Class 7: Machine Learning I

Katie

In this class we will explore clustering and dimensionality reduction methods.

K-means

Make up some input data where we know what the answer should be.

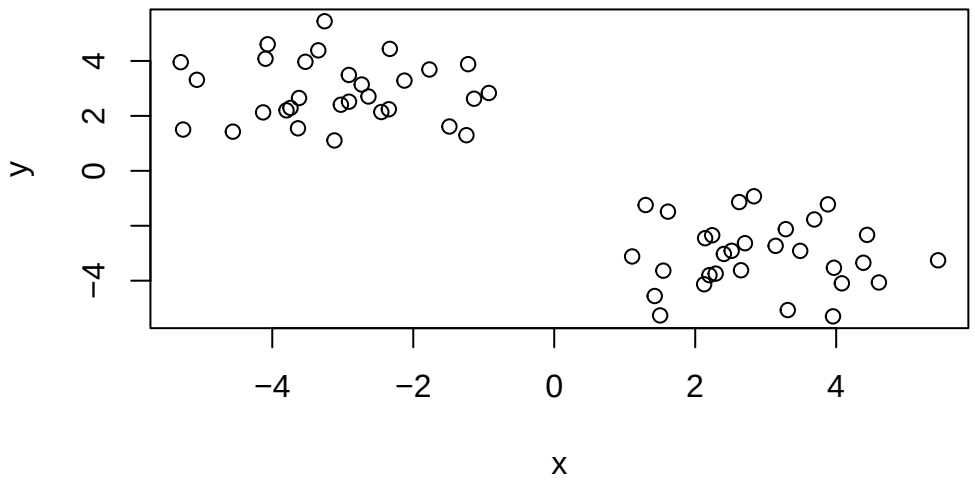
```
tmp <- c(rnorm(30, -3), rnorm(30, +3))
x <- cbind(x=tmp, y=rev(tmp))
head(x)
```

```
              x        y
[1,] -4.1303433 2.126936
[2,] -0.9275396 2.833164
[3,] -5.0697325 3.313276
[4,] -3.3469899 4.385616
[5,] -1.4872565 1.613397
[6,] -2.9136585 3.489752
```

```
#x <- cbind(tmp, rev(tmp))
```

Quick plot of x to see the two group at -3,+3 and +3,-3

```
plot(x)
```



Use the `kmeans()` function setting k to 2 and nstart=20

```
km <- kmeans(x, centers=2, nstart = 20)
km
```

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

Cluster means:
          x         y
1  2.896915 -3.060007
2 -3.060007  2.896915

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] 79.42828 79.42828
 (between_SS / total_SS =  87.0 %)

Available components:
```

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

Q. How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

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

```
km$cluster
```

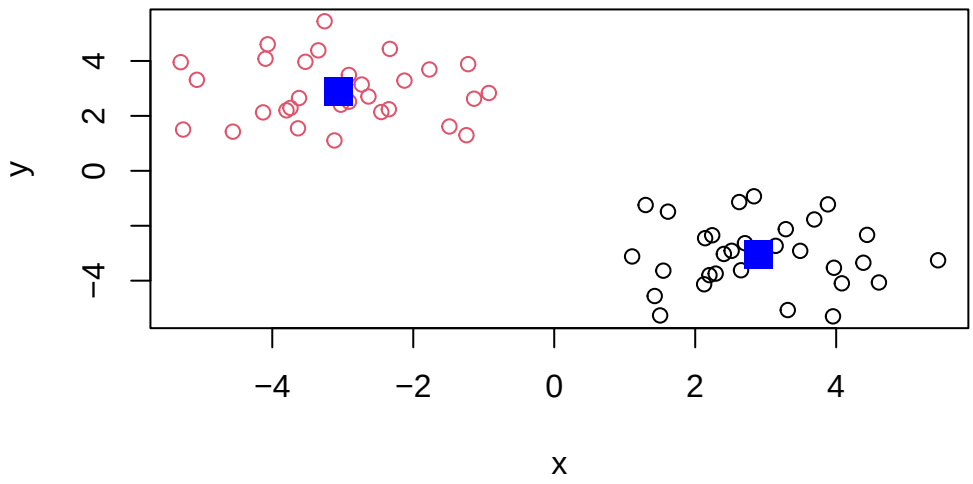
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
km$centers
```

```
          x         y
1  2.896915 -3.060007
2 -3.060007  2.896915
```

