Machine Learning (Lab 7)

Sofia Lanaspa, PID:A17105313

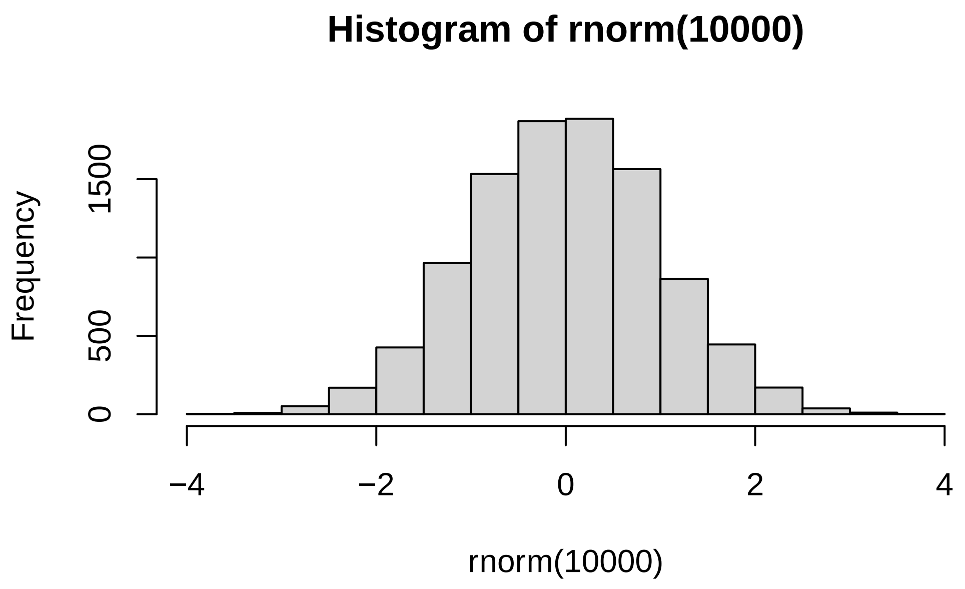
Today we are going to learn how to apply different machine learning methods, beginning with clustering

Goal: Find groups/clusters in input data

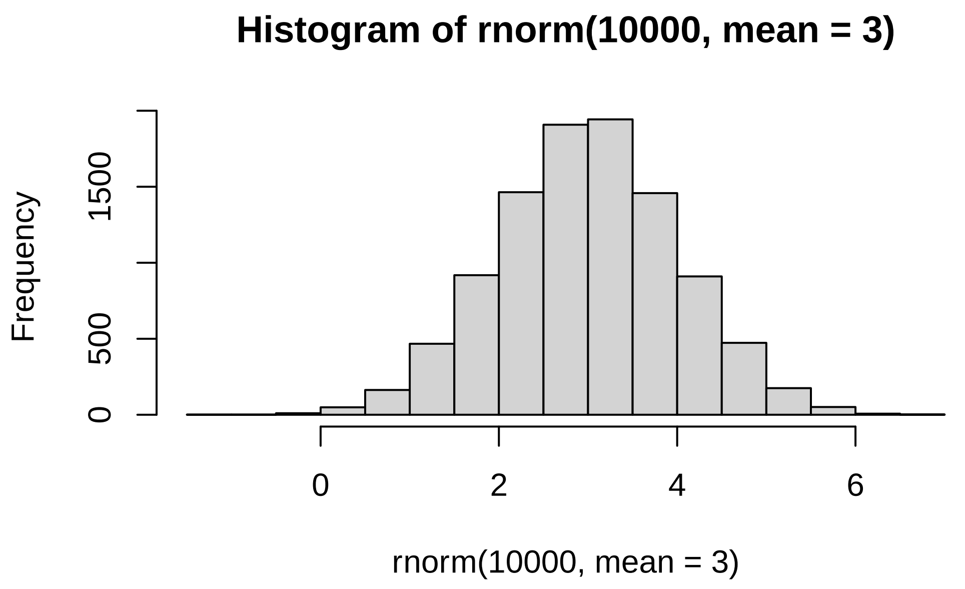
```
#help page for rnorm
?rnorm
#get 10 numbers (other arguments are default)
rnorm(10)
```

```
 [1]  0.13481506 -1.75298008  2.30383768  0.42239775 -0.02129222  0.91925772
 [7]  1.43019042 -0.03628594 -0.36832439  0.73908320
```

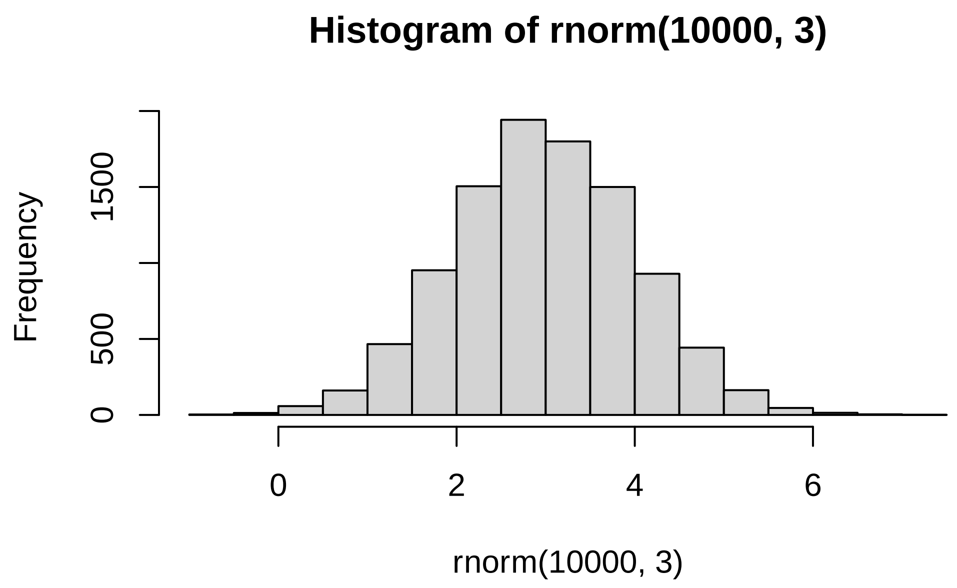
```
#histogram with center at 0 (default, mean=0, sd=1)
hist(rnorm(10000))
```



