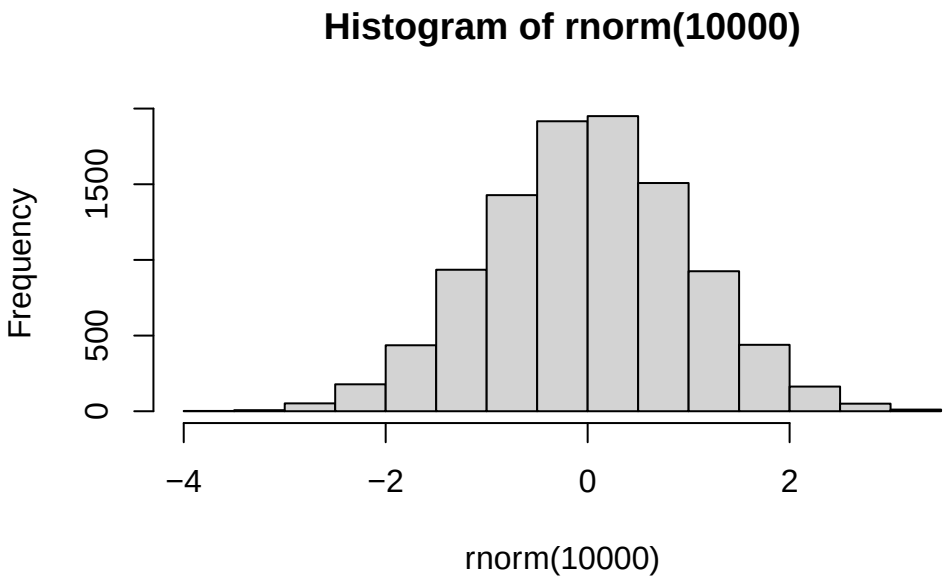
Machine Learning 1

Victor Yu

#First up kmeans ()

Demo ad using kmeans () function in base R. First make up some data with a known structure.

```
#normal distribution of 10000 observations
#centered around 0
hist (rnorm (10000))
```



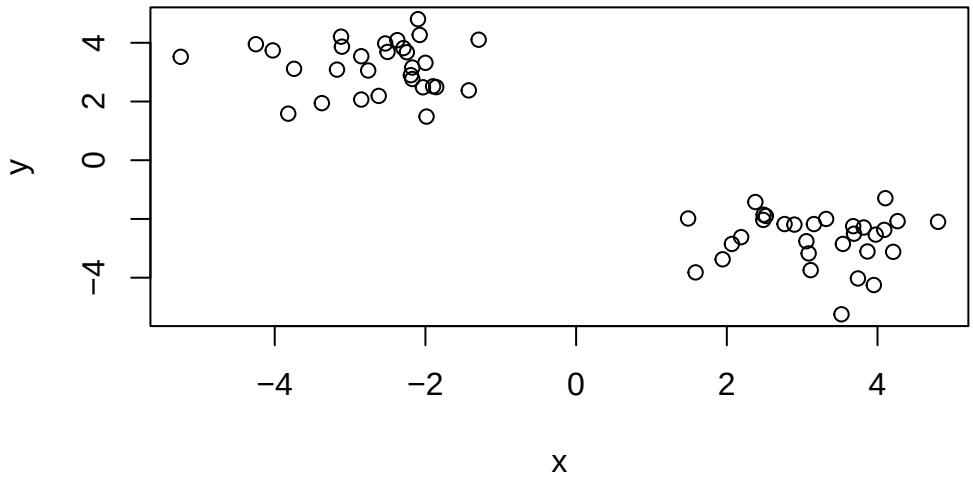

```
#30 observations within +- 3
tmp <- c (rnorm(30, -3), rnorm (30,3) )
```

```
#x <- cbind(tmp,rev(temp))
#rev(tmp) puts +3 first, then -3

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

```
            x        y
[1,] -1.999852 3.317971
[2,] -2.098565 4.803797
[3,] -1.857157 2.489039
[4,] -3.173479 3.086377
[5,] -1.424451 2.378902
[6,] -3.743367 3.113172
```

```
plot(x)
```



Now we have some made up data in **x** let's see how kmeans work with this data

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

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

Cluster means:
          x         y
1 -2.669993  3.192876
2  3.192876 -2.669993

Clustering vector:
 [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

Within cluster sum of squares by cluster:
[1] 43.77802 43.77802
 (between_SS / total_SS =  92.2 %)

Available components:

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

Q. How many points are in each cluster

```
k$size
```

```
[1] 30 30
```

Q. How Do we get to the cluster membership/assignment.

