

Class 7: Machine Learning 1

Hailey Heirigs (PID: A16962278)

Table of contents

Clustering	1
K-means	4
Heirarchical Clustering	8
Principal Component Analysis (PCA)	11
Data Import	11
PCA to the rescue	13

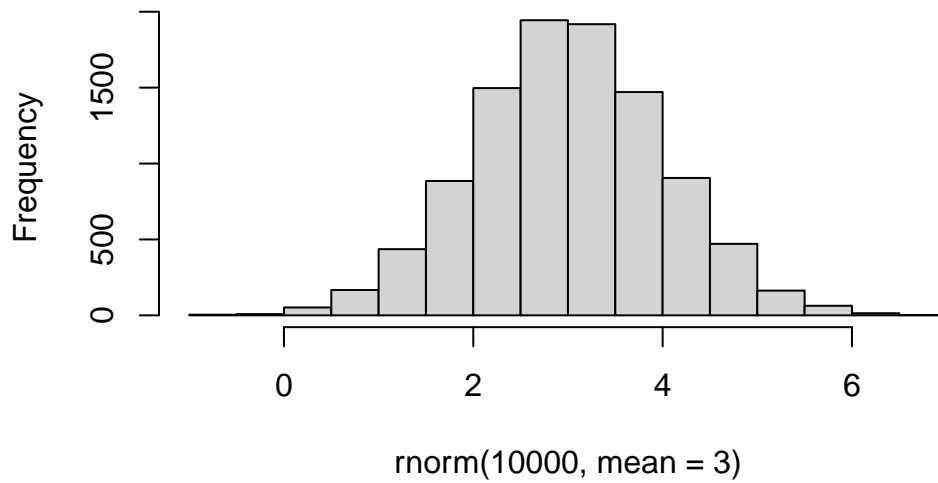
Today we will explore unsupervised machine learning methods starting with clustering and dimensionality reduction.

Clustering

To start let's make up some data to cluster where we know what the answer should be. The `rnorm()` function will help us here.

```
hist( rnorm(10000, mean=3) )
```

Histogram of rnorm(10000, mean = 3)