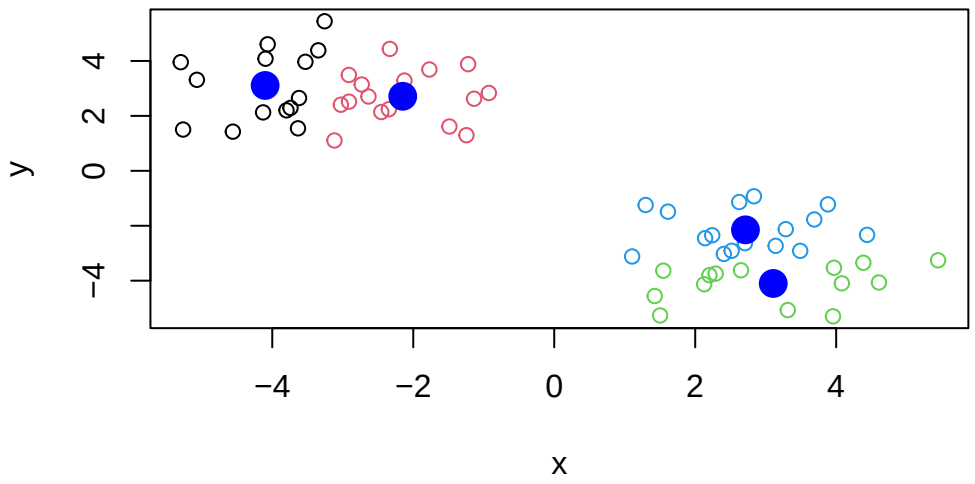
Q. Plot x colored by kmeans cluster assignment and add cluster centers as blue points

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



Play with kmeans and ask for different number of clusters

```
km <- kmeans(x, centers=4, nstart=20)
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=16, cex=2)
```



Hierarchial Clustering

This is another very useful and widely employed clustering methid which has the advantage over k-means in that it can help reveal something of the true grouping in your data.

The `hclust()` function wants a distance matrix as input. We can get this from the `dist()` function.

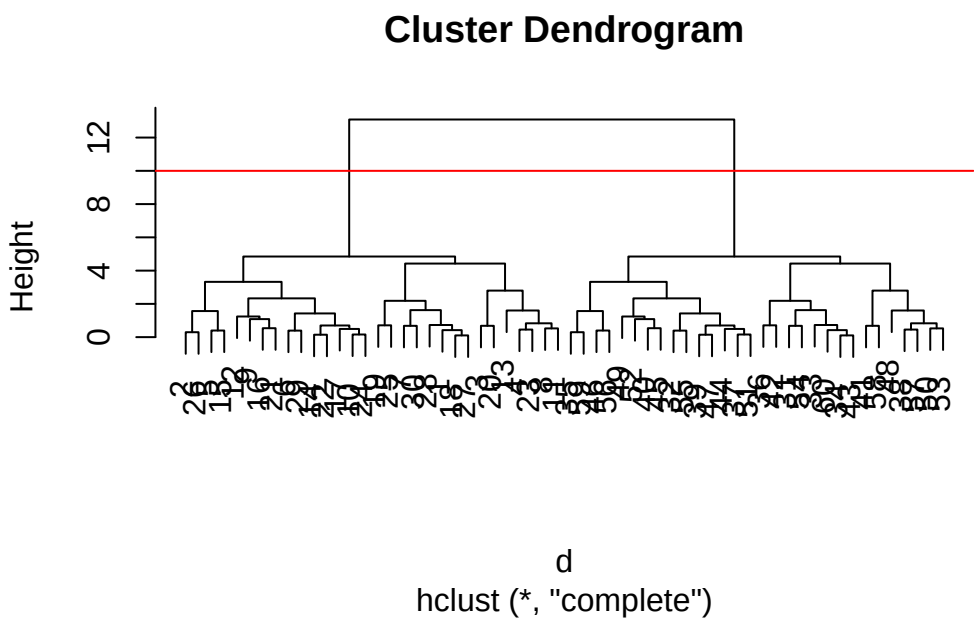
```
d <- dist(x)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

There is a plot method for hclust results:

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

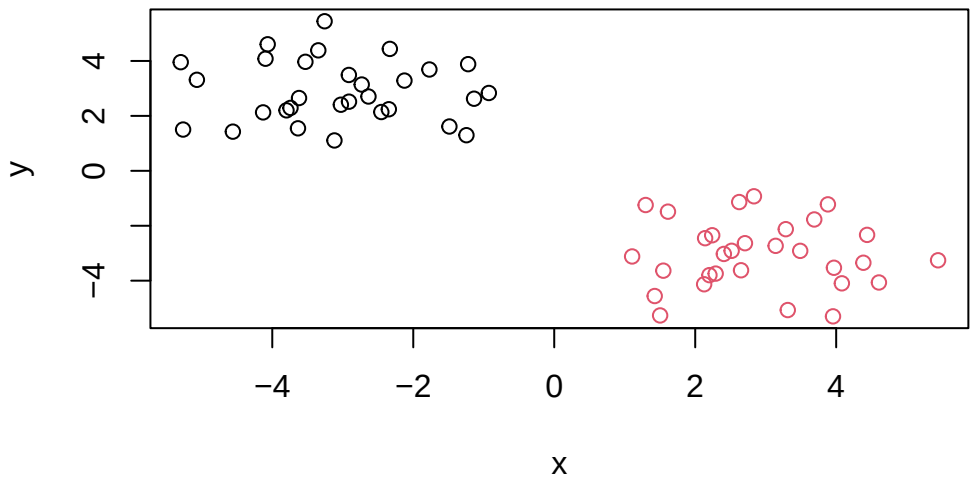


To get my cluster membership vector I need to "cut" my tree to yield sub-trees or branches with all the members of a given cluster reading on the same cut branch. The function to do this is called `cutree()`

