

Class7: Machine Learning 1

Rachel Lamm (PID:A18518313)

Table of contents

Background	1
K-means clustering	3
Hierarchical clustering	5
Principal Component Analysis (PCA)	9
PCA of UK food data	9
Spotting major differences and trends	10
Heatmap	12
PCA the rescue	13
Digging deeper (variable loadings)	17

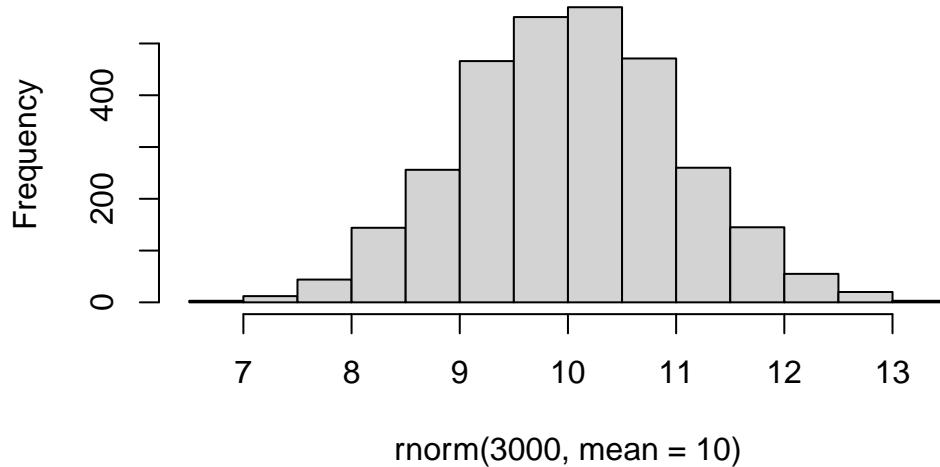
Background

Today we will begin our exploration of important machine learning methods with a focus on **clustering** and **dimensionallity reduction**.

To start testing these methods let's make up some sample data to cluster where we know what the answer should be.

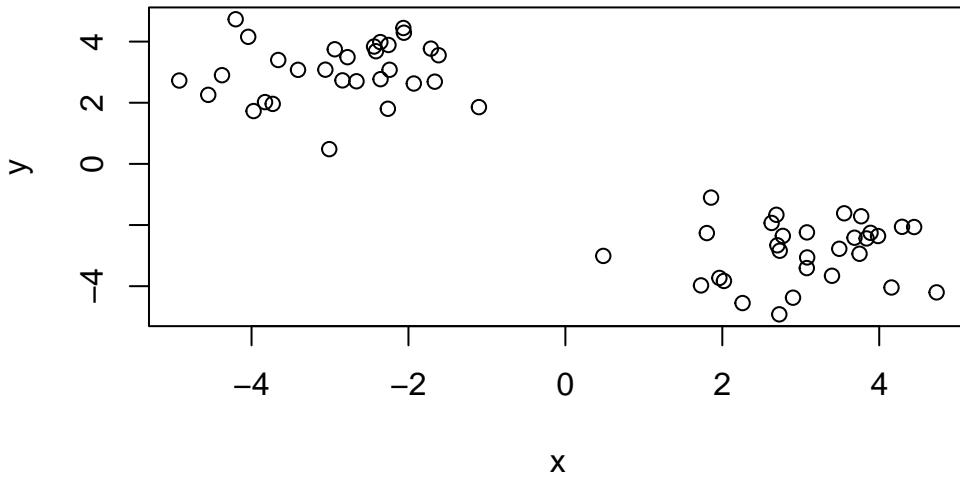
```
hist( rnorm(3000, mean=10) )
```

Histogram of rnorm(3000, mean = 10)



Q. Can you generate 30 numbers centered at +3 and 30 numbers at -3 taken at random from a normal distribution?

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



K-means clustering

The main function in “base R” for K-means clustering is called `kmeans()`, let’s try it out:

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

Q. What component of your `kmeans` result object has the cluster centers?

```
k$centers
```

	x	y
1	3.049351	-2.881904
2	-2.881904	3.049351

Q. What component of your `kmeans` result object has the cluster size (i.e. how many points are in each cluster)?

```
k$size
```