```
k$cluster
```

```
 [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
```

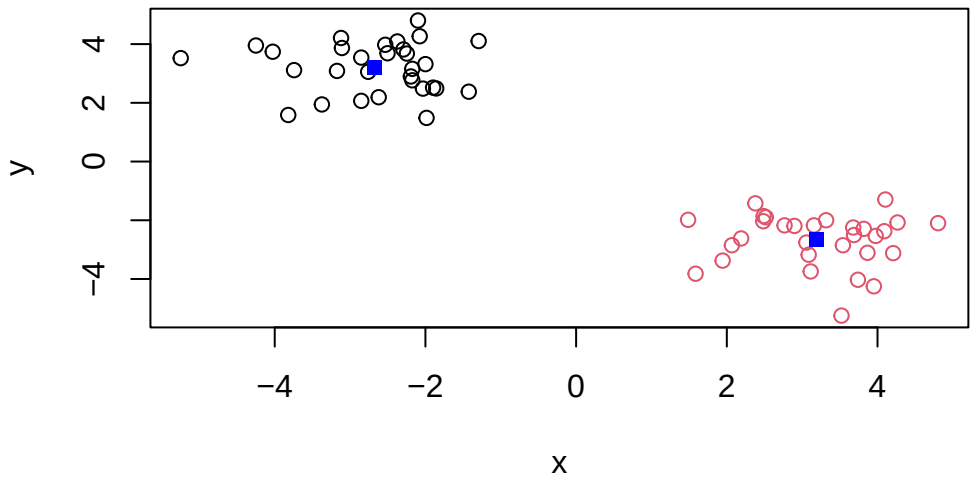
Q. What about cluster centers.

```
k$centers
```

```
          x         y
1 -2.669993  3.192876
2  3.192876 -2.669993
```

No w we got to the main results let's use them to plot our data with the kmeans result

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



Now for Hierarchical Clustering

We will cluster the same data `x` with the `hclust()`. In this case `hclust()` requires a distance matrix as input

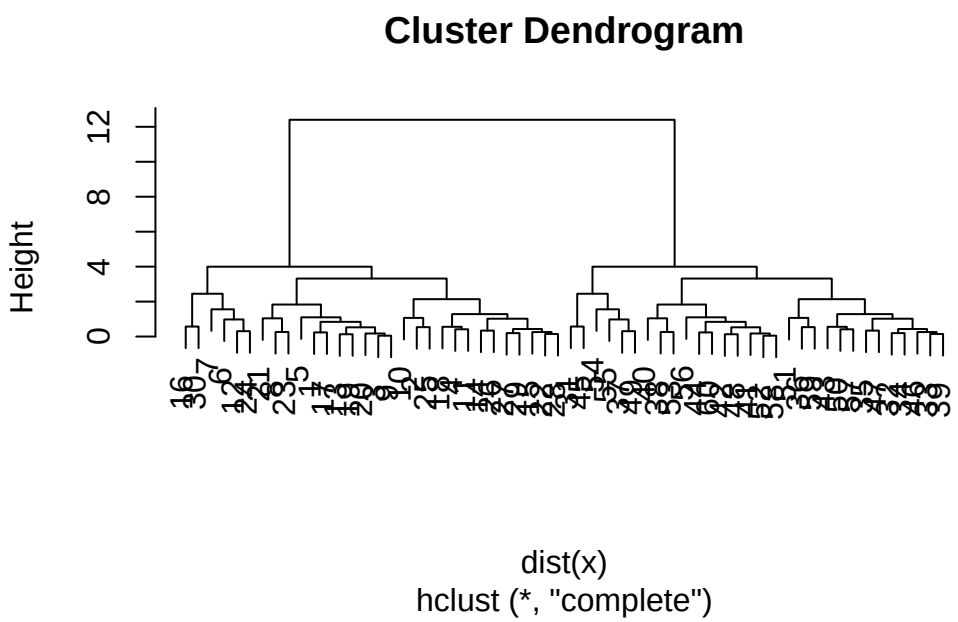
```
d <-  dist(x)
hc <- hclust( dist(x) )
hc
```

```
Call:
hclust(d = dist(x))

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

Let's plot our hclust result

```
plot (hc)
```



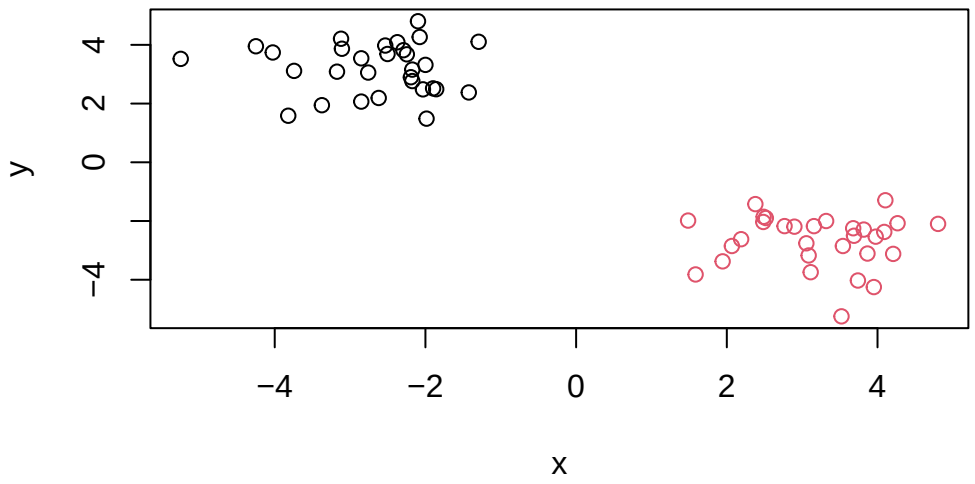
To get our cluster memberhsip vector we need to "cut" the tree with the `cutree()`

```
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
```

Now plot our data with the hclust() results.

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




```
#K = argument to cutree is a lot more helpful than the H, height
cutree(hc, k=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
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

#Principle Componenet Analysis (PCA)

PCA of UK food data

Read data from website and try a few visualizations.

```
url <- "https://tinyurl.com/UK-foods"
UK <- 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?

```
#setup / checking dimensiono of the data set
dim(UK)
```

```
[1] 17  5
```

```
head(UK)
```

```
              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
```

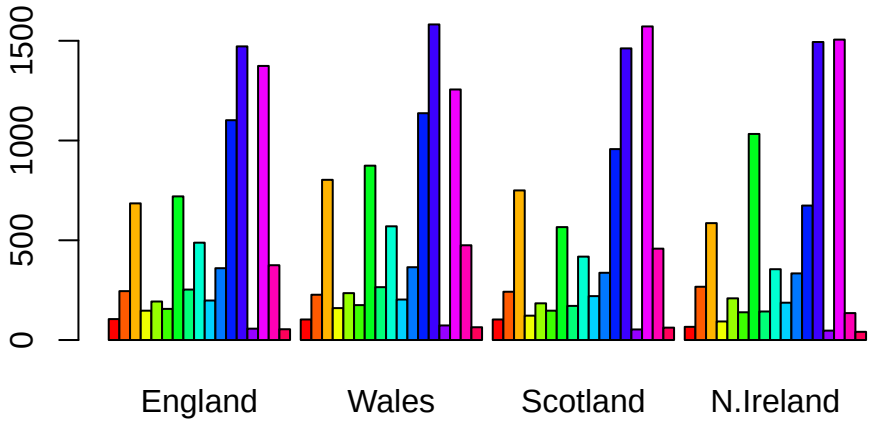
```
rownames (UK) = UK[,1]
UK <- UK[,-1]
dim (UK)
```

```
[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?

Answer: I prefer the second method since it is uch more concise. All i have to do i to remember to add row.names = 1 then i save so much time

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



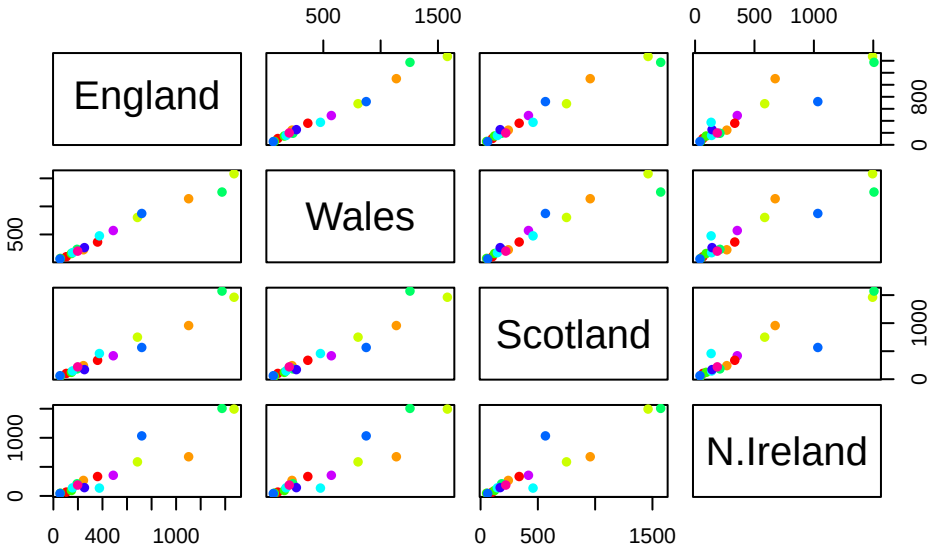
Q3: Changing what optional argument in the above barplot() function results in the following plot?

Answer: If you remove the beside optionor change it to false, the bar plot type changes.

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 colors in the figure correspond with a different food type. Each plot has a different corresponding 2 contries which makes up its X and Y axis.

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



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

N. Ireland diagnonal lines are not as a linear in comparison to the other countries. The dots are more spread out indicating increased variance.

PCA to the rescue!! The main base R PCA function is called `prcomp()`

```
#t(x) is the tranpose of the data. Test t(x) when you can't visuallize it yourself.
pca <- prcomp( t(UK) )
```

Summary of PCA data, on what it is doing

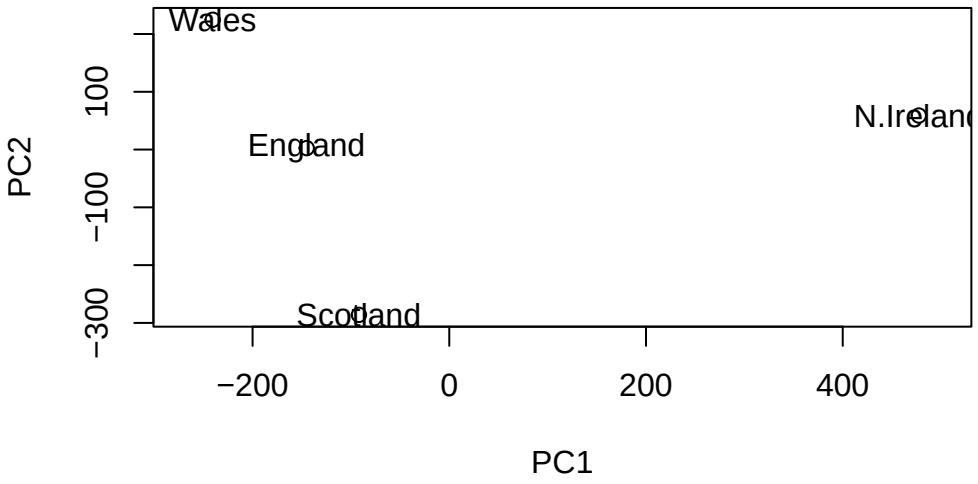
```
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3        PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

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

To make our new PCA plt (a.k.a PCA score plot) we access `pca$x`

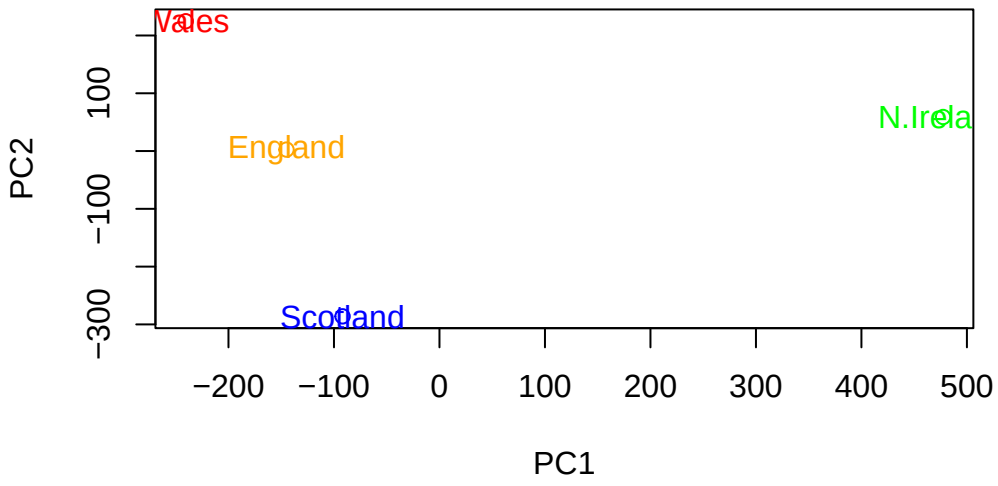
```
#xlim sets the limits in the graph. Make it more aesthetic
plot(pca$x [,1], pca$x[,2], xlab = "PC1", ylab = "PC2", xlim = c(-270, 500))
text(pca$x [,1], pca$x[,2], colnames(UK))
```



Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

Adding Color!

```
#c is vector of colors. Keep in mind
country_cols <- c("orange", "red", "blue", "green")
plot(pca$x [,1], pca$x[,2], xlab = "PC1", ylab="PC2", col = country_cols)
text(pca$x [,1], pca$x[,2], colnames(UK),
col = country_cols)
```



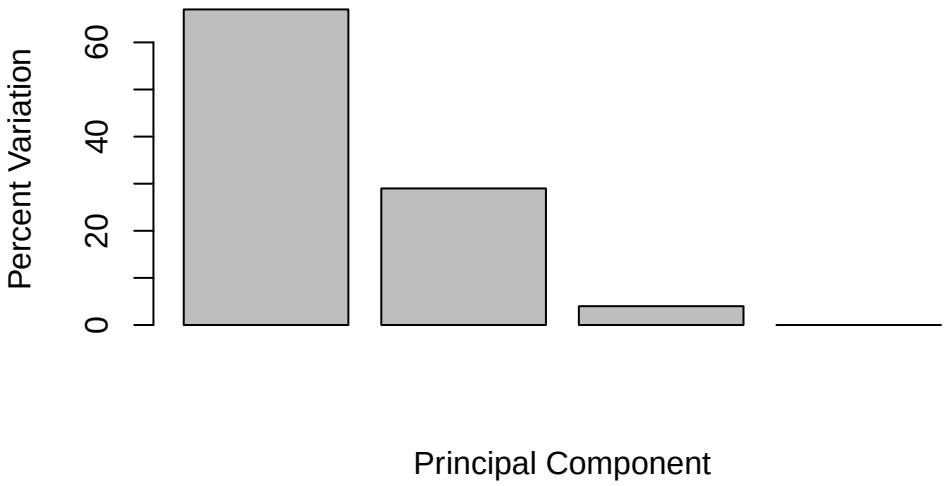

```
#variance colculation (must square standard dev. sdev^2)
v <- round( pca$sdev^2 / sum(pca$sdev^2) * 100)
v
```

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

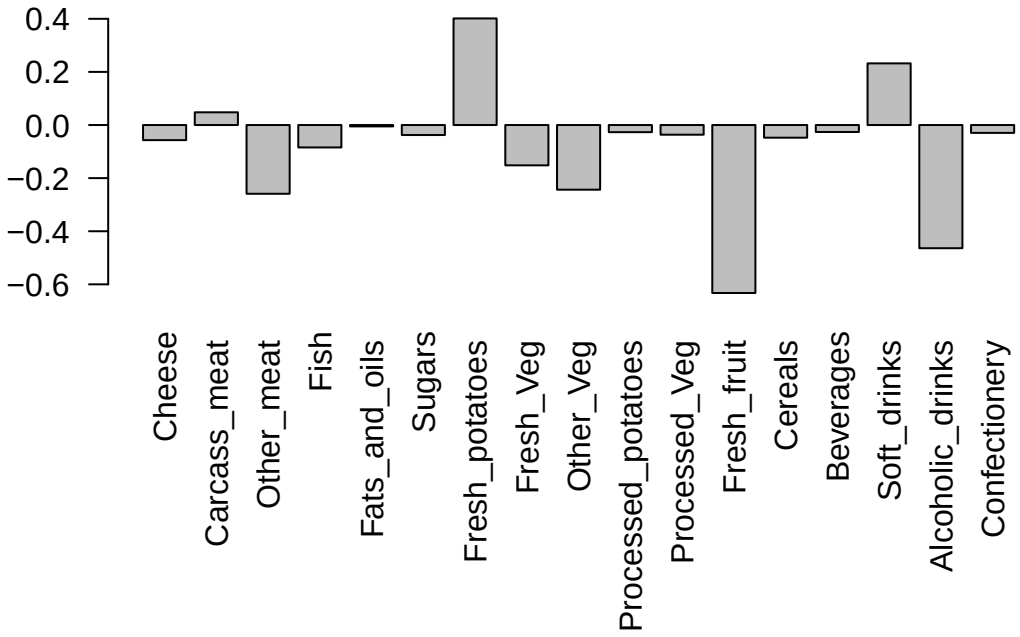
```
#Alternative, prints out sdev and variance and cumulation
x <- summary (pca)
x$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

```
#Bar plot calculating for variance
barplot(v, xlab= "Principal Component", ylab = "Percent Variation")
```



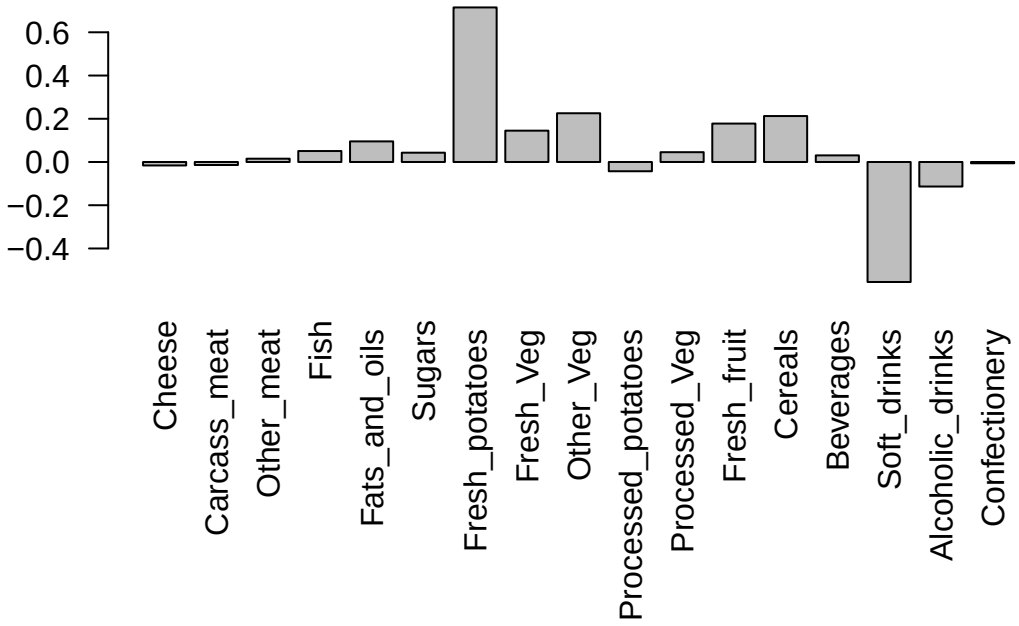

```
#PC1 looking > 90% variance
par(mar=c(10, 3, 0.35, 0))
barplot (pca$rotation[,1], las=2)
```



Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

The two prominent food groups are fresh potatoes and soft drinks. The PC2 tells us variance of the y-axis.

```
#PC2 looking > 90% variance
#Same code used in PC2 function
par(mar=c(10, 3, 0.35, 0))
barplot (pca$rotation[,2], las=2)
```




```
#Big Plot Function
biplot(pca)
```