

Class 7: Machine Learning 1

Michael McClellan (PID: A169692395)

2025-10-25

Table of contents

Clustering	1
K-means	4
Hierarchical Clustering	8
Principal Component Analysis (PCA)	10
Data import	10
PCA to the rescue	13

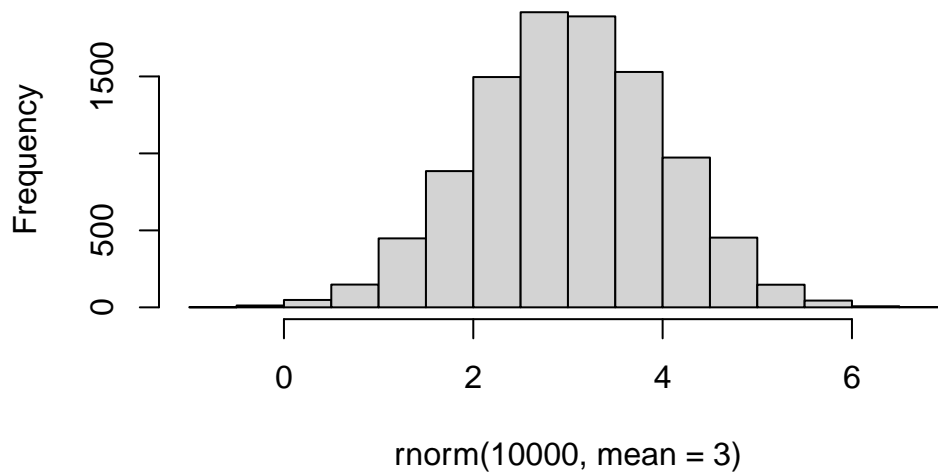
Today we will explore unsupervised machine learning methods starting with clustering and dimensionality reduction.

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

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

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

x
```

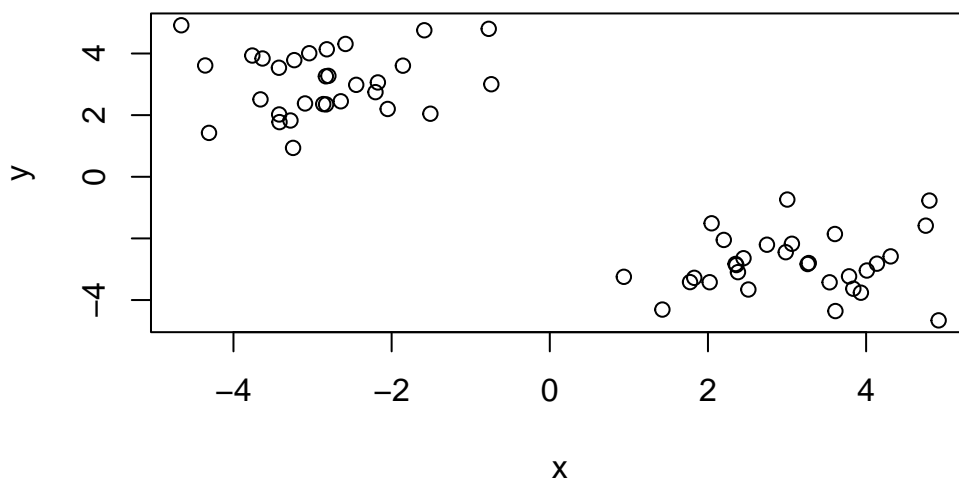
	x	y
[1,]	-3.2790995	1.8265134
[2,]	-0.7721671	4.8001937
[3,]	-3.6584019	2.5107826
[4,]	-2.7997913	3.2744855
[5,]	-1.5098985	2.0461795
[6,]	-2.8314615	3.2606003
[7,]	-2.8197801	4.1368119
[8,]	-4.6597462	4.9167964
[9,]	-3.4248179	3.5382925
[10,]	-0.7393091	3.0031464
[11,]	-3.6323763	3.8380702
[12,]	-2.6431007	2.4496259
[13,]	-3.0950977	2.3792966

[14,]	-2.2043866	2.7459713
[15,]	-3.2308835	3.7842662
[16,]	-1.5866685	4.7540845
[17,]	-1.8573270	3.6052253
[18,]	-3.0418971	4.0089483
[19,]	-4.3089416	1.4236195
[20,]	-2.4475529	2.9836882
[21,]	-3.4216146	2.0215698
[22,]	-2.1740021	3.0613737
[23,]	-3.2461727	0.9364243
[24,]	-2.8646660	2.3588774
[25,]	-2.5831496	4.3102518
[26,]	-3.4179945	1.7732562
[27,]	-3.7617574	3.9340042
[28,]	-2.0499697	2.2001789
[29,]	-4.3572554	3.6113855
[30,]	-2.8289070	2.3490229
[31,]	2.3490229	-2.8289070
[32,]	3.6113855	-4.3572554
[33,]	2.2001789	-2.0499697
[34,]	3.9340042	-3.7617574
[35,]	1.7732562	-3.4179945
[36,]	4.3102518	-2.5831496
[37,]	2.3588774	-2.8646660
[38,]	0.9364243	-3.2461727
[39,]	3.0613737	-2.1740021
[40,]	2.0215698	-3.4216146
[41,]	2.9836882	-2.4475529
[42,]	1.4236195	-4.3089416
[43,]	4.0089483	-3.0418971
[44,]	3.6052253	-1.8573270
[45,]	4.7540845	-1.5866685
[46,]	3.7842662	-3.2308835
[47,]	2.7459713	-2.2043866
[48,]	2.3792966	-3.0950977
[49,]	2.4496259	-2.6431007
[50,]	3.8380702	-3.6323763
[51,]	3.0031464	-0.7393091
[52,]	3.5382925	-3.4248179
[53,]	4.9167964	-4.6597462
[54,]	4.1368119	-2.8197801
[55,]	3.2606003	-2.8314615
[56,]	2.0461795	-1.5098985