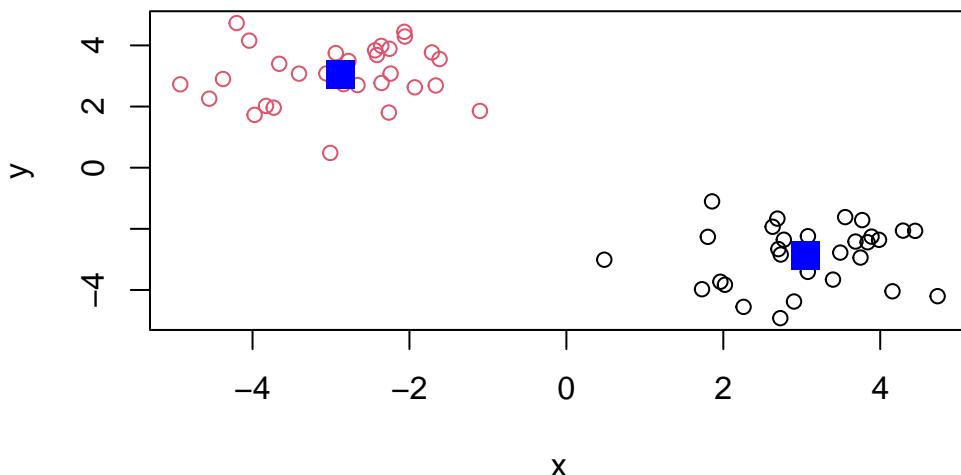
```
[1] 30 30
```

Q. What component of your kmeans result object has the cluster membership vector (i.e. the main clustering result: which points are in which cluster)?

k\$cluster

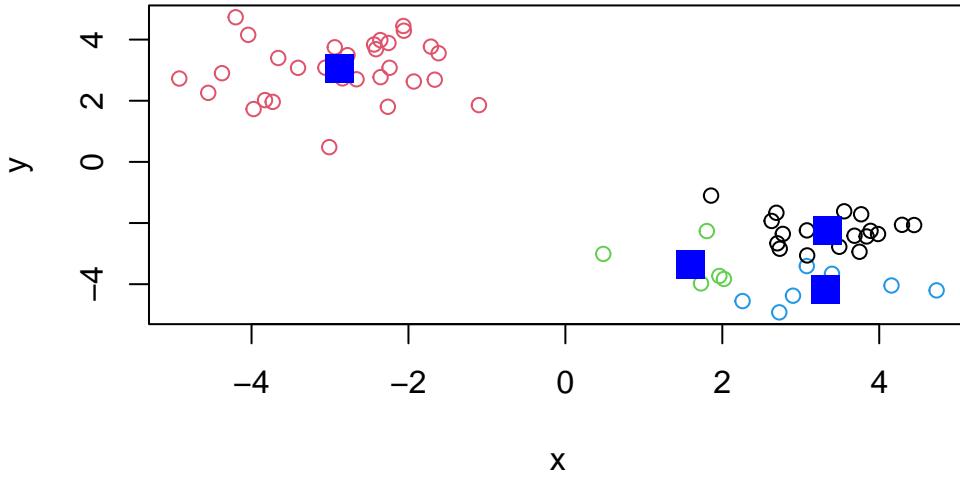
Q. Plot the results of clustering (i.e. our data colored by the clustering result) along with the cluster centers.

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



Q. Can you run `kmeans()` again and cluster `x` into 4 clusters and plot the results just like we did above with coloring by cluster and the cluster centers shown in blue?

```
k4<- kmeans(x, centers=4)
plot(x, col=k4$cluster)
points(k4$centers, col="blue", pch=15, cex=2)
```



Key-point: Kmeans will always return the clustering that we ask for (this is the “K” or “centers” in K-means)!

```
k$tot.withinss
```

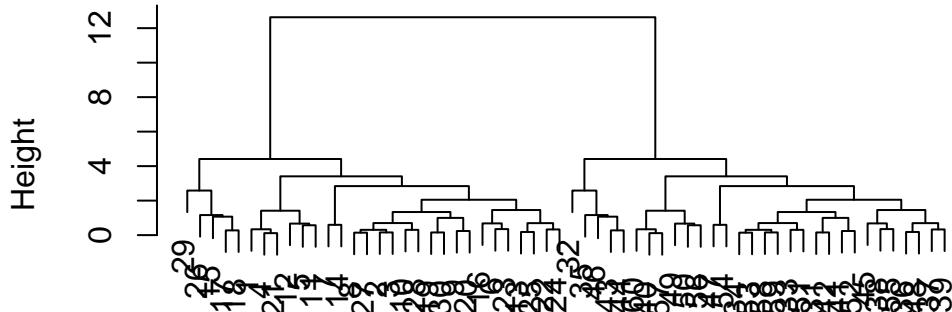
```
[1] 109.4512
```

Hierarchical clustering

The main function for Hierarchical clustering in base R is called `hclust()`. One of the main differences with respect to the `kmeans()` function is that you can not just pass your input data directly to `hclust()` - it needs a “distance matrix” as input. We can get this from lot’s of places including the `dist()` function.