```
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
```

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



It is often helpful to use the `k=` argument to cutree rather than the `h=` height of cutting with `cutree()`. This will cut the tree to yield the number of clusters you want.

```
cutree(hc, k=4)
```

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

Principal Component Analysis (PCA)

principal like most important

The R function for PCA is called `prcomp()`

PCA of UK food data

```
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
```

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  5
```

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

#dim is rows and colums nrow and ncol gives them seperately.

```
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
```

```
dim(x)
```

```
[1] 17  4
```

```
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
```

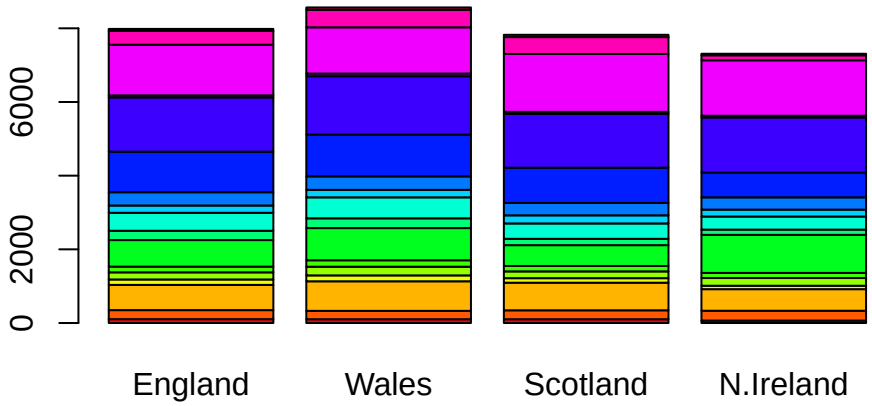
```
dim(x)
```

```
[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?

The second approach is better because the first approach deletes a column each time you run it, as a result of the x <- x[,-1], so you have to you have to set the row.names argument of read.csv() to be the first column. The second one is more robust.

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

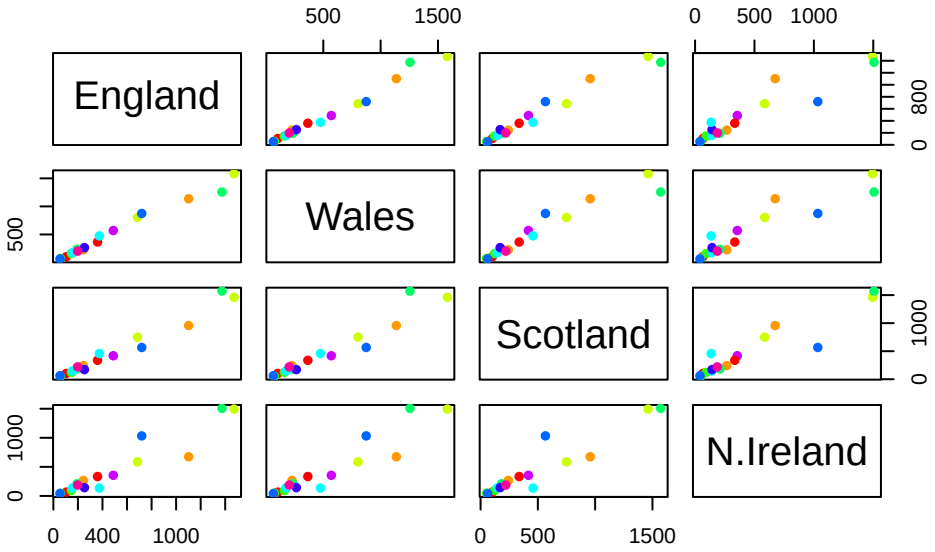