Return 30 numbers centered on -3 and +3

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

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

x
```

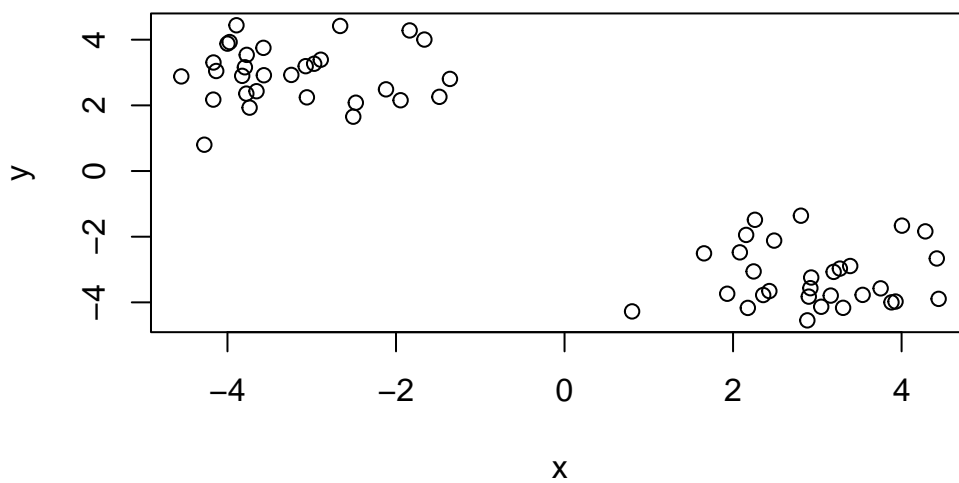
	x	y
[1,]	-4.2748713	0.8016329
[2,]	-4.1336867	3.0462346
[3,]	-2.5073503	1.6552029
[4,]	-3.8933191	4.4395569
[5,]	-1.4851408	2.2595452
[6,]	-3.9999514	3.8798118
[7,]	-3.5734989	3.7516583
[8,]	-3.0576779	2.2439553
[9,]	-3.0717977	3.1935085
[10,]	-3.7763379	2.3575296
[11,]	-1.8380964	4.2816150
[12,]	-4.1683991	2.1753669
[13,]	-4.5481888	2.8814663

[14,]	-3.2425033	2.9273634
[15,]	-3.8252983	2.8986505
[16,]	-3.9746794	3.9280050
[17,]	-4.1663664	3.3058628
[18,]	-1.3599124	2.8052710
[19,]	-3.6554448	2.4305409
[20,]	-3.7372701	1.9306727
[21,]	-2.8927380	3.3900746
[22,]	-3.5680137	2.9172564
[23,]	-1.6631848	4.0042973
[24,]	-3.7933500	3.1592601
[25,]	-2.9712445	3.2677511
[26,]	-1.9454581	2.1552745
[27,]	-2.1185544	2.4874278
[28,]	-3.7714854	3.5376503
[29,]	-2.6625572	4.4177890
[30,]	-2.4770201	2.0805820
[31,]	2.0805820	-2.4770201
[32,]	4.4177890	-2.6625572
[33,]	3.5376503	-3.7714854
[34,]	2.4874278	-2.1185544
[35,]	2.1552745	-1.9454581
[36,]	3.2677511	-2.9712445
[37,]	3.1592601	-3.7933500
[38,]	4.0042973	-1.6631848
[39,]	2.9172564	-3.5680137
[40,]	3.3900746	-2.8927380
[41,]	1.9306727	-3.7372701
[42,]	2.4305409	-3.6554448
[43,]	2.8052710	-1.3599124
[44,]	3.3058628	-4.1663664
[45,]	3.9280050	-3.9746794
[46,]	2.8986505	-3.8252983
[47,]	2.9273634	-3.2425033
[48,]	2.8814663	-4.5481888
[49,]	2.1753669	-4.1683991
[50,]	4.2816150	-1.8380964
[51,]	2.3575296	-3.7763379
[52,]	3.1935085	-3.0717977
[53,]	2.2439553	-3.0576779
[54,]	3.7516583	-3.5734989
[55,]	3.8798118	-3.9999514
[56,]	2.2595452	-1.4851408

```
[57,] 4.4395569 -3.8933191
[58,] 1.6552029 -2.5073503
[59,] 3.0462346 -4.1336867
[60,] 0.8016329 -4.2748713
```

Make a plot of x

```
plot(x)
```



K-means

The main function in “base” R for K-means clustering is called `kmeans()`:

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

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

Cluster means:

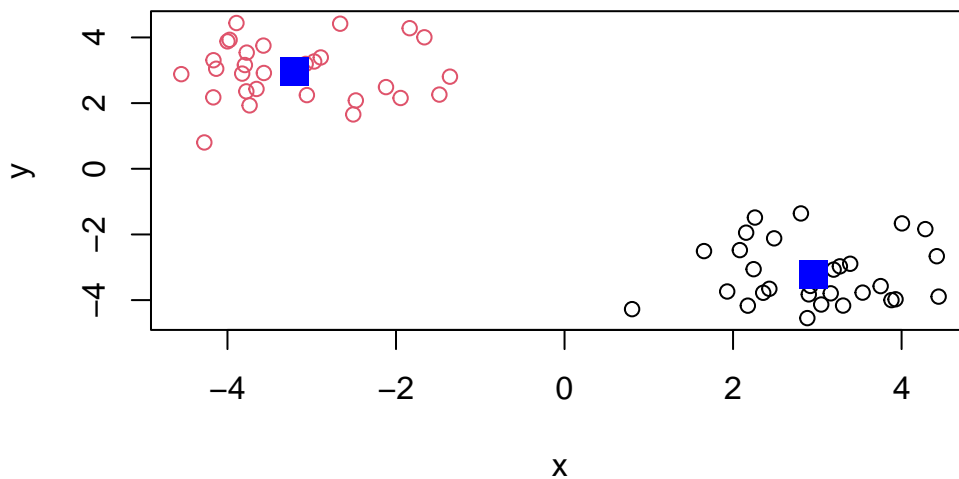
	x	y
1	2.953694	-3.205113


```
km$centers
```

```
      x      y
1 2.953694 -3.205113
2 -3.205113  2.953694
```

Q. Make a plot of our `kmeans()` results showing cluster assignment using different colors for each cluster/group of points and cluster centers in blue.

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



Q. Run `kmeans()` again on `x` and this time cluster into 4 groups/clusters and plot the same result figure as above.