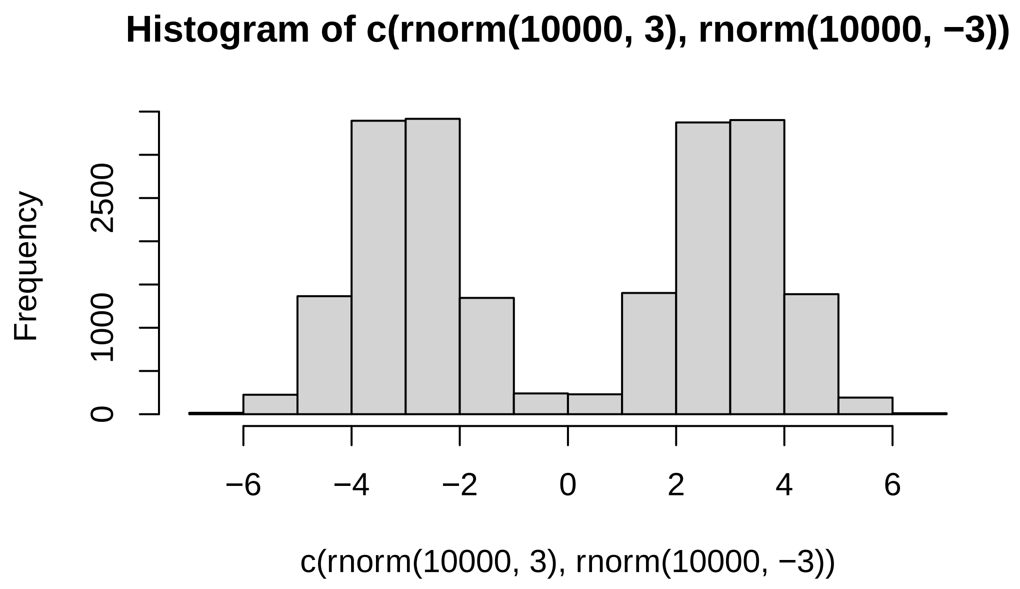
```
#histogram with center at 3 (mean=3)
hist(rnorm(10000,mean = 3))
```



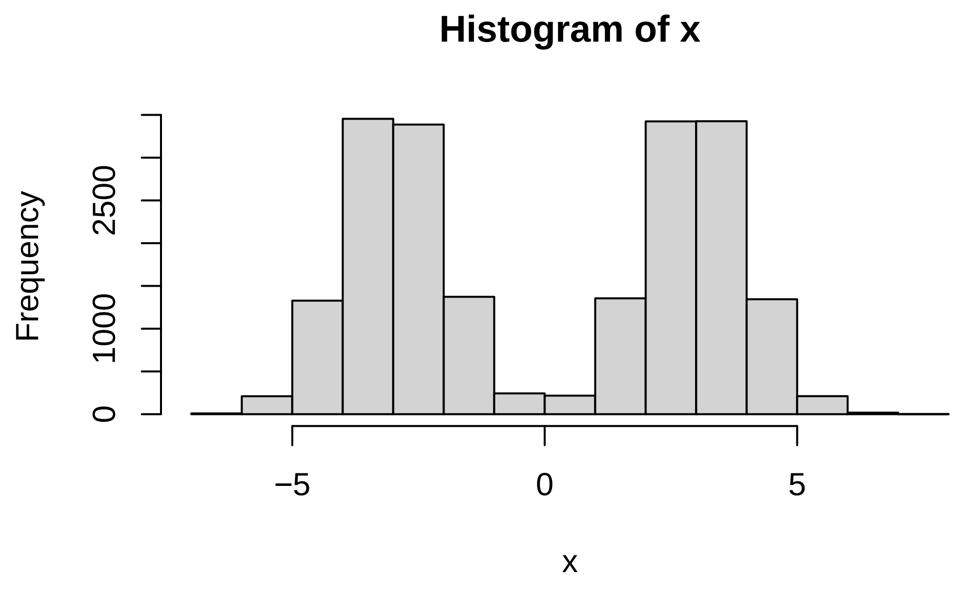
```
hist(rnorm(10000,3))
```



```
#histogram with 2 peaks
hist(c(rnorm(10000,3),rnorm(10000,-3)))
```