```
[57,] 3.2744855 -2.7997913
[58,] 2.5107826 -3.6584019
[59,] 4.8001937 -0.7721671
[60,] 1.8265134 -3.2790995
```

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:

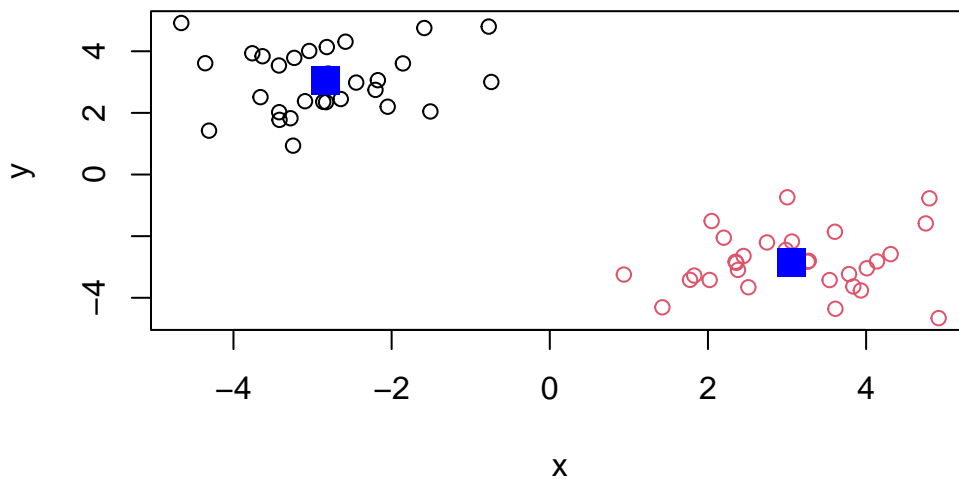
	x	y
1	-2.841606	3.061431