```
km4 <- kmeans(x, centers = 4)
km4
```

K-means clustering with 4 clusters of sizes 8, 16, 6, 30

Cluster means:

	x	y
1	-3.456796	1.959435
2	-3.630542	3.433869
3	-1.735058	2.998905
4	2.953694	-3.205113

Clustering vector:

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

Within cluster sum of squares by cluster:

```
[1] 5.373129 8.155230 4.623940 46.131022
(between_SS / total_SS = 94.8 %)
```

Available components:

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

```
attributes(km4)
```

\$names

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

\$class

```
[1] "kmeans"
```

```
km4$size
```

```
[1] 8 16 6 30
```

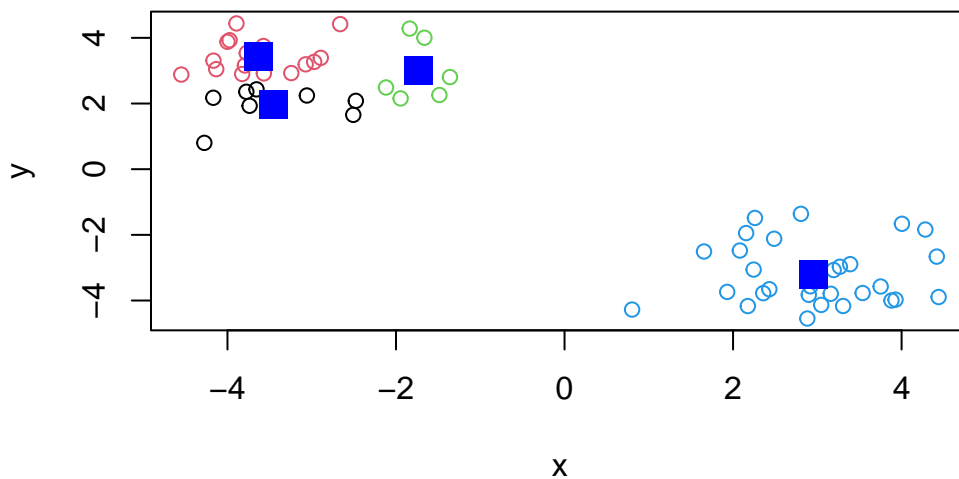
```
km4$cluster
```

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

```
km4$centers
```

	x	y
1	-3.456796	1.959435
2	-3.630542	3.433869
3	-1.735058	2.998905
4	2.953694	-3.205113

```
plot(x, col=km4$cluster )
points(km4$centers, col="blue", pch=15, cex=2)
```



key-point: K-means clustering is super popular but can be mis-used. One big limitation is that it can impose a clustering pattern on your data even if clear natural grouping doesn't exist - i.e. it does what you tell it to do in terms of centers.

Heirarchical Clustering

The main function in “base” R for Heirarchical clustering is called `hclust()`. You must give “distance matrix” as input. We can get this from the `dist()` function in R.

You can't just pass our dataset as is into `hclust()`

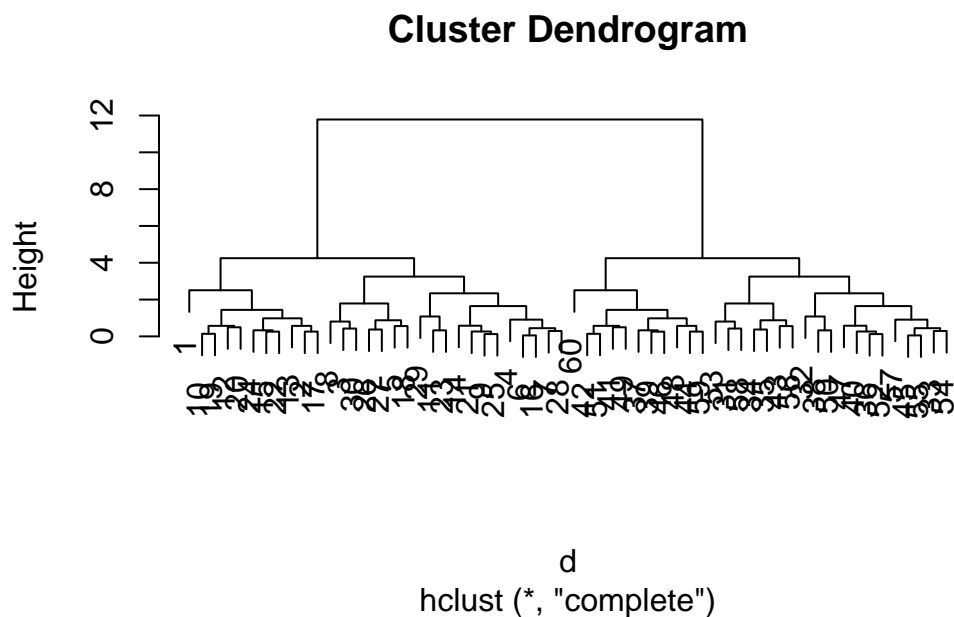

```
d <- dist(x)
hc <- hclust(d)
hc
```

Call:
hclust(d = d)

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

The results of `hclust()` don't have a useful `print()` method but do have a special `plot()` method.

```
plot(hc)
```



To get our main cluster assignment (membership vector) we need to “cut” the tree at the big gap posts...

```
cutree(hc, h=8)
```

```
[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
```

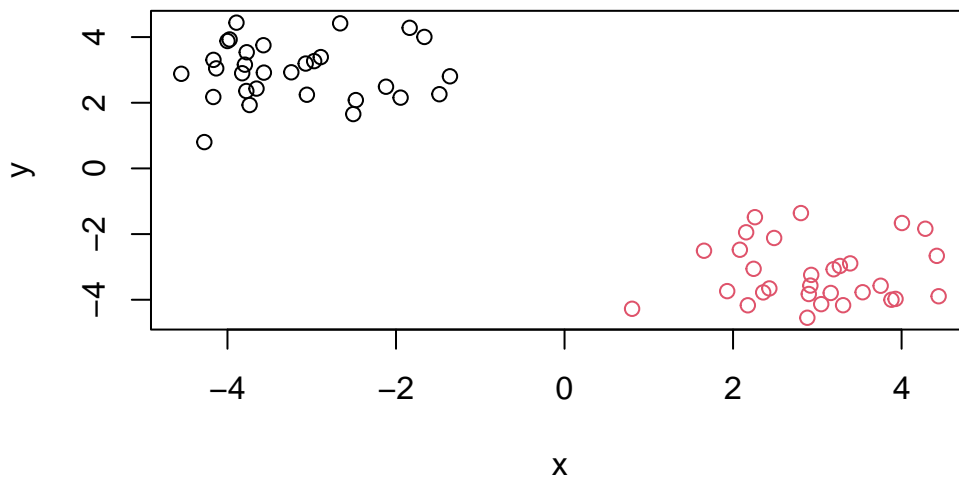
```
grps <- cutree(hc, h=8)
grps
```

```
[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 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
```

```
table(grps)
```

```
grps
 1  2
30 30
```

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



Hierarchical Clustering is distinct in that the dendrogram (tree figure) can reveal the potential grouping in your data (unlike K-means).

Principal Component Analysis (PCA)

PCA is a common and highly useful dimensionality reduction technique used in many fields - particularly bioinformatics.

Here we will analyze some data from the UK on food consumption.

Data Import

```
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

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

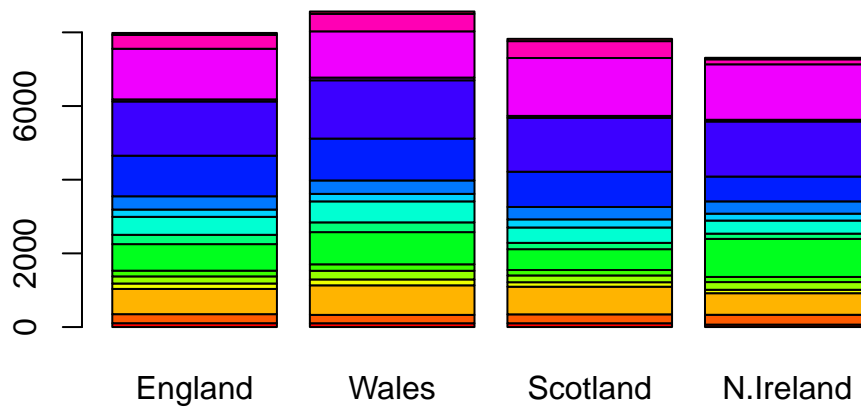
	England	Wales	Scotland	N.Ireland
Cheese	105	103	103	66
Carcass_meat	245	227	242	267
Other_meat	685	803	750	586
Fish	147	160	122	93
Fats_and_oils	193	235	184	209
Sugars	156	175	147	139

```
x <- read.csv(url, row.names = 1)
head(x)
```

	England	Wales	Scotland	N.Ireland
Cheese	105	103	103	66

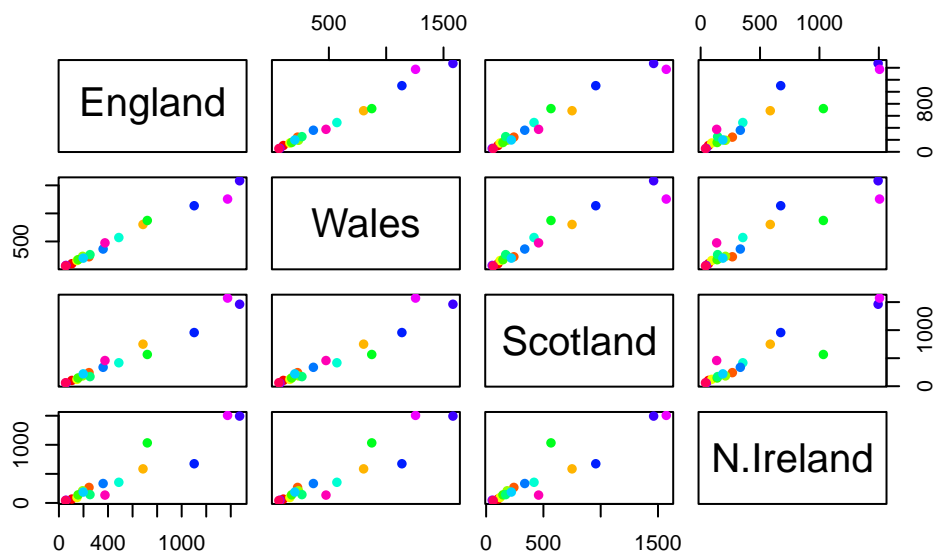
Carcass_meat	245	227	242	267
Other_meat	685	803	750	586
Fish	147	160	122	93
Fats_and_oils	193	235	184	209
Sugars	156	175	147	139

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



One conventional plot that can be useful is called a “pairs” plot.