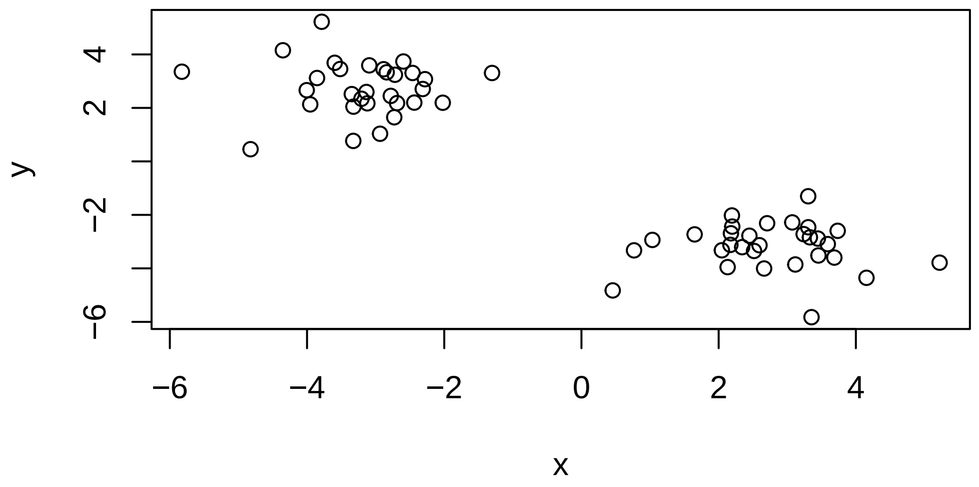
```r
#more clear code
n <- 10000
x <- c(rnorm(n,3),rnorm(n,-3))
hist(x)
```



```r
n <- 30
x <- c(rnorm(n,-3),rnorm(n,3))
y <- rev(x)
z <- cbind(x,y)
head(z)
```

```
             x         y
[1,] -3.596927 3.6860745
[2,] -4.004860 2.6624147
[3,] -4.823664 0.4555457
[4,] -3.517540 3.4540608
[5,] -3.206459 2.3422874
[6,] -2.313235 2.7063135
```

```r
plot(z)
```



kmeans function

kmeans assigns cluster based on how far a point is from each mean

Q. How many points are in each element?

```
#clusters of sizes 30, 30 (because n=30), so 30 points in each element
km <- kmeans(z,centers=2)
km
```

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

Cluster means:
          x         y
1 -3.175094  2.736827
2  2.736827 -3.175094

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 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

Within cluster sum of squares by cluster:
```

```
[1] 51.71048 51.71048
 (between_SS / total_SS =  91.0 %)

Available components:

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

Results in kmeans object 'km'

Q. What 'component' of your results object details: cluster size? cluster assignment/member? cluster center?

```r
attributes(km)
```

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

$class
[1] "kmeans"
```

```r
#size
km$size
```

```
[1] 30 30
```

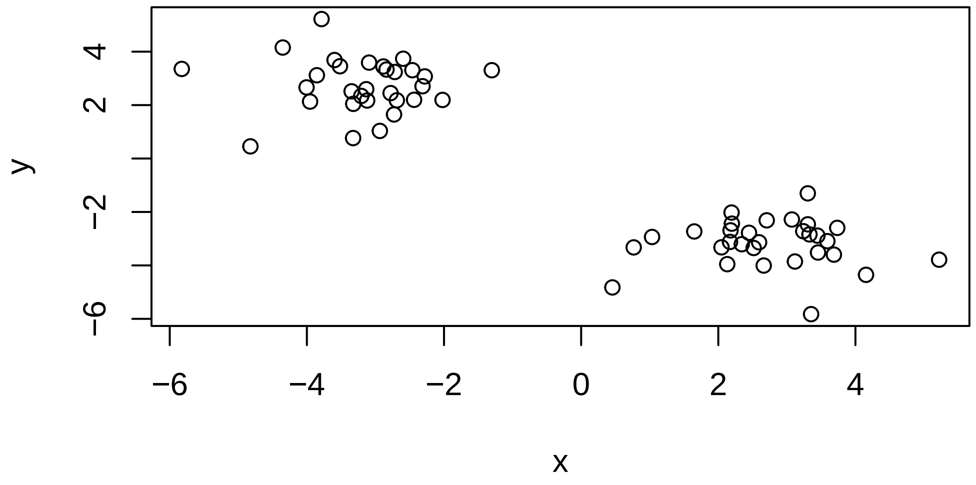
```r
#assignment/member
km$
#center
km$center
```