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

Cluster Dendrogram

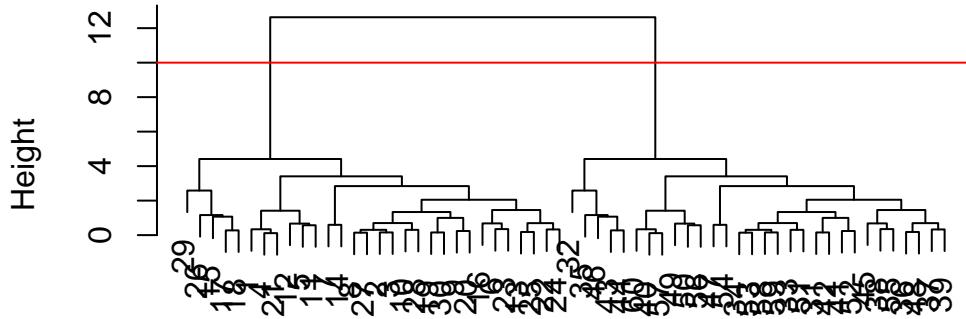


```
d  
hclust (*, "complete")
```

We can “cut” the dendrogram or “tree” at a given height to yield our “clusters”. For this we use the function `cutree()`

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

Cluster Dendrogram



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

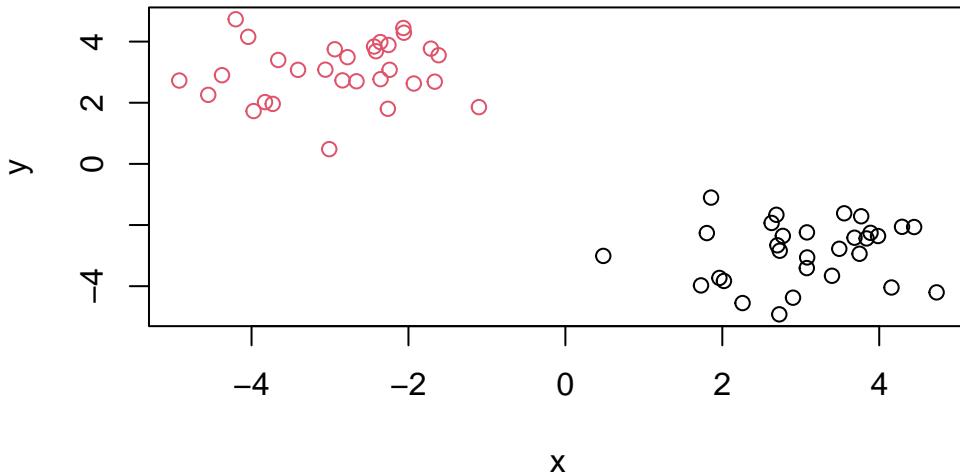
```
grps<- cutree(hc,h=10)
```

```
grps
```

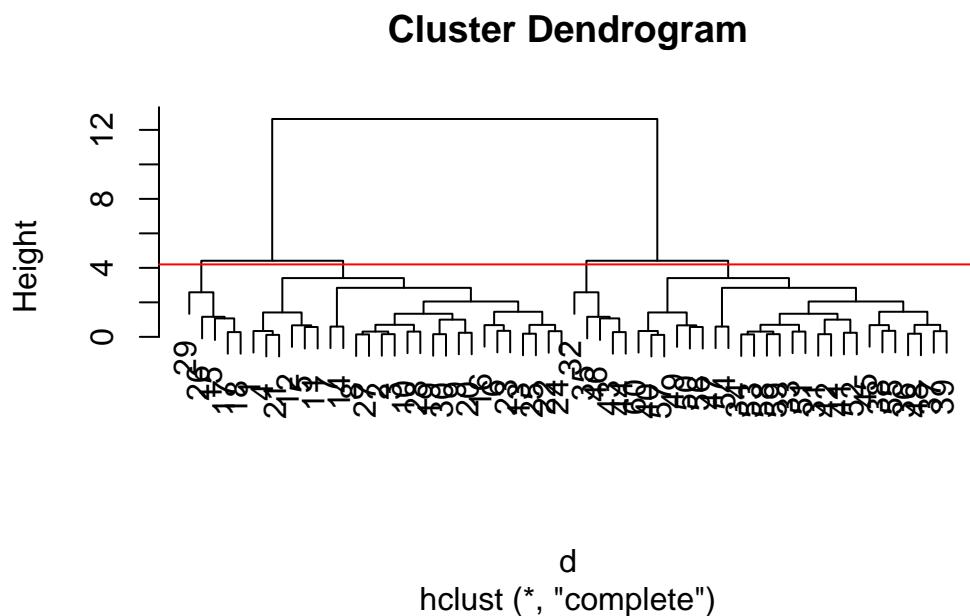
```
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2  
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Q. Plot our data `x` colored by the clusterinv result from `hclust()` and `cutree()`?

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



```
plot(hc)
abline(h=4.2, col="red")
```



```
cutree(hc, h=4.2)
```

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

Principal Component Analysis (PCA)

PCA is a popular dimensionality reduction technique that is widely used in bioinformatics.

PCA of UK food data

Read data on food consumption in the UK

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

It looks like the row names are not set properly. We can fix this

