class 7: Machine Learning 1

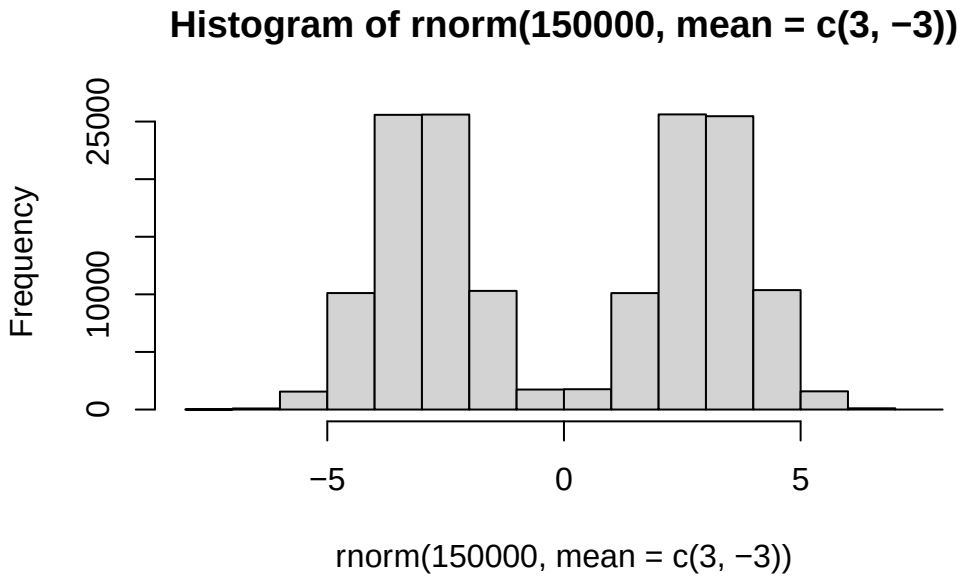
Eli Sobel A69027989

Before we get into clustering methods, let's make some sample data to cluster where we know what the answer should be.

To help with this, I will use the `rnorm()` function.

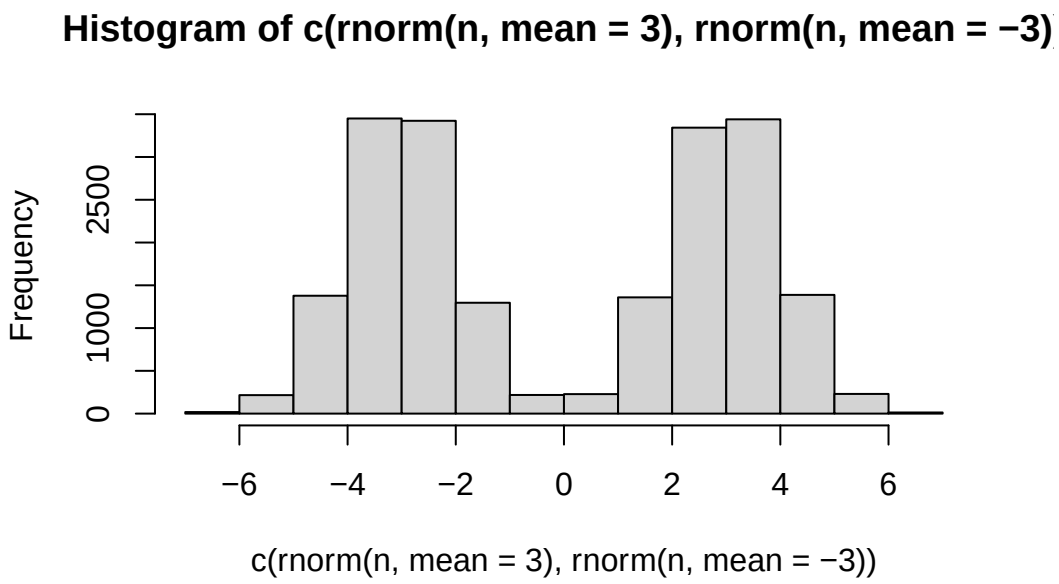
We can make a histogram of the random numbers generated with rnorm. Can give this two means by making the mean a vector containing 3 and -3.

```
hist(rnorm(150000, mean=c(3,-3)))
```