```
NULL
```

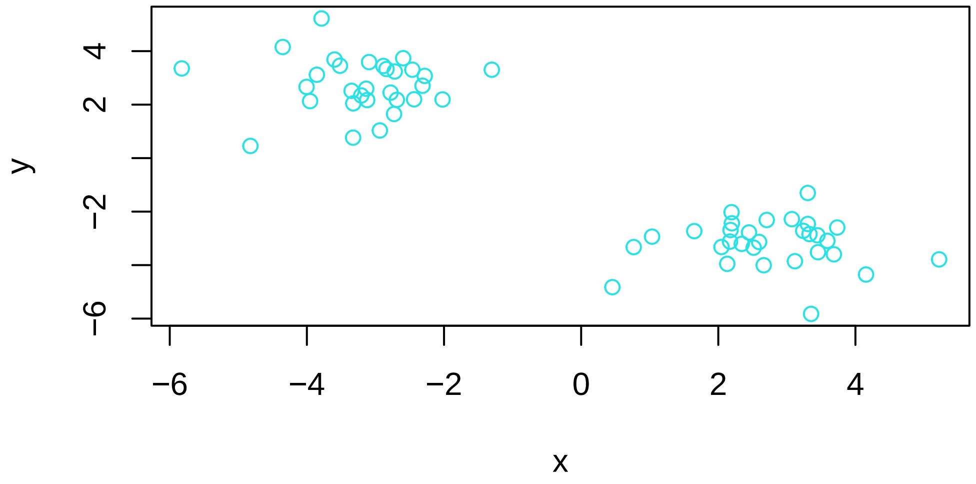
Q. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

**R will recycle shorter color vector to be the same length as the longger (num of data points) in z

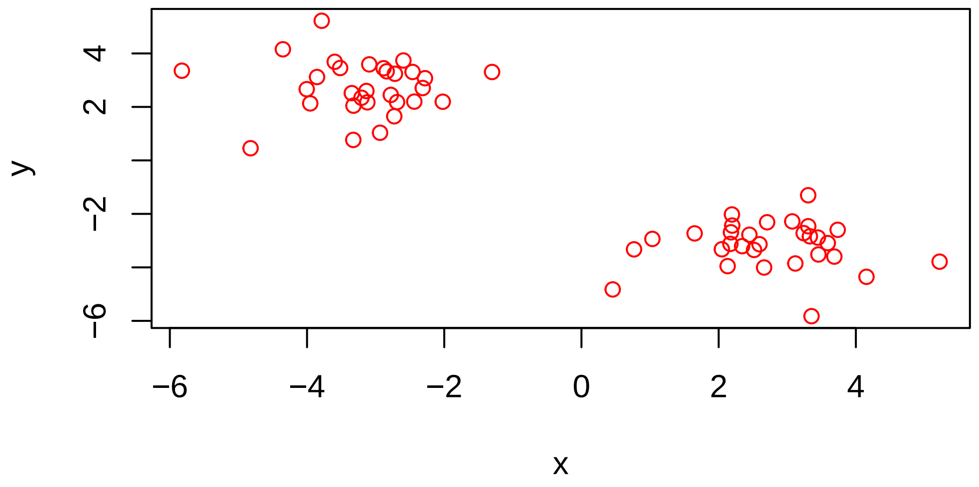
```r
plot(z,col=1) #numbers are diff. colors
```



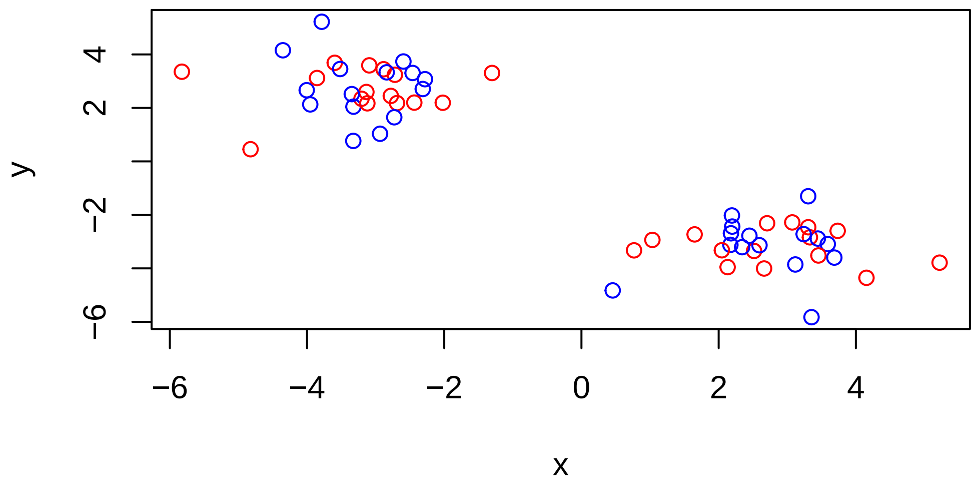
```r
plot(z,col=5)
```



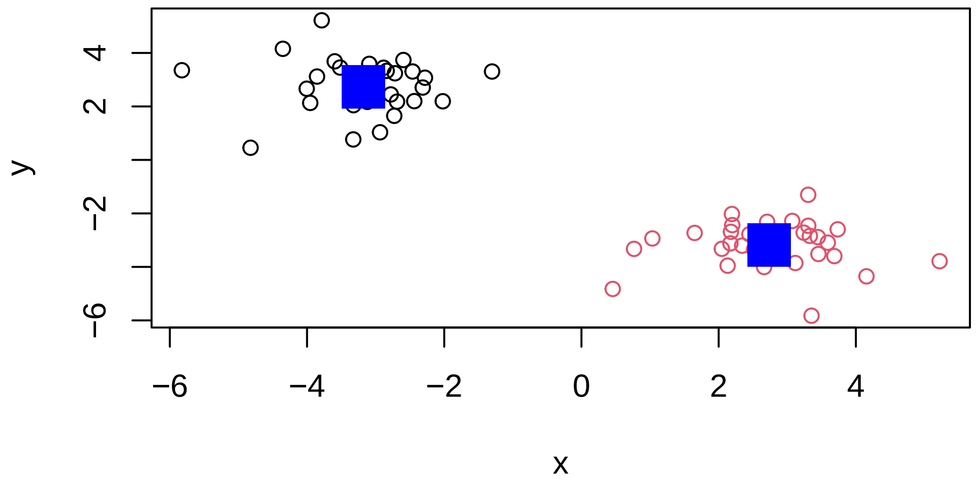
```r
plot(z, col="red")
```



```r
#function below alternates blue and red points
plot(z,col=c("red","blue"))
```



```
plot(z,col=km$cluster) #use clustering by km to assign color per cluster
points(km$centers, col="blue", pch = 15, cex=3) #make points in center blue, 15 makes the sha
```

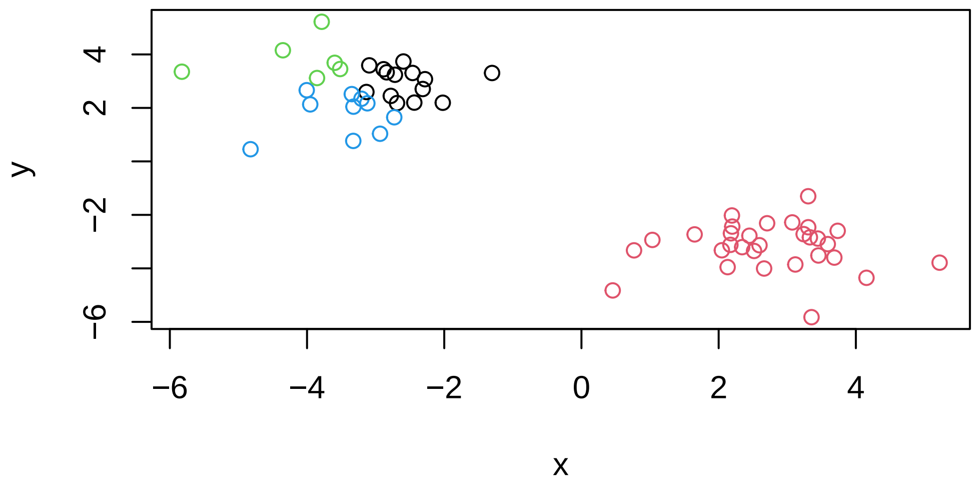


Q. Run kmeans and ask for 4 clusters, plot results as above

```r
km4 <- kmeans(z,centers=4)
km4
```

```
K-means clustering with 4 clusters of sizes 14, 30, 6, 10

Cluster means:
          x         y
1 -2.539331  2.953137
2  2.736827 -3.175094
3 -4.155264  3.830986
4 -3.477059  1.777499

Clustering vector:
 [1] 3 4 4 3 4 1 1 1 1 1 1 4 3 4 4 1 1 3 1 4 1 4 3 1 1 4 1 4 1 3 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

Within cluster sum of squares by cluster:
[1]  6.853319 51.710481  6.706845  8.774106
 (between_SS / total_SS =  93.6 %)

Available components:

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

```r
plot(z,col=km4$cluster)
```



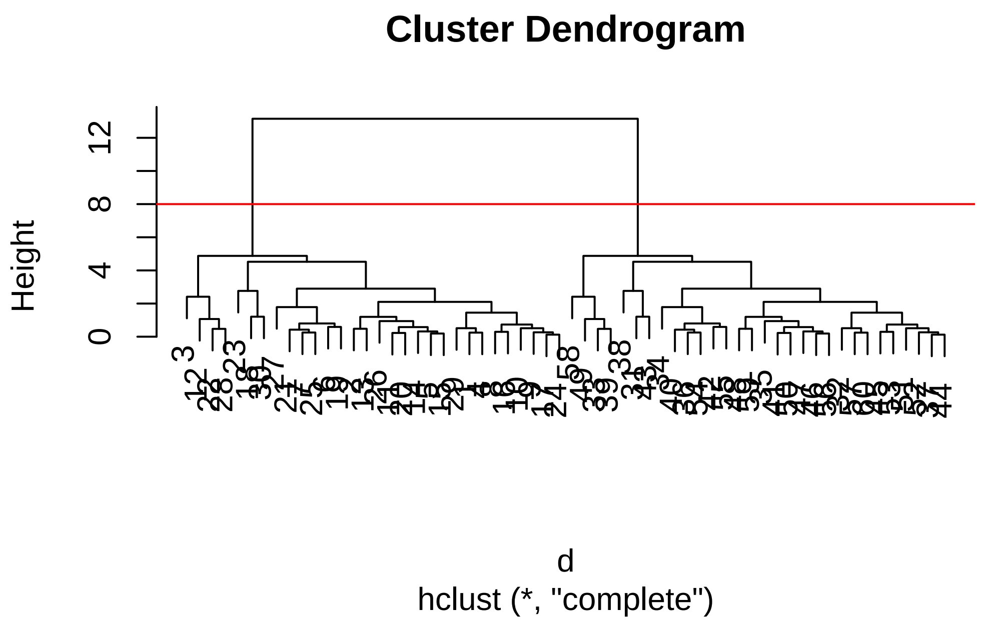
```
#ISSUE!!! kmeans makes as many clusters as we tell it, no reasoning for how many we need
#¡¡¡IT CHANGES THE CLUSTERS EACH TIME!!! (want to do what we tell it but badly)
```

Hierarchical Clustering

Lets take our same data 'z' and see how hclust() works

First we need to find distance between rows of our data matriz 'z'

```
?hclust #Hierarchical cluster analysis on a set of dissimilarities and methods for analyzing
d <- dist(z)
hc <- hclust(d)
plot(hc)
abline(h=8, col="red") #red line shows where clusters should be made
```