```
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

A better way to do this is fix the row names assignment at import time:

```
x<- read.csv(url, 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?

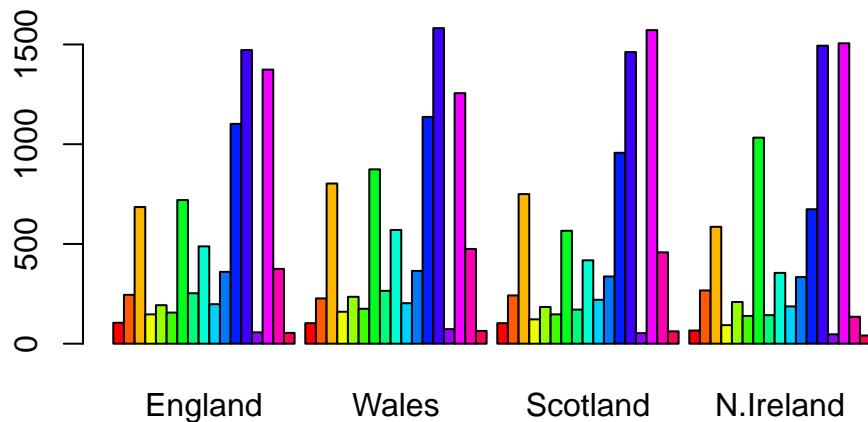
```
dim(x)
```

```
[1] 17 4
```

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?

Spotting major differences and trends

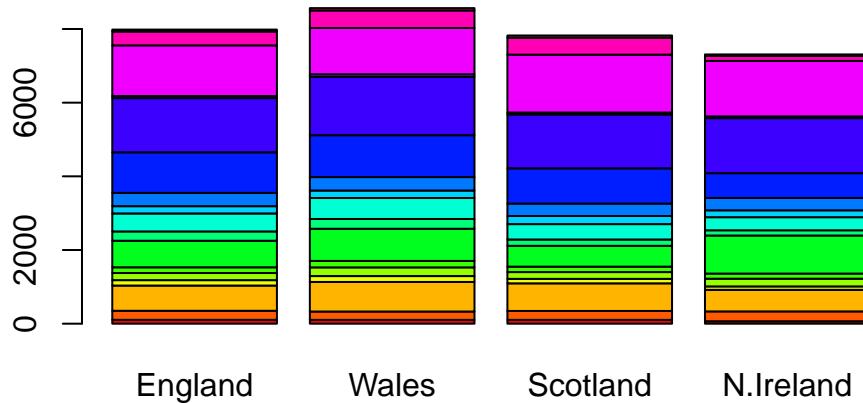
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

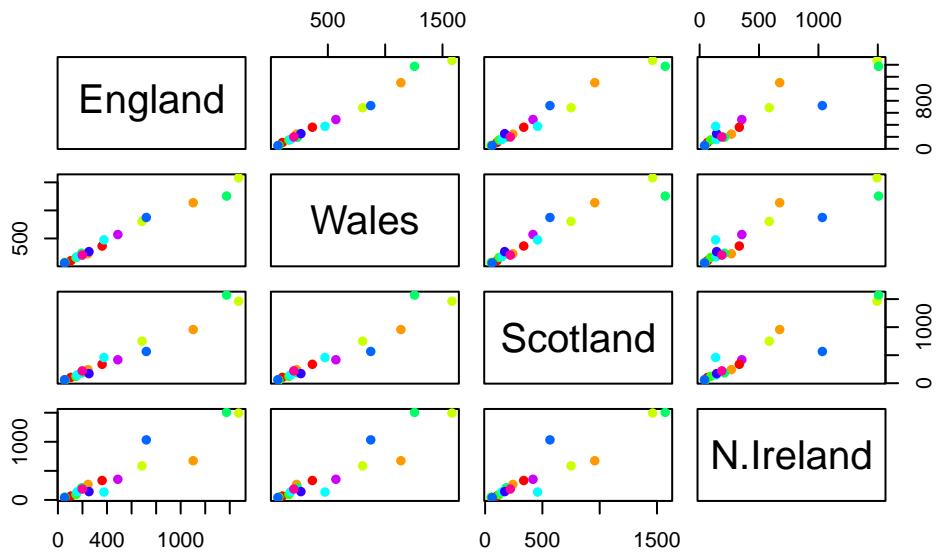
- You would want to change the beside column from true to false so they wil stack instead of go next to each other.

```
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?

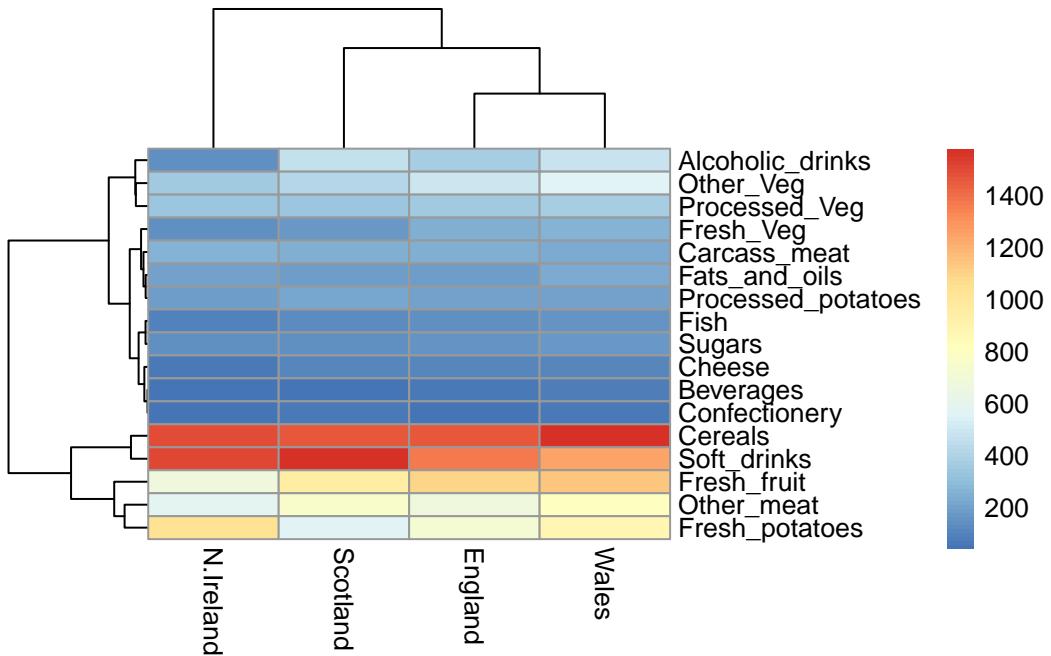
```
pairs(x, col=rainbow(10), pch=16)
```



Heatmap

We can install the **heatmap** package with the `install.packages()` command that we used previously. Remember

```
library(pheatmap)
pheatmap( as.matrix(x) )
```



Of all these plot really only the `pairs()` plot was useful. This however took a bit of work to interpret and will to scale when I am looking at much bigger datasets.

PCA 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	2.7e-14
Proportion of Variance	0.6744	0.2905	0.03503	0.0e+00
Cumulative Proportion	0.6744	0.9650	1.00000	1.0e+00

Q. How much variance is captured in the first PC?

67.4%

Q. How many PCs do I need to capture at least 90% of the total variance in the dataset?

Two PCs capture 96.5% of the total variance.

Q. Plot our main PCA result. Folks can call this different things depending on their field of study e.g. “PC plot”, “ordination plot” “Score plot”, “PC1 vs. PC2 plot”...

```
attributes(pca)
```

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

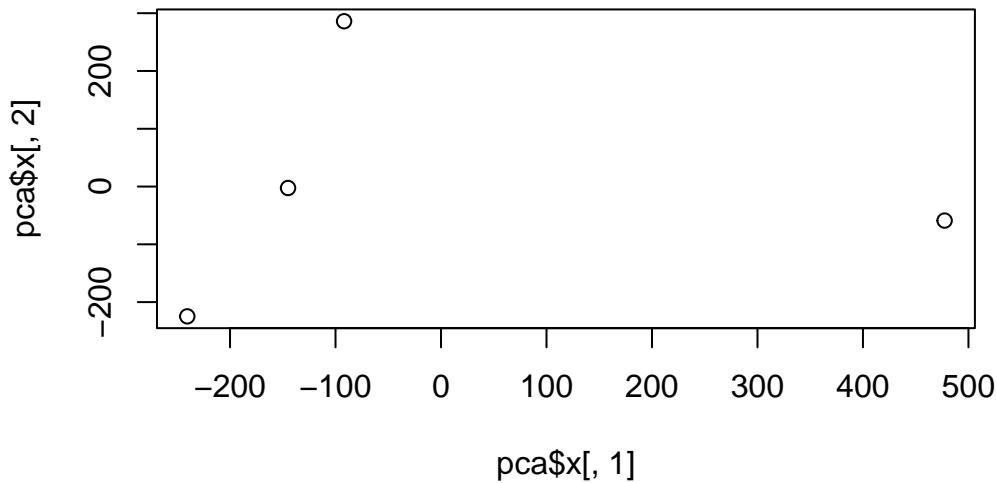
$class
[1] "prcomp"
```

To generate our PCA score plot we want the `pca$x` component of the result object

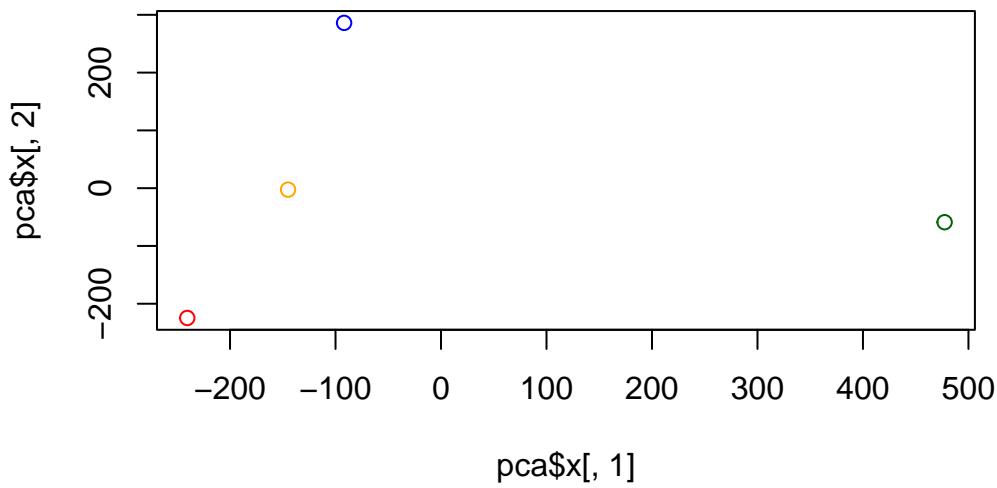
```
pca$x
```

	PC1	PC2	PC3	PC4
England	-144.99315	-2.532999	105.768945	1.612425e-14
Wales	-240.52915	-224.646925	-56.475555	4.751043e-13
Scotland	-91.86934	286.081786	-44.415495	-6.044349e-13
N.Ireland	477.39164	-58.901862	-4.877895	1.145386e-13