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



PCA to the rescue

The main function in base R 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	2.921e-14
Proportion of Variance	0.6744	0.2905	0.03503	0.000e+00
Cumulative Proportion	0.6744	0.9650	1.00000	1.000e+00

The `prcomp()` function returns a list object of our results with five attributes/components

```
attributes(pca)
```

```
$names
```

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

```
$class
```

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

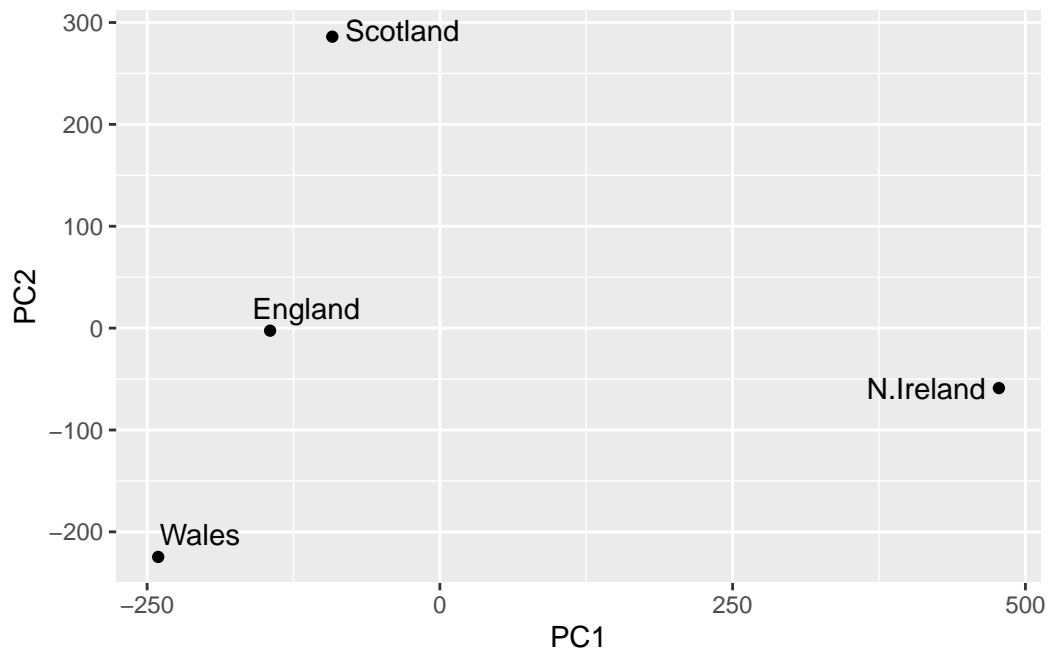
The two main “results” in here are `pca$x` and `pca$rotation`. The first of these (`pca$x`) contains the scores of the data on the new PC axis - we use these to make our “PCA plot”.

```
pca$x
```

	PC1	PC2	PC3	PC4
England	-144.99315	-2.532999	105.768945	-9.152022e-15
Wales	-240.52915	-224.646925	-56.475555	5.560040e-13
Scotland	-91.86934	286.081786	-44.415495	-6.638419e-13
N.Ireland	477.39164	-58.901862	-4.877895	1.329771e-13

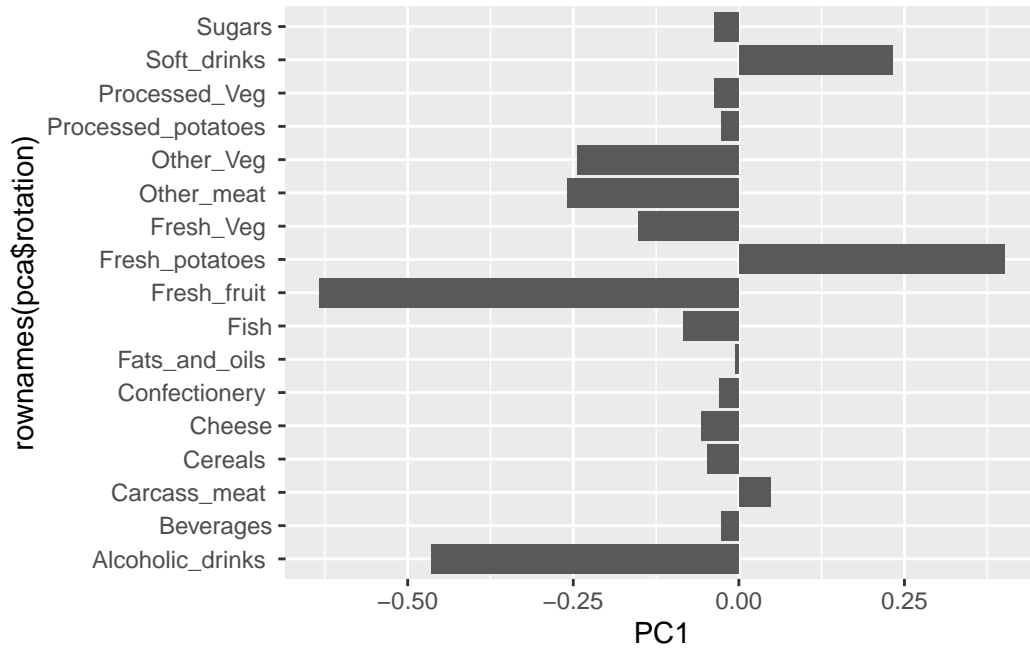
```
library(ggplot2)
library(ggrepel)

# Make a plot of pca$x with PC1 vs PC2
ggplot(pca$x) +
  aes(PC1, PC2, label=rownames(pca$x)) +
  geom_point() +
  geom_text_repel()
```

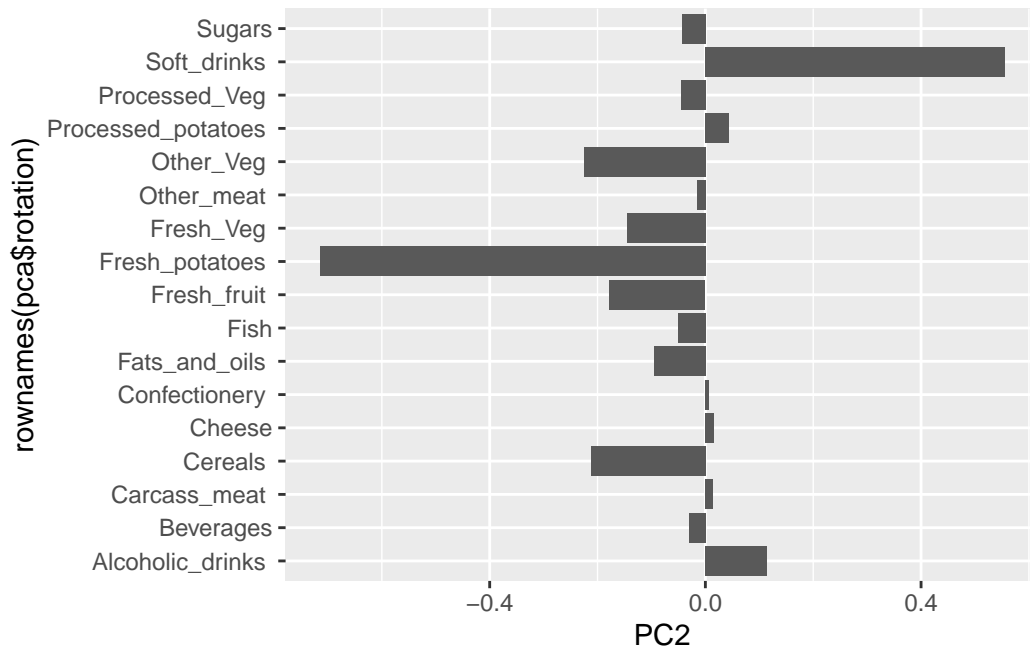


The second major result is contained in the `pca$rotation` object or component. Let's plot this to see what PCA is picking up...

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



```
ggplot(pca$rotation) +
  aes(PC2, rownames(pca$rotation)) +
  geom_col()
```



These plots show comparisons between the different countries and their food eating habits. It compares the different kinds of foods and helps us to understand their individual health habits as well. We saw that Ireland was an outlier when making a scatter plot from ggplot2, as well as plotted and compared the values from PC1 and PC2 as dimensionality reduction techniques to see where the differences deviate and are more similar. The generated bar plots from using geom_col helped to illustrate what stuck out strongly for comparison as well.