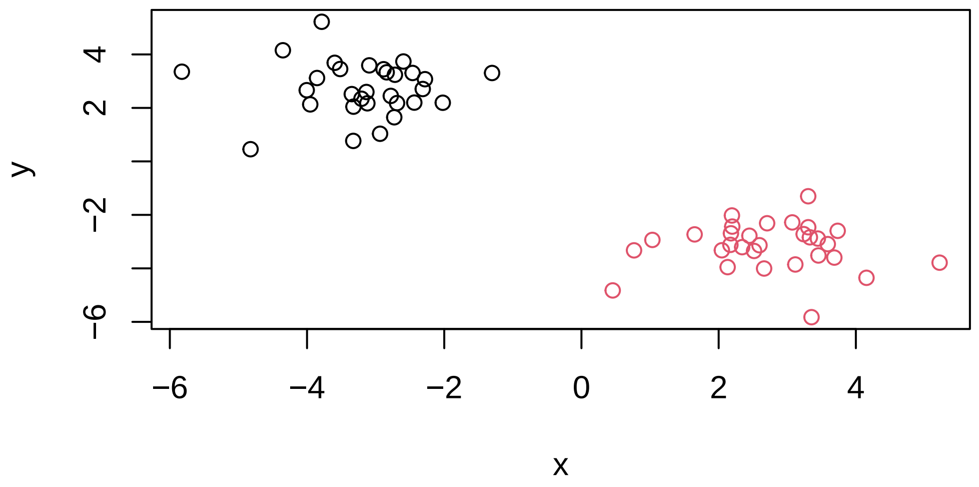
I can get my cluster membership vector by "cutting the tree" with the 'cutree()' function:

```
?cutree #cuts tree to give you groups of your choosing
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 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
```

Plot 'z' colored by our hclust results

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



PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal components are new low dimensional axis (or surfaces) closest to the observations

PCA tries to find axis of best fit to look at data in groups

PCA of UK food data

Q. 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?

Read data from the UK on food consumption in different parts of the UK

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
#rownames(x) <- x[,1]     other option for remoing numbers column (1st col)
#x <- x[,-1]
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
```

```r
x <- read.csv(url, row.names=1) #to remove column of only numbers, useless
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
```

Q. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

```r
ncol(x)
```

```
[1] 4
```

```r
nrow(x)
```

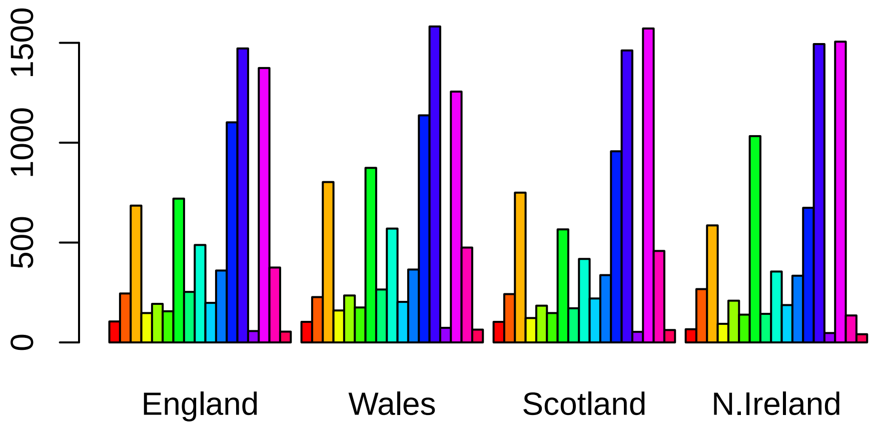
```
[1] 17
```

```r
dim(x)
```

```
[1] 17  4
```

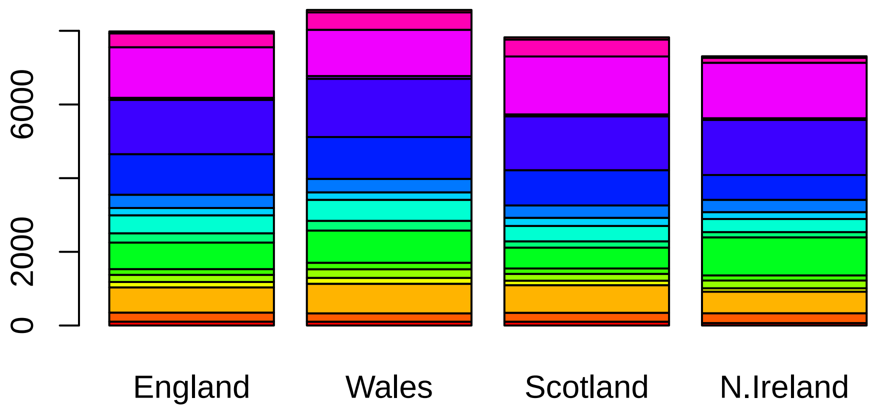
4 columns and 17 rows

```r
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?

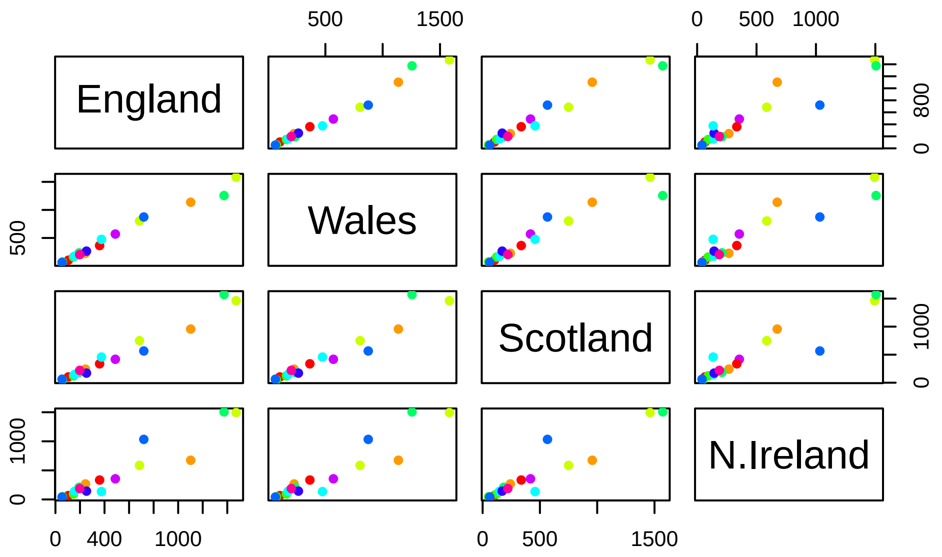
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