```
plot(pca$x[,1],pca$x[,2])
```



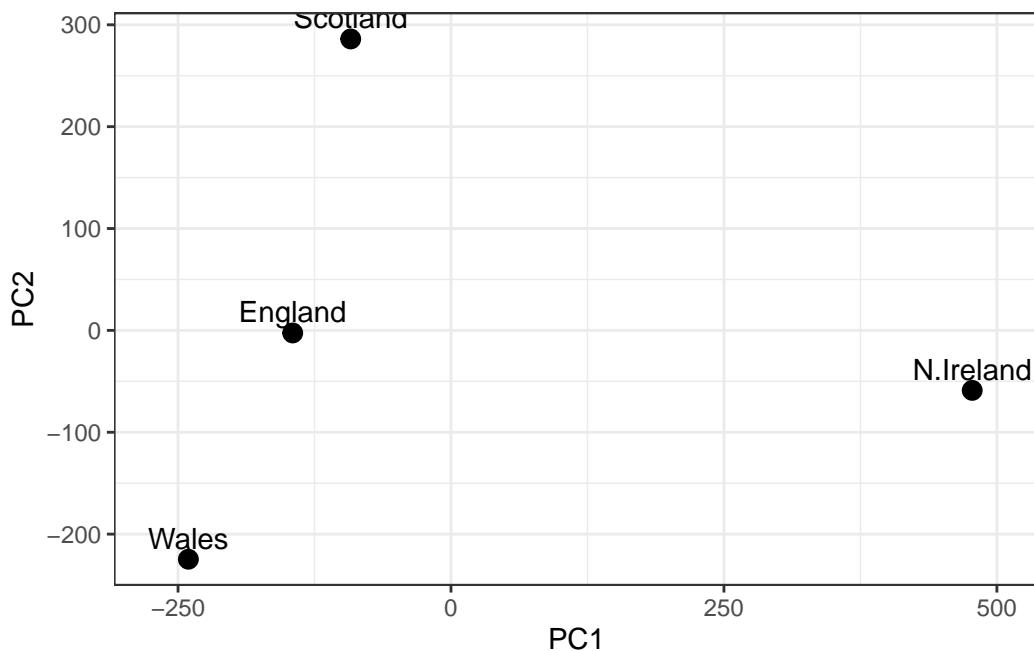
```
my_cols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=my_cols)
```



```

df <- as.data.frame(pca$x)
df$Country <- rownames(df)
library(ggplot2)
# Plot PC1 vs PC2 with ggplot
ggplot(pca$x) +
  aes(x = PC1, y = PC2, label = rownames(pca$x)) +
  geom_point(size = 3) +
  geom_text(vjust = -0.5) +
  xlim(-270, 500) +
  xlab("PC1") +
  ylab("PC2") +
  theme_bw()

```



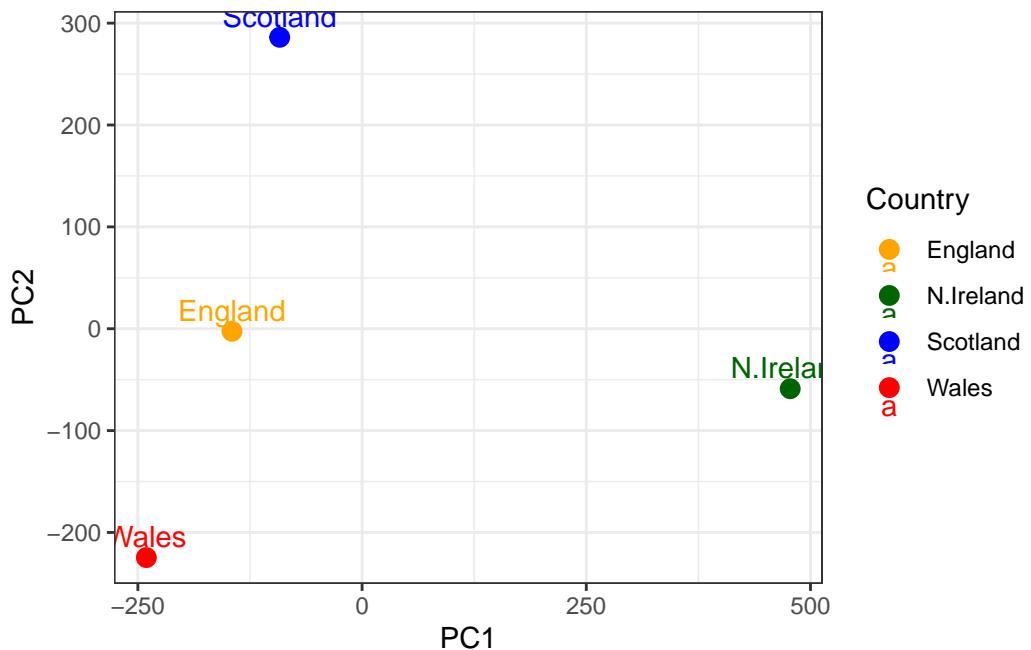
```

countries <- c("England", "Scotland", "Wales", "N.Ireland")
country_colors <- c(
  "England" = "orange",
  "Scotland" = "blue",
  "Wales" = "red",
  "N.Ireland" = "darkgreen")

library(ggplot2)

```

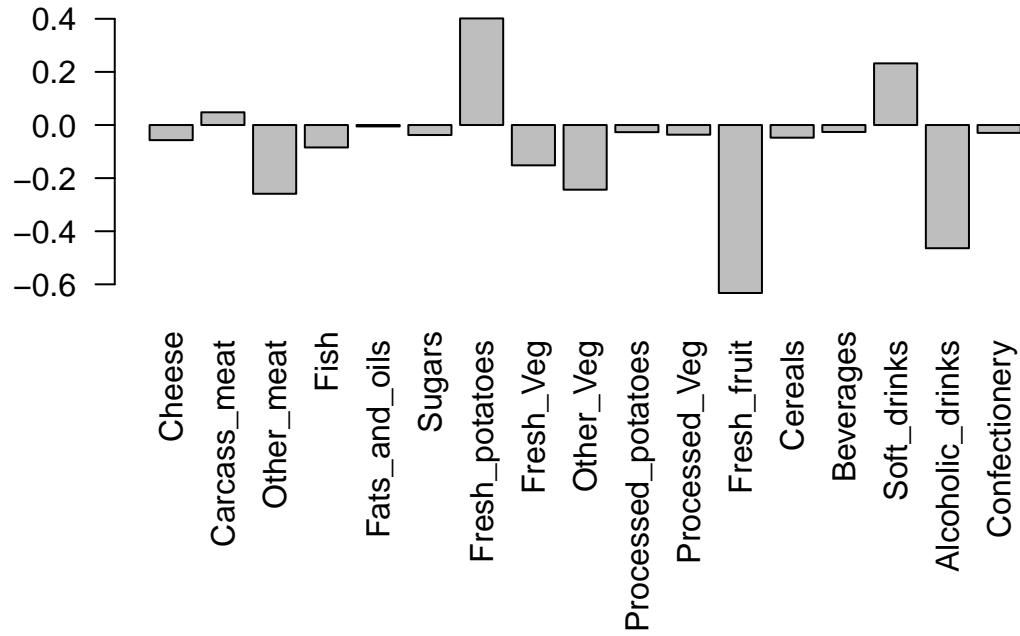
```
ggplot(df, aes(x = PC1, y = PC2, color = Country, label = Country)) +
  geom_point(size = 3) +
  geom_text(vjust = -0.5) +
  scale_color_manual(values = country_colors) +
  xlab("PC1") +
  ylab("PC2") +
  theme_bw()
```



Digging deeper (variable loadings)

How do the original variables (i.e. the 17 different foods) contribute to our new PCs?