```
?barplot
```

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

When changing the beside=FALSE in the barplot() the columns of height are portrayed as stacked bars.

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



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 figures compare England, Wales, Scotland, and N.Ireland two at a time.It plots all the countries against each other. If a given point lies on the diagonal for a given plot, it means the two countries being compared have a significantly similar numerical value at the same category.

Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

The main differnces between N.Ireland and the other countries is that the dark blue and orange points stray further from the diagonal line.

```
pca <- prcomp( t(x) )
summary(pca)
```

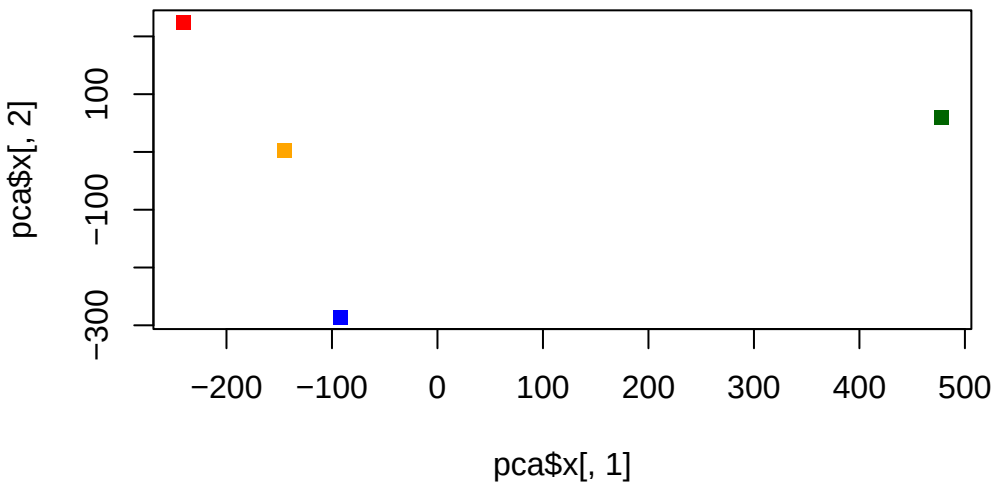
```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 5.552e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

A “PCA plot” (a.k.a “Score plot”, PC1vsPC2 plot, etc.)

```
pca$x
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  1.042460e-14
Wales     -240.52915  224.646925   56.475555  9.556806e-13
Scotland   -91.86934 -286.081786   44.415495 -1.257152e-12
N.Ireland  477.39164   58.901862    4.877895  2.872787e-13
```