Differences in categories between countries are so unclear

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)
```



It is hard to see structure and trends in even this small data set, so how will we do this when we have ig data sets with thousands or tens of thousands of things we are measuring??

Lets see how PCA deals with this dataset, so main function in base R to do PCA is called 'prccomp()'

```
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
```

PC1 captures 67.44% of the data

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

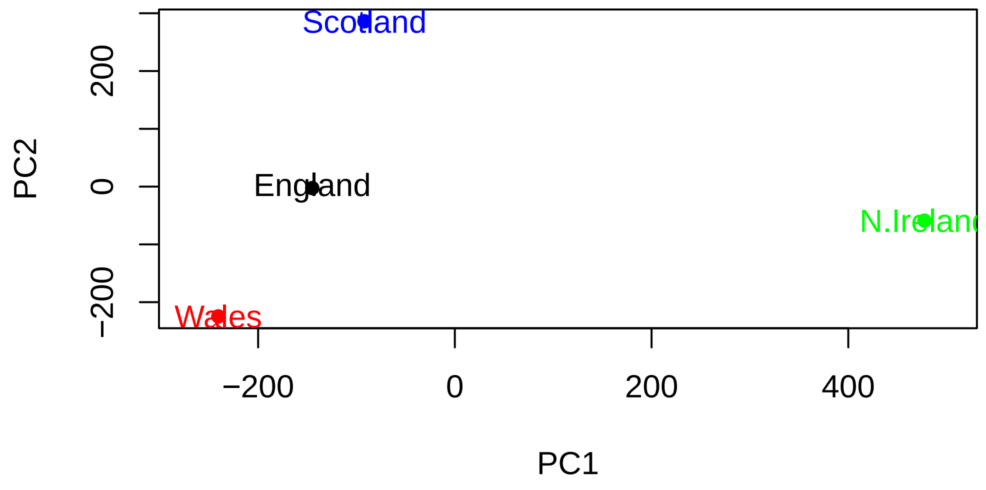
```
pca$x
```

```
                 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
```

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

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.

```
plot(pca$x[,1],pca$x[,2],col=c("black","red","blue","green"),pch=16,xlab="PC1", ylab="PC2", 
text(pca$x[,1], pca$x[,2], colnames(x),col=c("black","red","blue","green"))
```



Bar chart, shows variation in each component

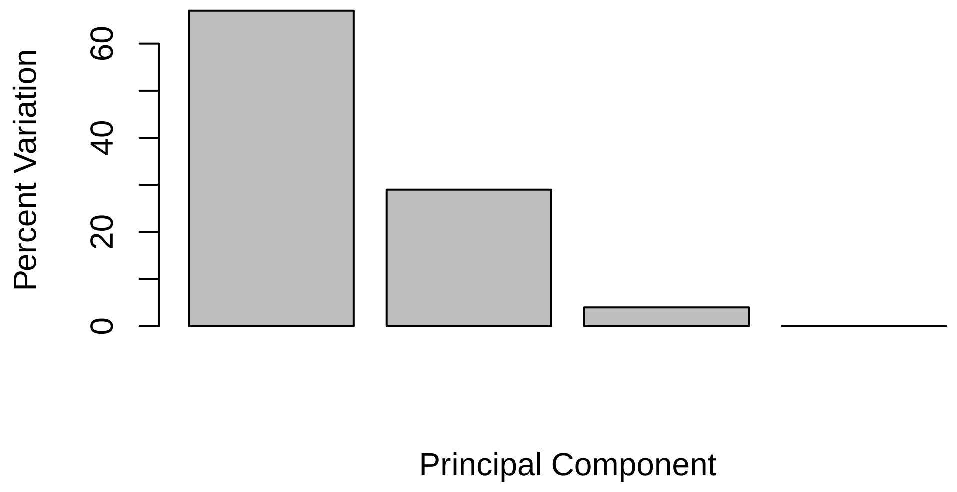
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

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

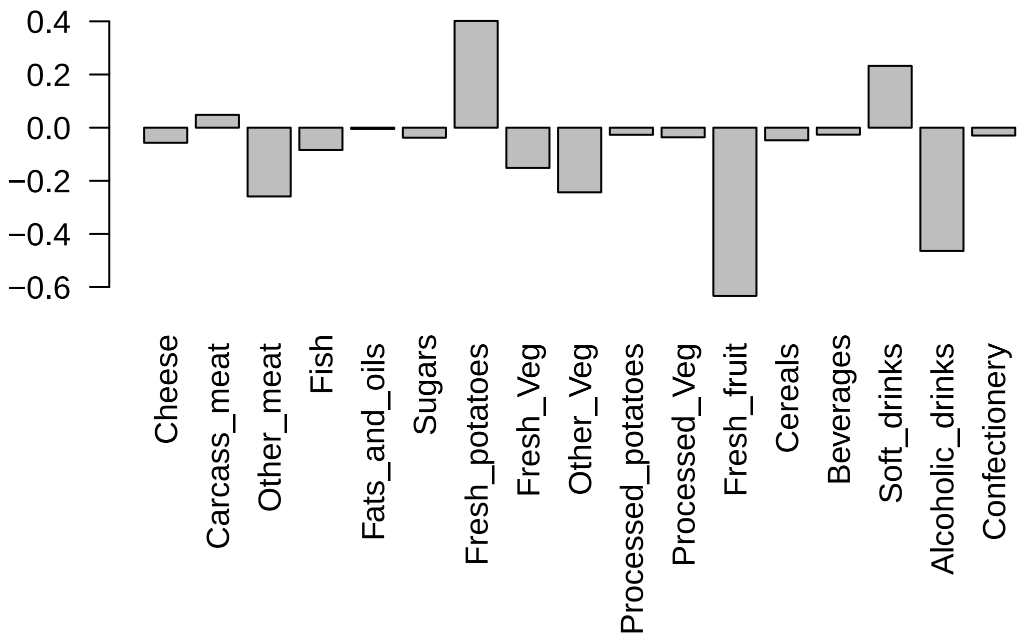
```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 2.921348e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