```
km$centers
```

```
      x      y
1 -2.841606  3.061431
2  3.061431 -2.841606
```

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 cluster into 4 groups/clusters and plot the same result figure as above.

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

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

Cluster means:

	x	y
1	-2.841606	3.061431
2	3.061431	-2.841606

Clustering vector:

[illegible]

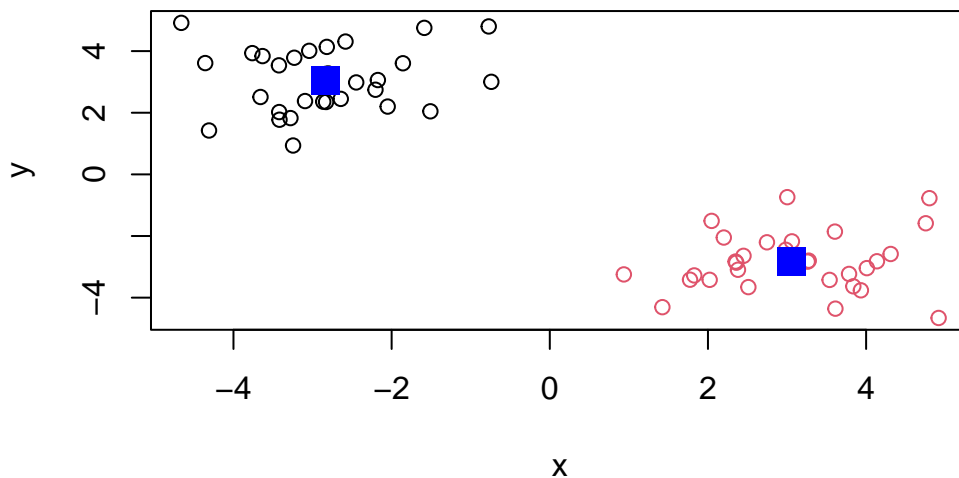
Within cluster sum of squares by cluster:

```
[1] 57.50848 57.50848
(between_SS / total_SS = 90.1 %)
```

Available components:

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

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



key-point: K-means clustering is super popular but can be misused. 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.

Hierarchical Clustering

The main function in “base” R for Hierarchical Clustering is called `hclust()`.

You can't just pass our dataset as is into `hclust()`, you must give “distance matrix” as input. We can get this from the `dist()` function in R.

```
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)
abline(h=8, col="red")
```



```
hclust (*, "complete")
```

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

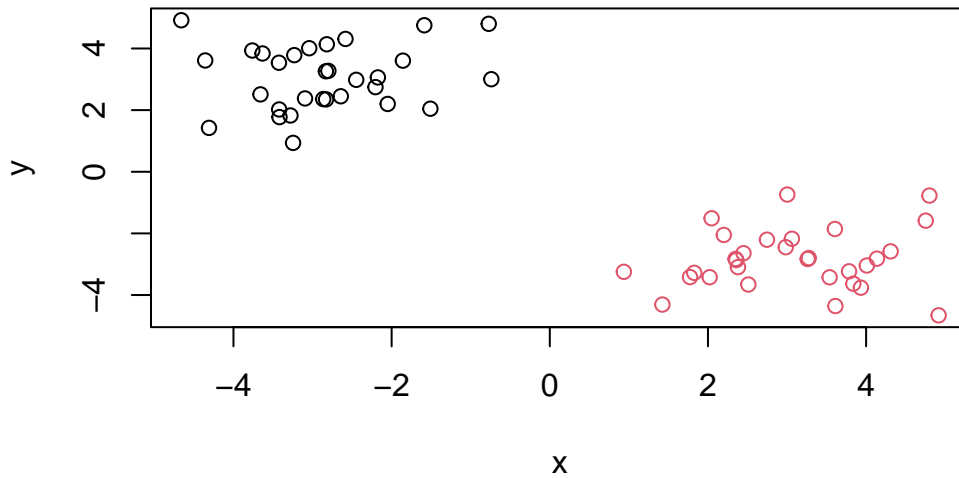
```
grps <- cutree(hc, h=8)
grps
```

```
[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 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
```

```
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)
```

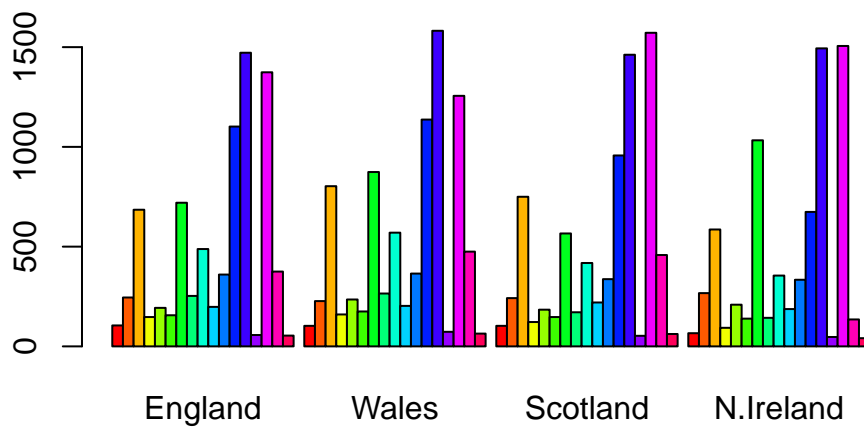
	X	England	Wales	Scotland	N.Ireland
1	Cheese	105	103	103	66
2	Carcass_meat	245	227	242	267
3	Other_meat	685	803	750	586

4	Fish	147	160	122	93
5	Fats_and_oils	193	235	184	209
6	Sugars	156	175	147	139

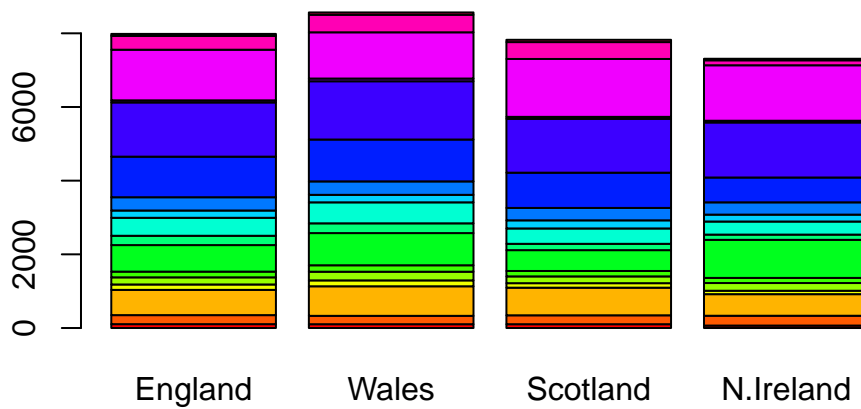
```
x <- read.csv(url, row.names = 1)
head(x)
```

	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=T, col=rainbow(nrow(x)))
```

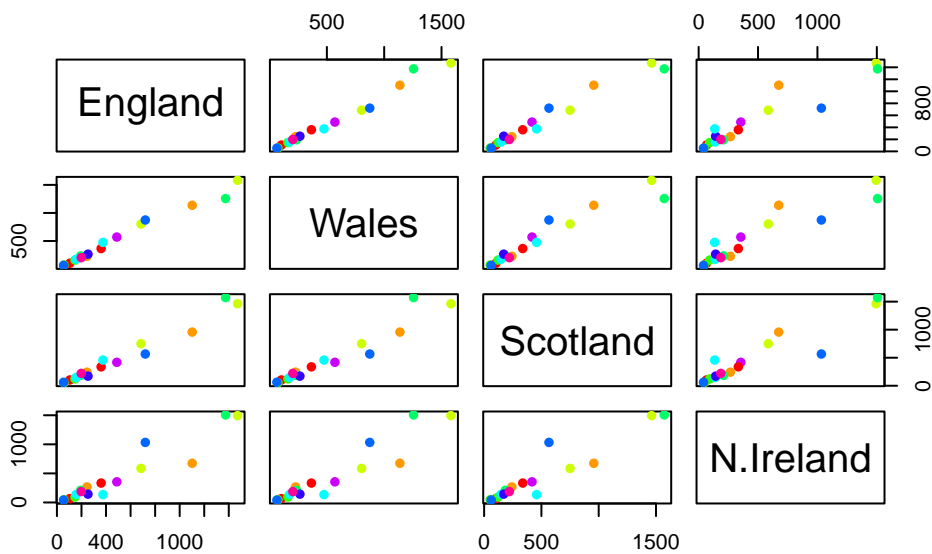


```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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

```
pairs(x, col=rainbow(10), 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"
```

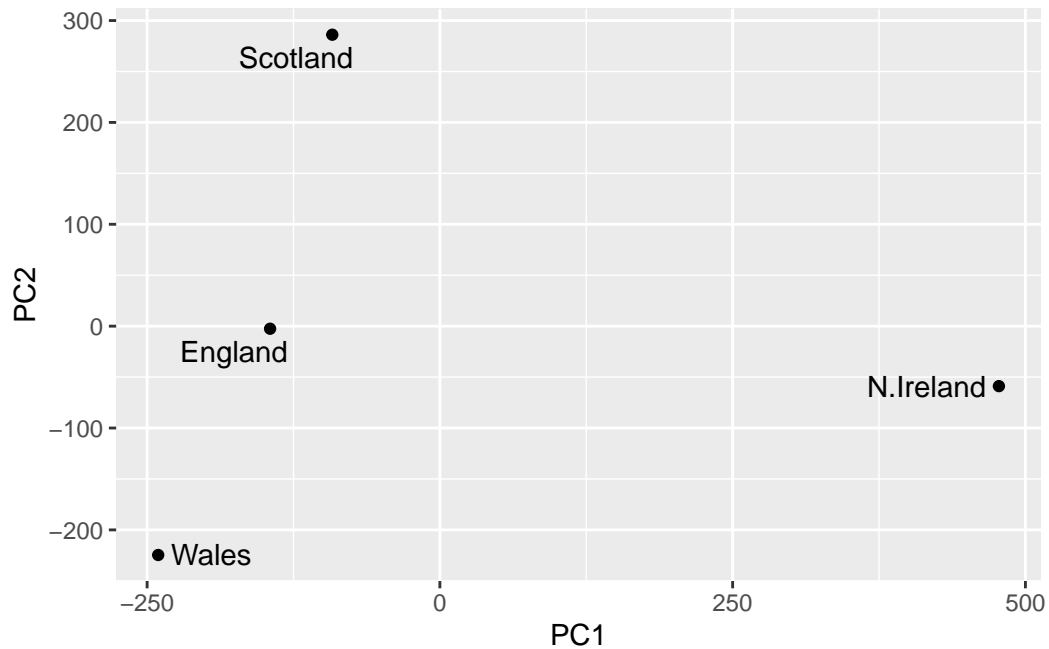
The two main “results” in here are `pca$x` and `pca$rotation`. The first of these (`pca$x`) contains the scores of the data on the new PC axis - we use these to make our “PCA plot”.

```
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
```

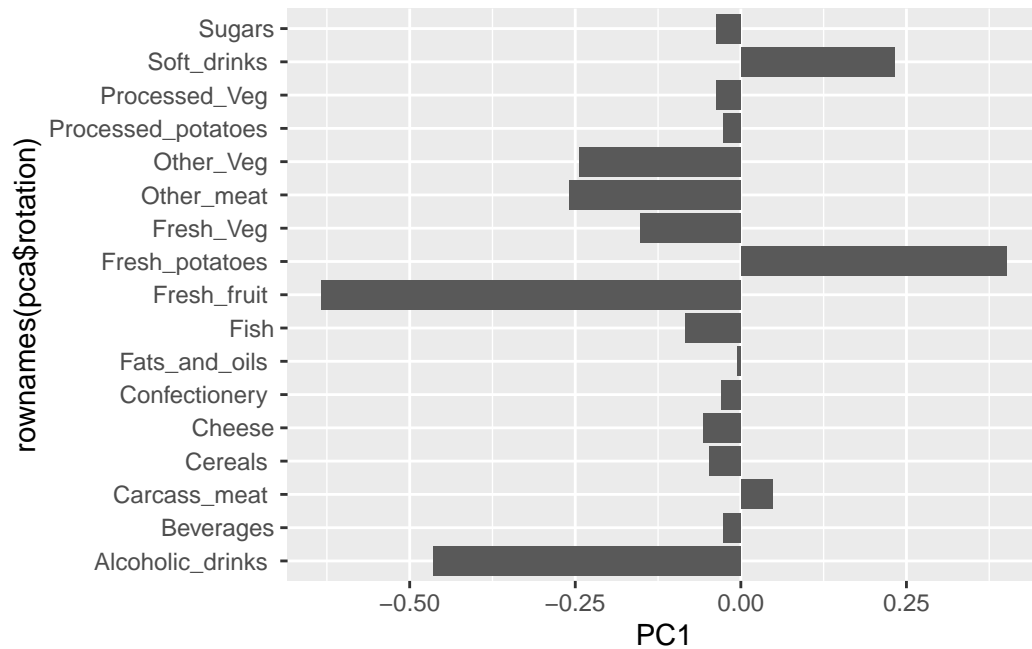
```
ggplot(pca$x) +
  aes(PC1, PC2, label=rownames(pca$x)) +
  geom_point() +
  geom_text_repel()
```



I think this plot shows that N. Ireland is very different from Wales and England and Scotland because the point is far from the other ones. I think it also shows that N.Ireland encompasses high variance.

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()
```



I think this plot works in conjunction with the other plot to show which foods are more common in different countries. For example, Fresh potatoes are more commonly eaten in N. Ireland, contributing more to the PC1 compared to England, Wales, and Scotland.