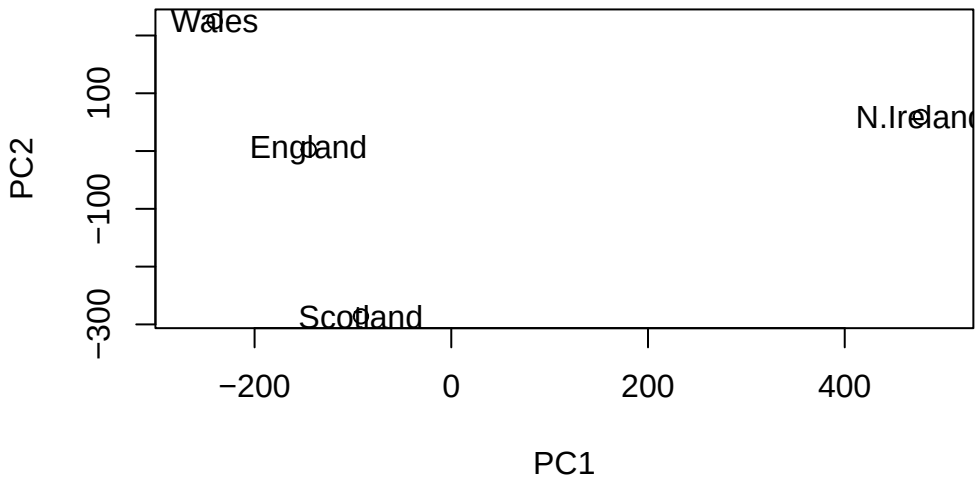
```
plot(pca$x[,1], pca$x[,2], col=c("orange", "red", "blue", "darkgreen"),pch=15)
```



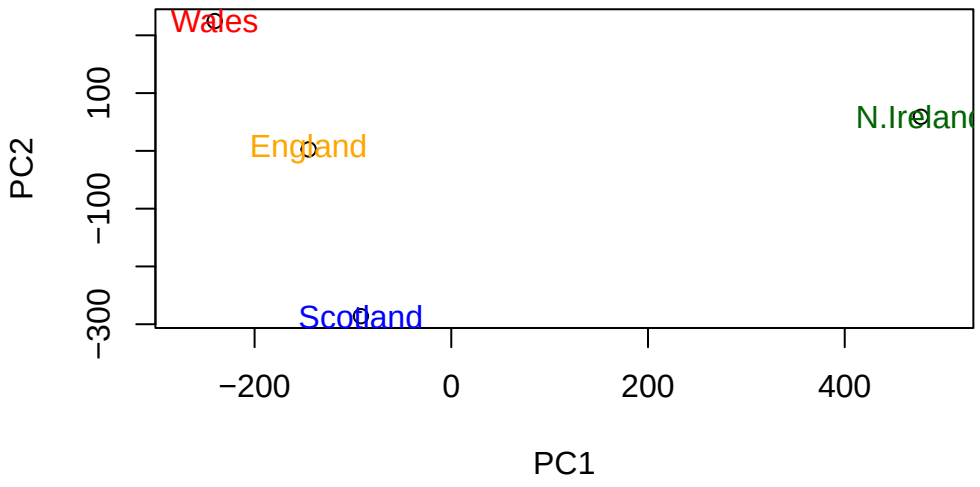
this one plot shows me in terms of the first axis PC1 is most important.

> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points. Hide

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```




```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x),col=c("orange", "red", "blue", "darkgreen"), pch=15
```



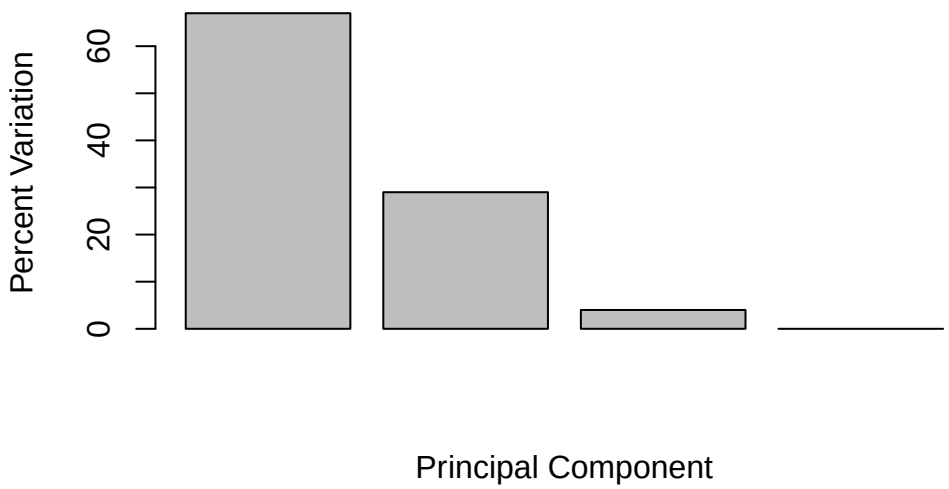
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