```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

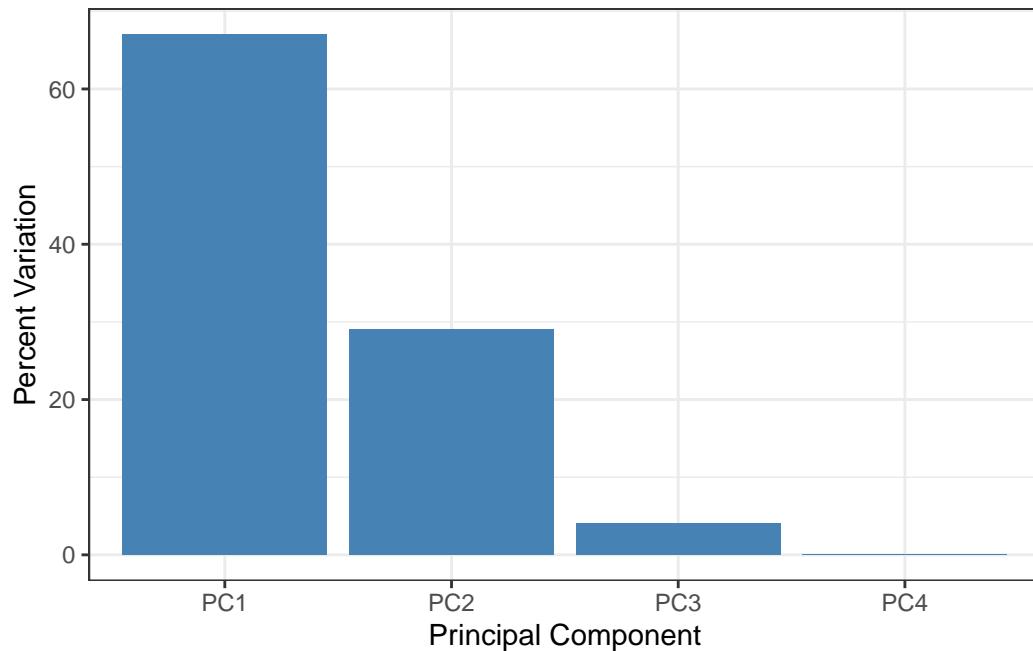
```
[1] 67 29 4 0
```

```
z <- summary(pca)
z$importance
```

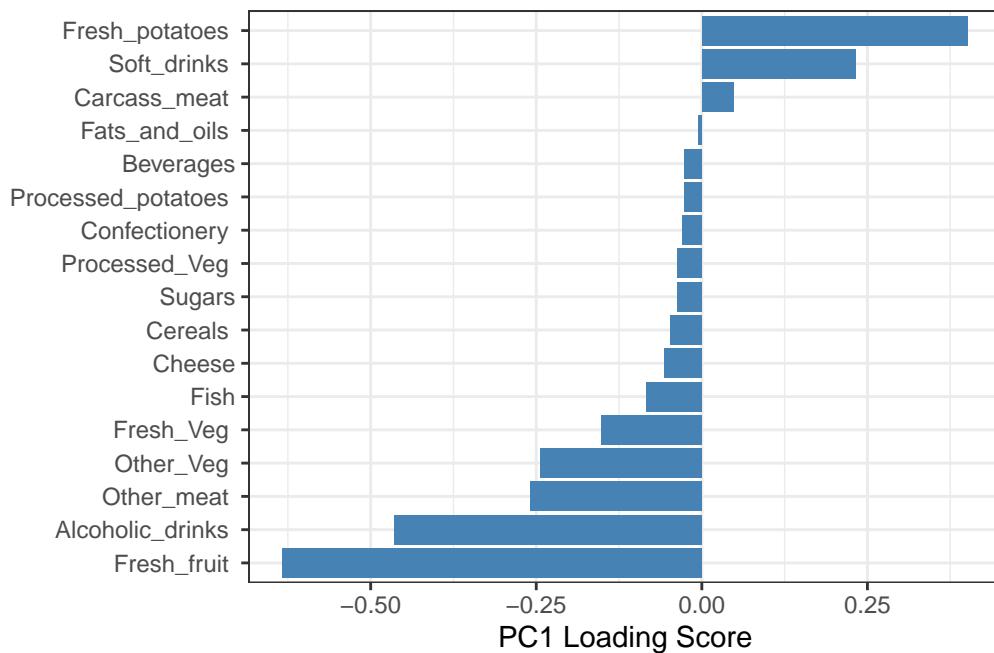
	PC1	PC2	PC3	PC4
Standard deviation	324.15019	212.74780	73.87622	2.699876e-14
Proportion of Variance	0.67444	0.29052	0.03503	0.000000e+00
Cumulative Proportion	0.67444	0.96497	1.00000	1.000000e+00

```
variance_df <- data.frame(
  PC = factor(paste0("PC", 1:length(v)), levels = paste0("PC", 1:length(v))),
  Variance = v
)
ggplot(variance_df) +
  aes(x = PC, y = Variance) +
  geom_col(fill = "steelblue") +
  xlab("Principal Component") +
```

```
ylab("Percent Variation") +  
theme_bw() +  
theme(axis.text.x = element_text(angle = 0))
```



```
ggplot(pca$rotation) +  
aes(x = PC1,  
y = reorder(rownames(pca$rotation), PC1)) +  
geom_col(fill = "steelblue") +  
xlab("PC1 Loading Score") +  
ylab("") +  
theme_bw() +  
theme(axis.text.y = element_text(size = 9))
```



```
ggplot(pca$rotation) +
  aes(x = PC2,
      y = reorder(rownames(pca$rotation), PC2)) +
  geom_col(fill = "steelblue") +
  xlab("PC2 Loading Score") +
  ylab("") +
  theme_bw() +
  theme(axis.text.y = element_text(size = 9))
```