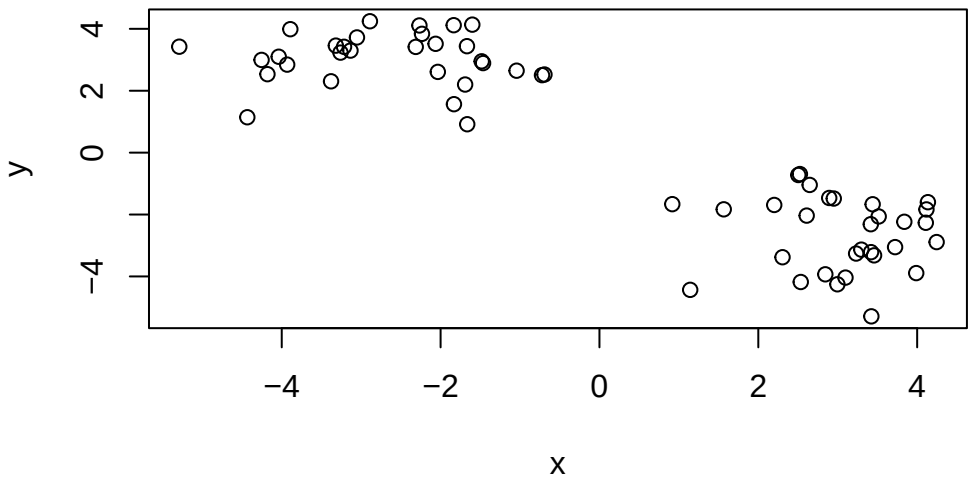
Another method of making a histogram with two centers

```r
n=10000

hist(c(rnorm(n, mean=3), rnorm(n,mean=-3)))
```



We can also take the reverse of our rnorm values to generate the scatter plot Barry drew on the board, with clusters centered on (3,-3) and (-3,3).

```r
n=30
x <- c(rnorm(n,mean=3),rnorm(n,mean=-3))
y <- rev(x)

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



K-means clustering.

The function in base R for k-means clustering is called `kmeans()`.

```
km <- kmeans(z, 2)
km
```

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

Cluster means:
          x         y
1 -2.629246  3.036185
2  3.036185 -2.629246

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] 62.42781 62.42781
 (between_SS / total_SS =  88.5 %)
```

```
Available components:

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

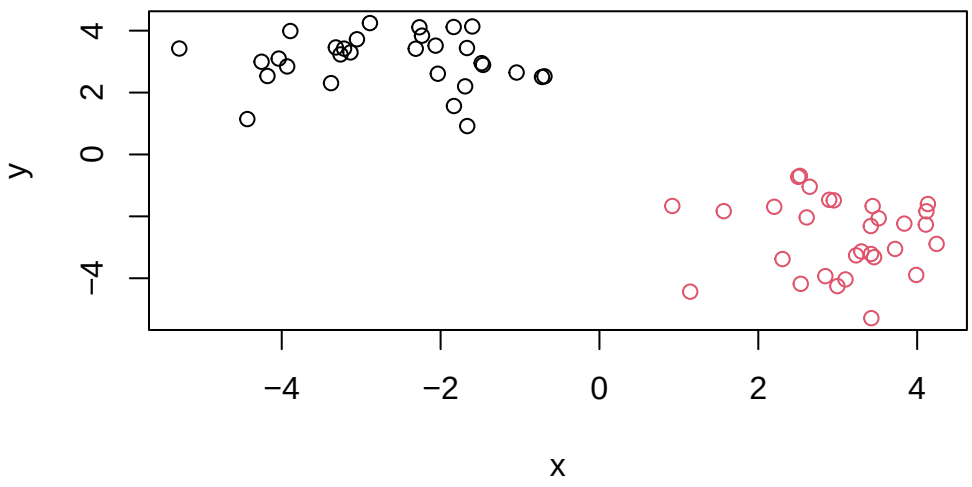
Q. Print out the cluster membership vectors.

```
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
```

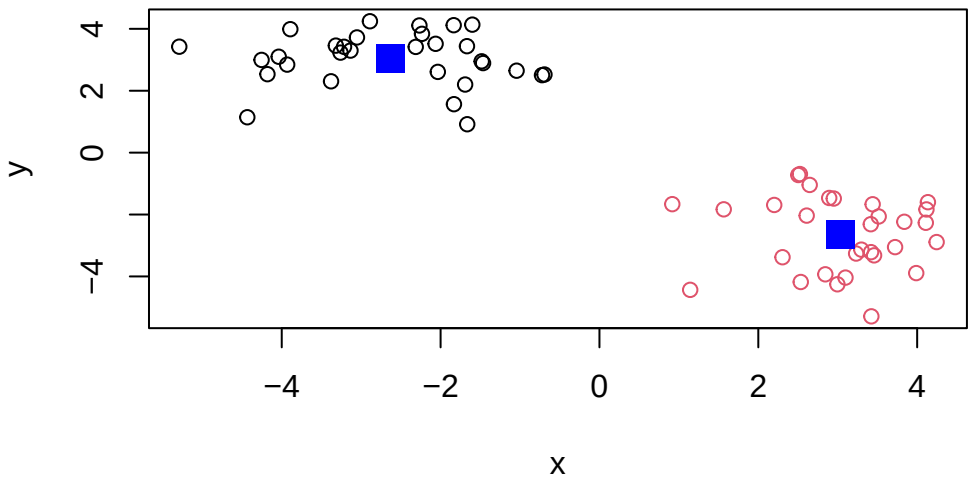
plot the data points, coloring each point by the cluster it belongs to

```
plot(z, col=km$cluster)
```



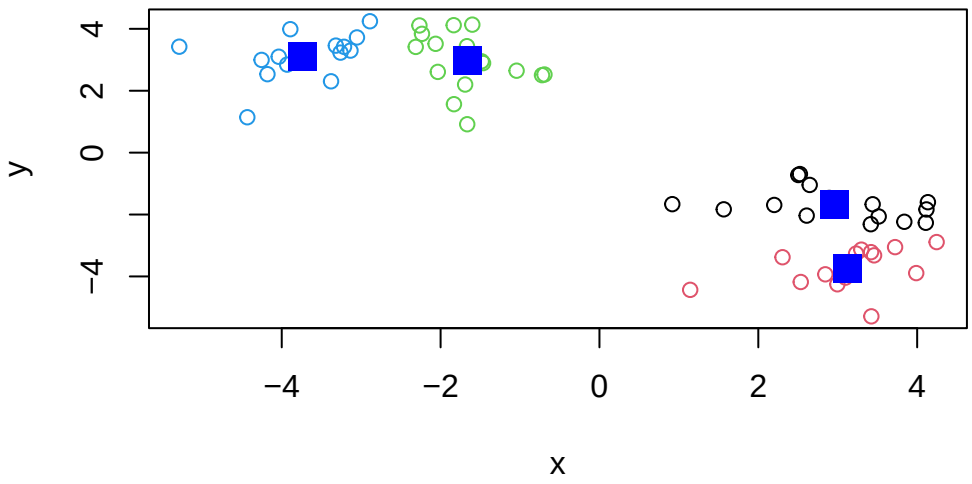
Plot with clustering result and add cluster centers.

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



Q. Can you cluster our data in z into 4 clusters?

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



The issue with this approach is that you have to specify how many centers your data has. You can generate a scree plot to make this kmeans strategy a bit more reliable.

Hierarchical clustering. Can be bottom-up or bottom-down. We will do bottom-up. Group and merge them in a step-by-step fashion.

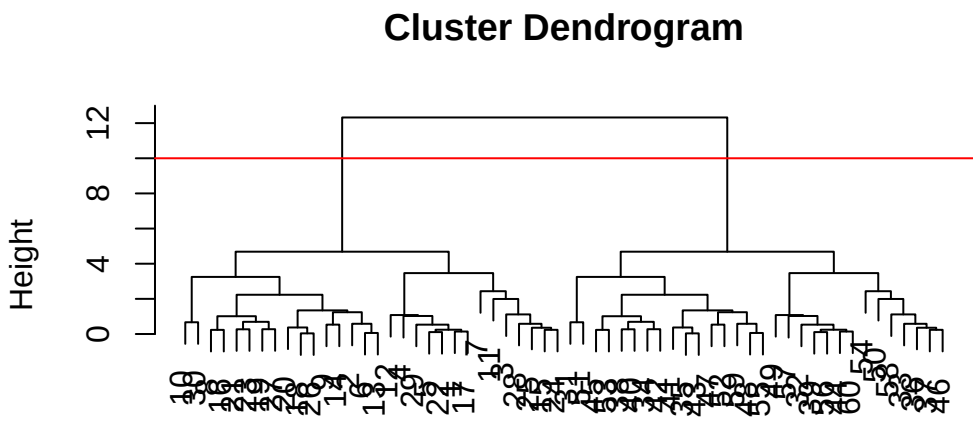
The main function for hierarchical clustering in base R is called `hclust()`.

Unlike `kmeans()` I can not just pass in my data as input. I first need a distance matrix.

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

```
Call:
hclust(d = d)

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

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



To get my clustering result (i.e. the membership vector), I can "cut" my tree at a given height. To do this, I will use the `cutree()` function.

```
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 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
```

Principal Component Analysis

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

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) # this tells you both rows and columns, but you can use nrow() or ncol() to count ther
```

```
[1] 17  5
```

```
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
```

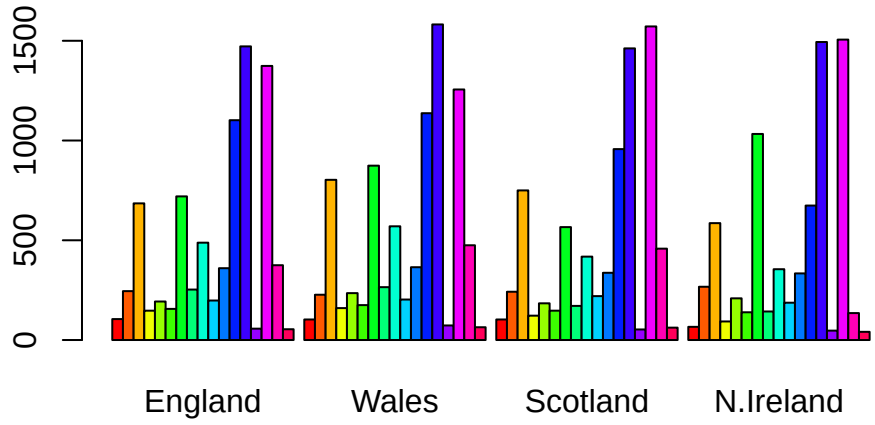
Change the first column into row names

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

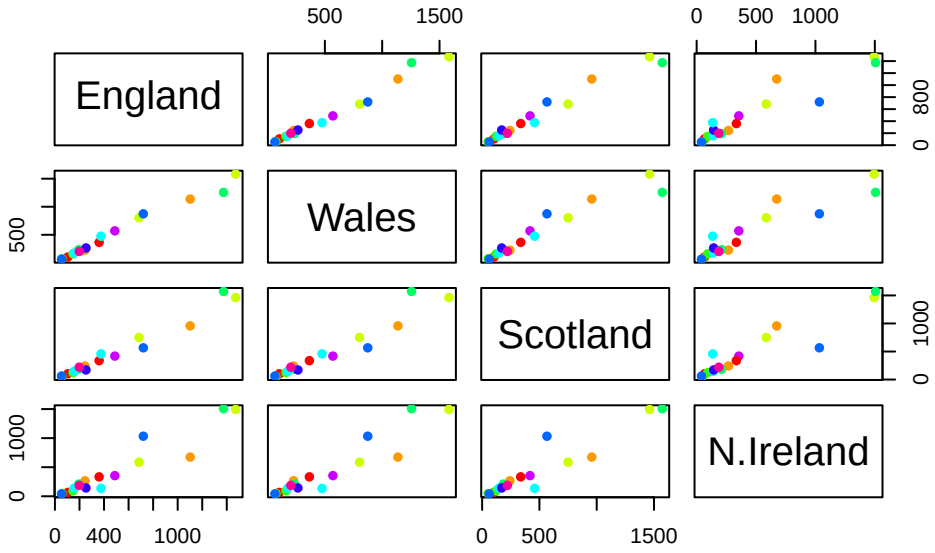
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars        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

Barplots won't be super helpful either. Can

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




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



PCA to the rescue

The main function to do PCA in base R is `prcomp()`.

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

Let's see what is inside our result object `pca` that we just calculated:

```
attributes(pca)
```

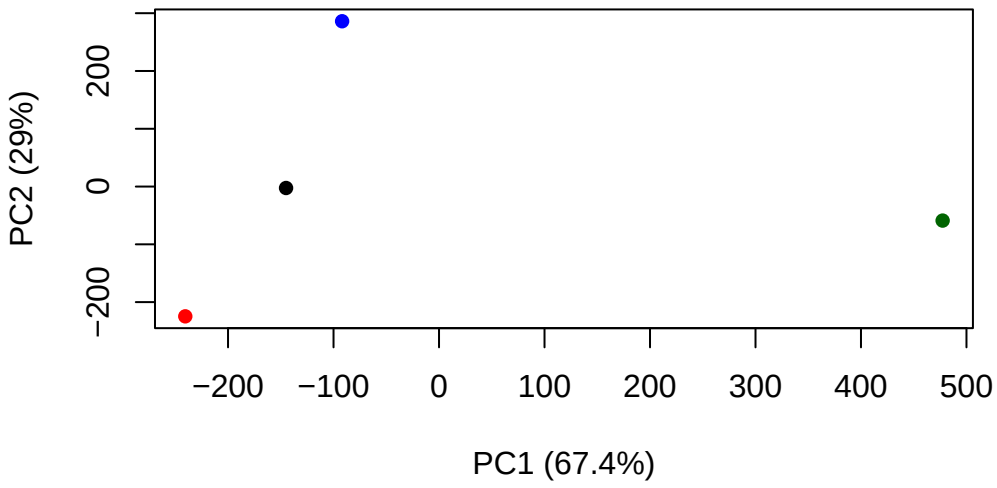
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
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
```

To make our main result figure, called a "PC plot" (or "score plot", "ordination plot" or "PC1 vs PC2 plot")

```
plot(pca$x[,1], pca$x[,2], col=c("black", "red", "blue", "darkgreen"), pch=16, xlab="PC1 (67
```



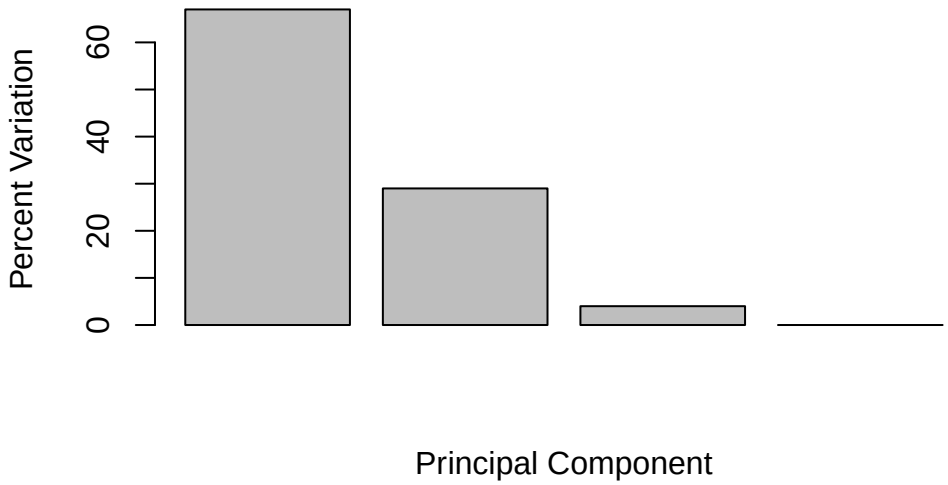

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

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

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 3.175833e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

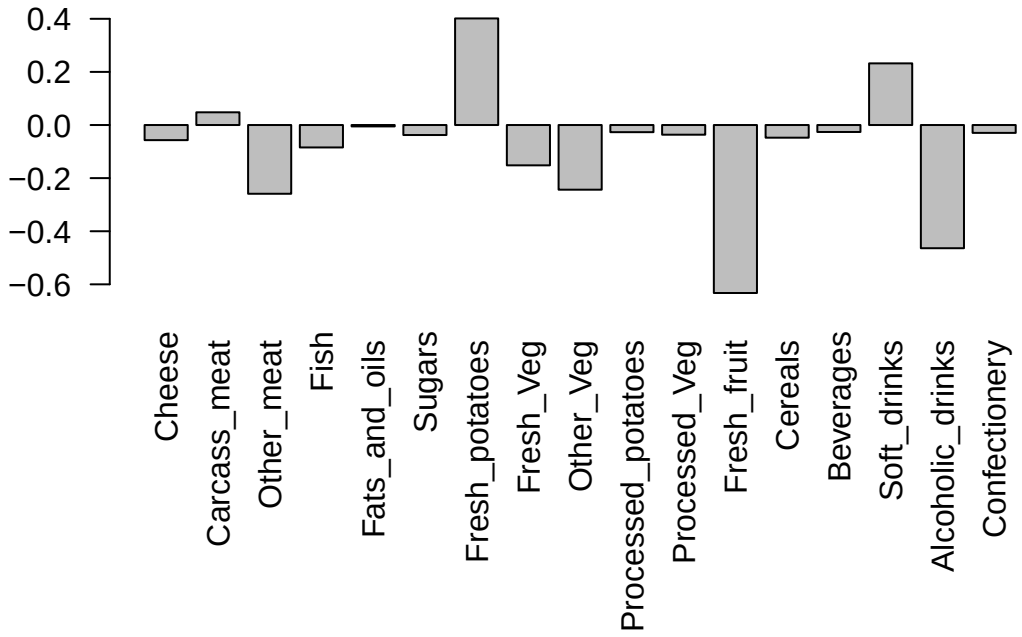
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Variable Loadings plot

Can give us insight on how the original variables (in this case the foods) contribute to our new PC axis

```
## 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 )
```


0.4
0.2
0.0
−0.2
−0.4
−0.6
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