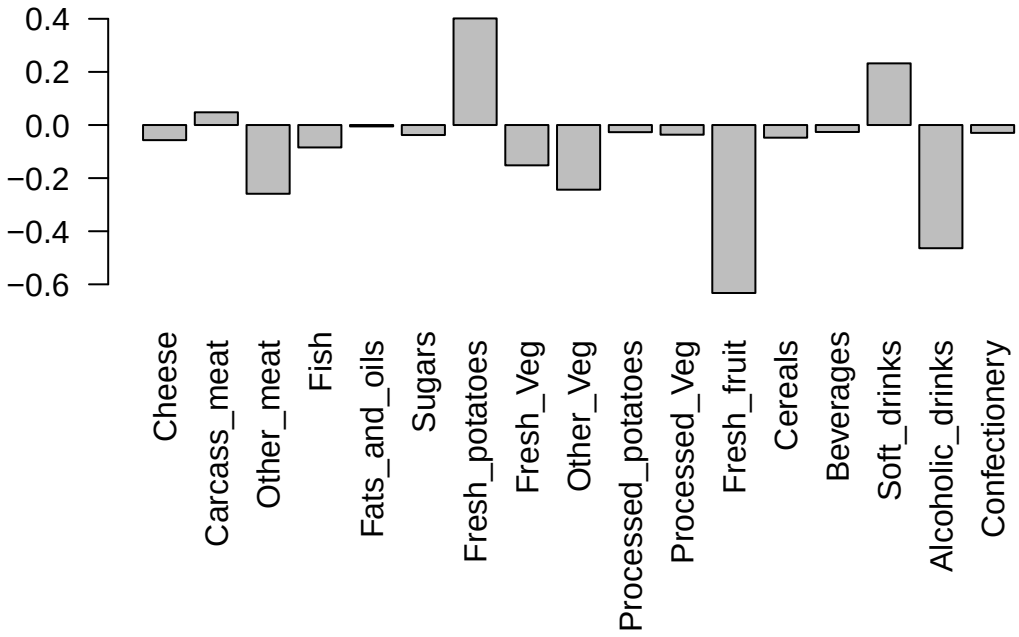
```
z <- summary(pca)
z$importance
```

```
                            PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 5.551558e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

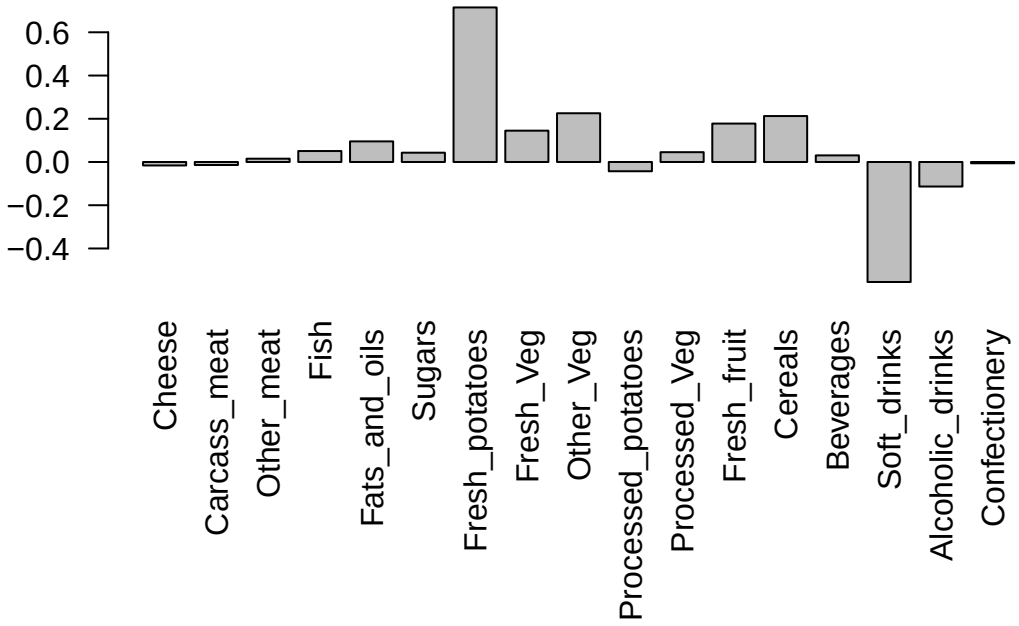
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```




```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```




```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```


0.6
0.4
0.2
0.0
−0.2
−0.4
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery