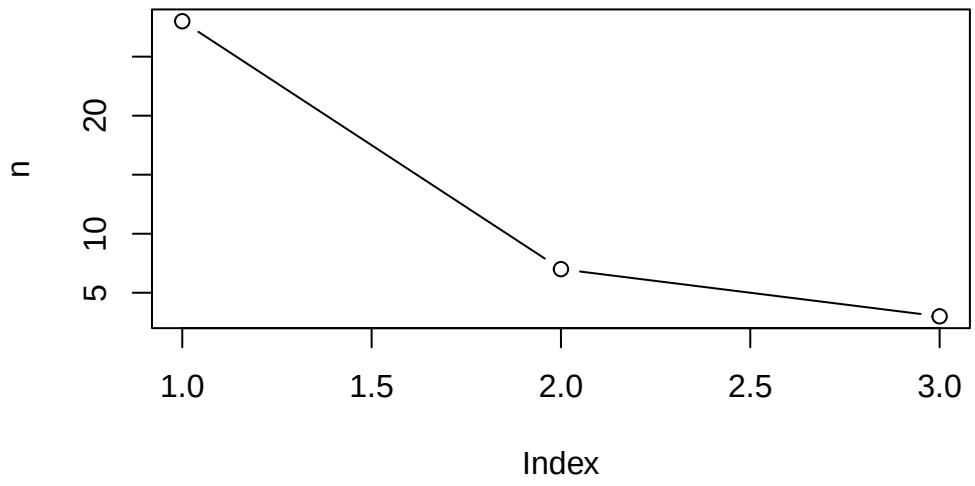


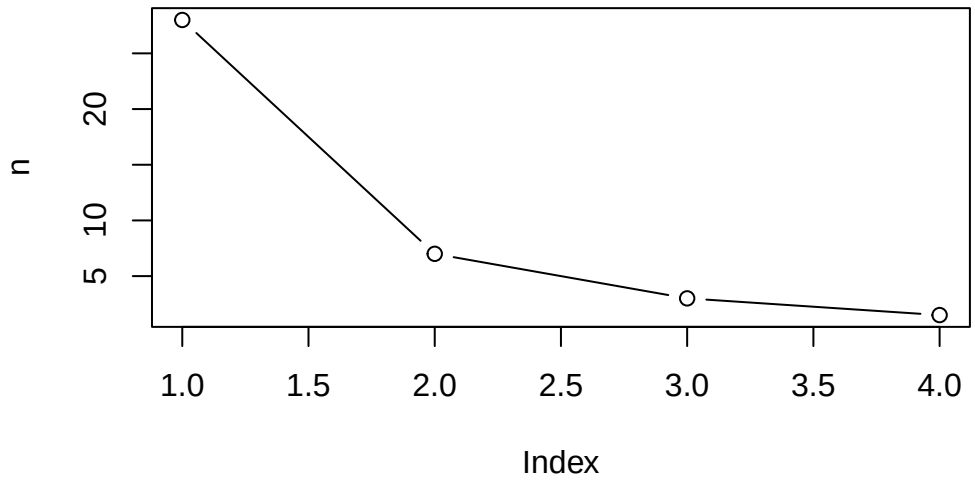
or we can use `for()` loop

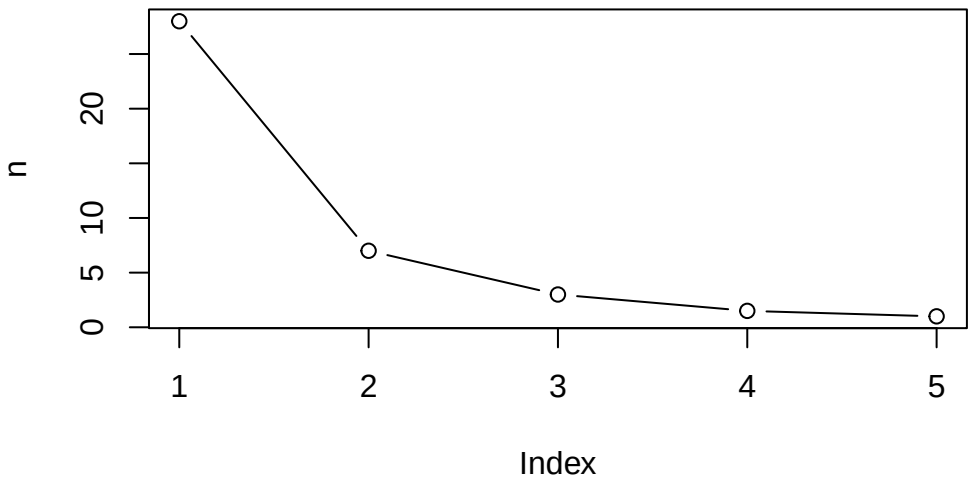
```r
n <- NULL
for(i in 1:5) {
  n <- c(n,kmeans(x, centers = i)$tot.withinss)
  plot(n, typ="b")
}
```


n
20
25
30
35
40
0.6
0.8
1.0
1.2
1.4
Index


n
10
15
20
25
1.0
1.2
1.4
1.6
1.8
2.0
Index


n
5
10
20
1.0
1.5
2.0
2.5
3.0
Index


n
5
10
20
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Index



Hierarchical Clustering

this meain "base" R function for Hierarchical Clustering is called `hclust()`

- the only required input of `hclust()` is d, a disimilarity matrix generated by the `dist()` function

```
hclust()
```

```
Error in hclust(): argument "d" is missing, with no default
```

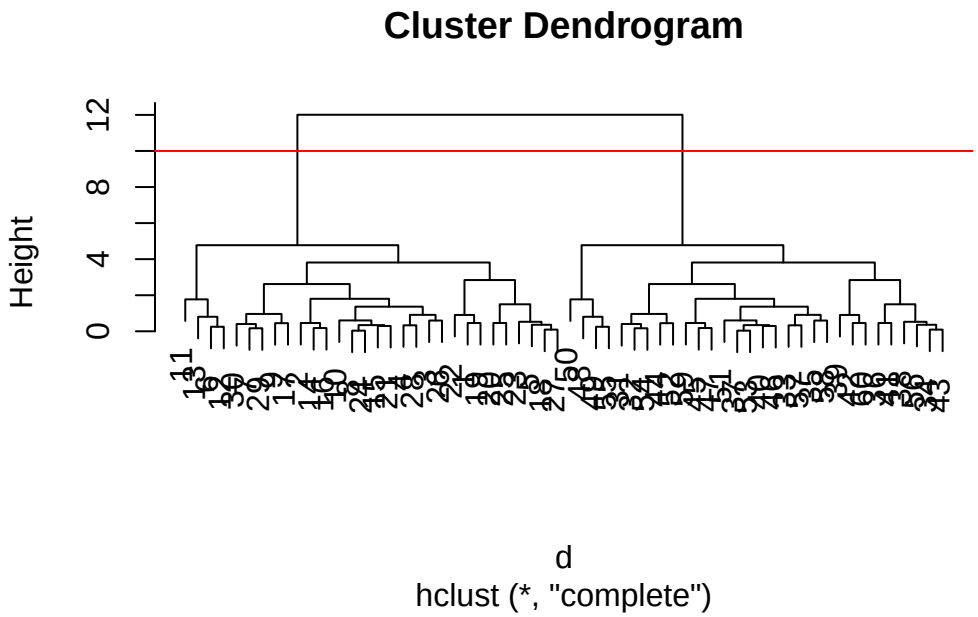
- `dist()` function calculate a distance matrix for our data, its a analysis of all paralle comparision of every point in data (it gives us a triangle because it is symmetrical)

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

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

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

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




```r
cutree(hc,h=10)
```

```
 [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
```

if we look at the dendrogram, we can clearly see that there are two large clusters, and higher numbers are clustered on one side of the plot, and lower numbers are clustered on the other side of the plot (this is related to how we build the data, we get 30 numbers with one mean, and 30 other numbers with a different mean)

Horizontal line contains the important information. We can see that if we cut the dendrogram at the dark red line with the `cutree ()` function, it would yield a membership output

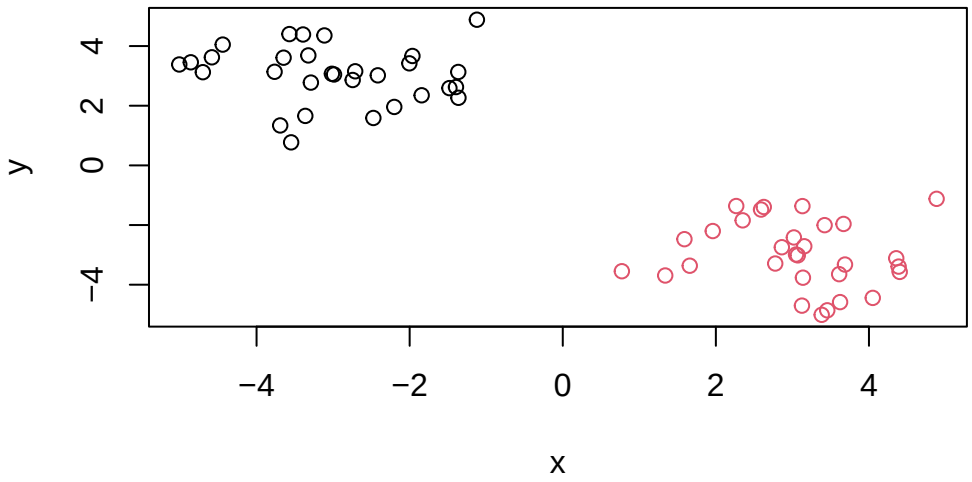
To get our cluster membership output (i.e. our main clustering results), we can cut the tree at a given height or at a height that yields a diven “k”

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

Q: Plot the data with our hclut results coloring

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



Dimensionality Reduction: Principle Component Analysis(PCA)

The major goal of PCA is dimensionality reduction and lost as little information/essence of data as possible

2D-Example: - The first principle component (PC) follows a best fit through the data points. - The second principle component picks up the wiggling of the data along the first PC - Combined, these captures most of the spread of the data and can act as new axis - The data should have max variation on PC1, PC2 captures the rest of the variants (Generally, for PC1,2,3, the variation captured decreases as the numbering goes up)

Principal Component Analysis (PCA)

PCA of UK Food Data

Import food data from an online CSV file.

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

> 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 for the first column, the name is X, and the row names are not food but numbers To fix this: We can try....

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

This is no good because the code is overwriting itself, so each time we run it, we lose some data.

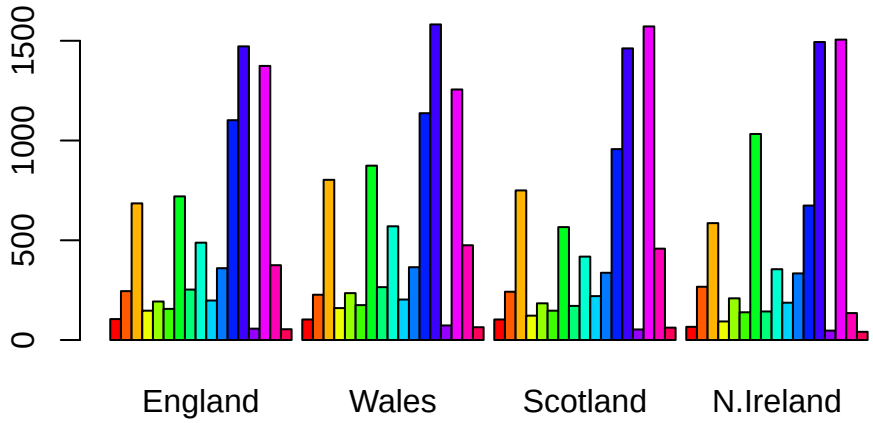
Instead, we do

```
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

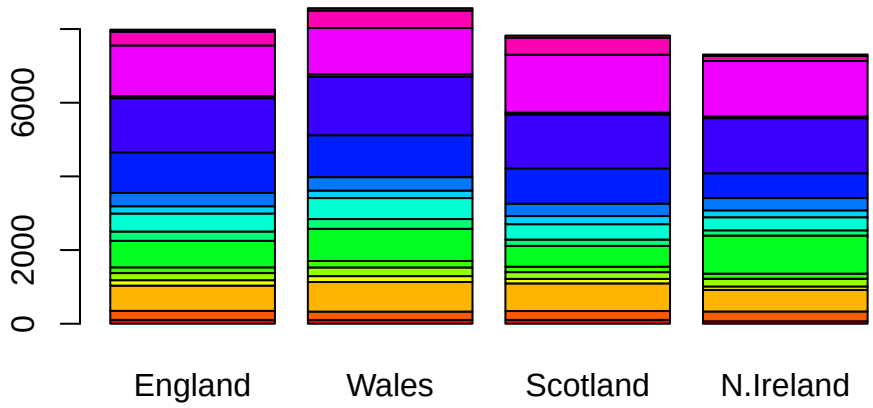
Now we can try some base figures:

```
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)))
```

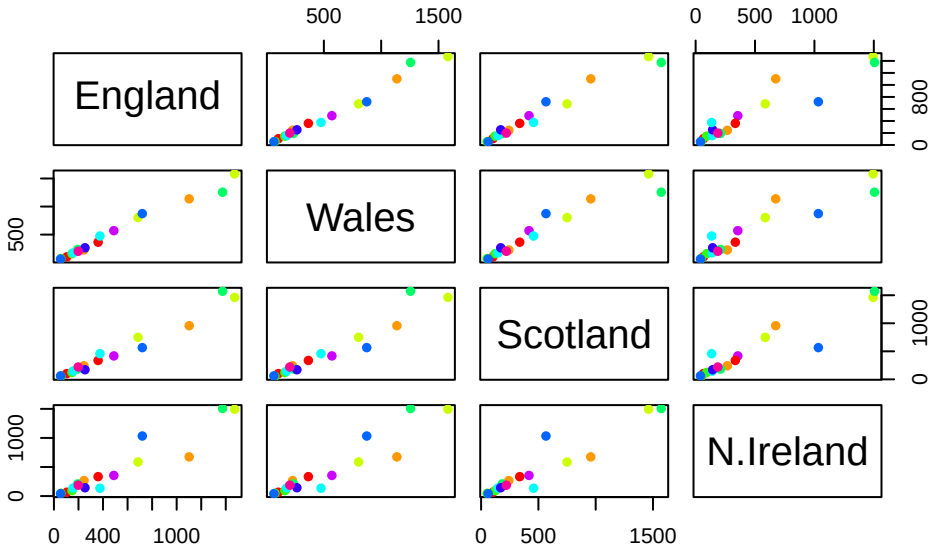


These plot are usless

> Q5: We can use the pairs() function to generate all pairwise plots for our countries. 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?

There is one plot that can be useful FOR SAMLL DATASET

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



To read this plot, if we take the first row where it starts with "England", we can see England vs Wales, England vs Scotland, England vs N. Ireland

- If the country eats exactly the same, it will be an diagnal line
- deviation from the diagnal highligths the differences in eating
- from this we can see that N. Ireland is very different >Main point: it can be difficult to spot major trends and patterns even in relatively small multivariate dataset (here we only have 17 dimension, typically it could be 1000s)

PCA to the rescue

Now let's try PCA on this data to see if it is actually useful. The main functionin "base" R for PCA is called `prcomp()`

- First we transpose the data so food is in the columns. This allow us to do PCA on the food with `t()` transpose function.

```
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))
PCA
```

```
Standard deviations (1, .., p=4):
[1] 3.241502e+02 2.127478e+02 7.387622e+01 3.175833e-14

Rotation (n x k) = (17 x 4):
                            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
```

```
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
```

Note this summary tells us that PC1 captures 67.44% of the variants, PC2 29.05%, and PC3 3.5%. Together, PC1 and 2 captures >95% of the variance

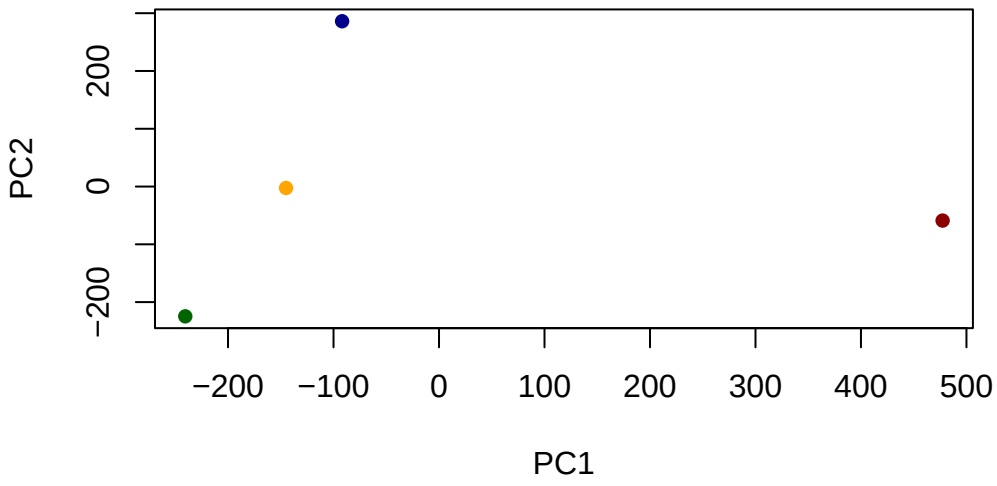
Now let's try to plot the results in "base" R and ggplot

- First figure out what we are plotting, we are plotting X, which contains the PCs, and since PC1 and PC2 capture most of the variation, we are only gonna plot them two

```
PCA$x
```

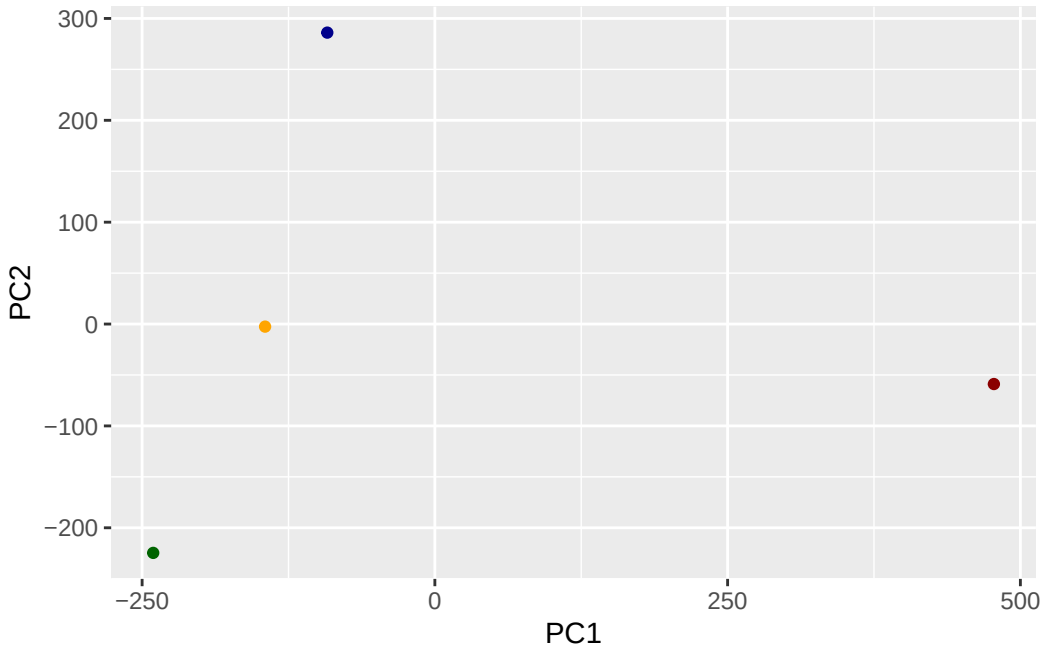
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
cols <- c("orange","darkgreen","darkblue","darkred")
plot(PCA$x[,1], PCA$x[,2], col=cols,pch=16, xlab = "PC1", ylab = "PC2")
```



If we are gonna use ggplot >Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
library(ggplot2)
ggplot(PCA$x)+
  aes(PC1,PC2)+
  geom_point(col=cols)
```



From this graph, we can clearly see that N. Ireland is distinct from the rest of the countrys from the survey.

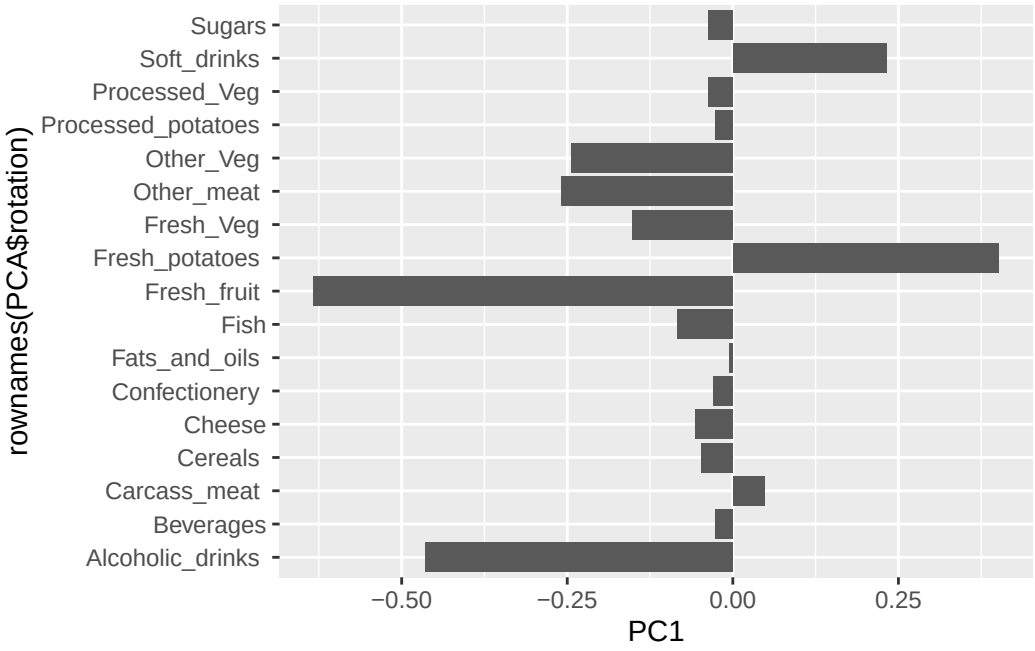
But what food contributes to these PCs? We can figure this out, with `PCA$rotation`, we can see the contribution of each food to the different PCs, now if we plot it we can visualize what contributes to the PCs

```
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
```

```
ggplot(PCA$rotation)+
  aes(PC1,rownames(PCA$rotation))+
  geom_col()
```



This graph tells us that Ireland consume more soft drink and fresh potatoes

PCA looks super useful, and we will come back to describe this further next class