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



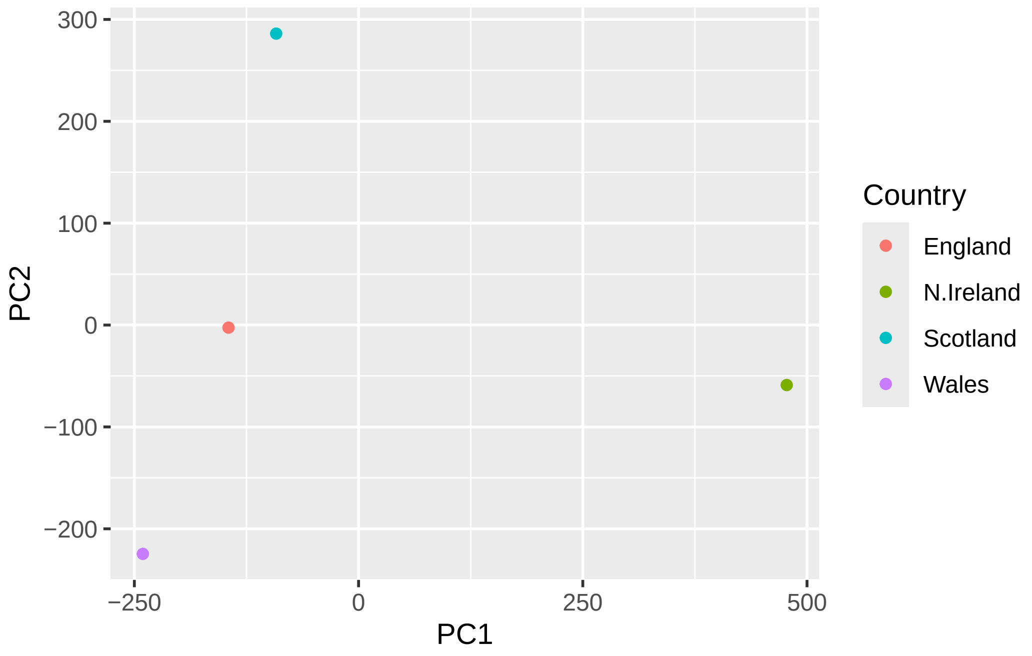
More useful bar graph to visualize data

```r
## 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 )
```

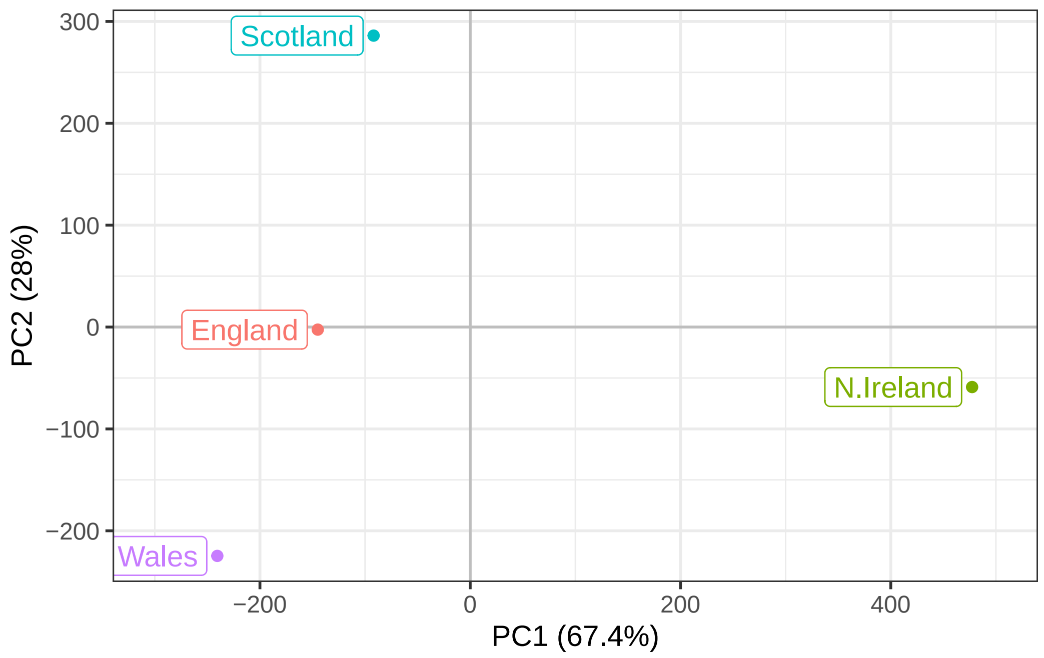


Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

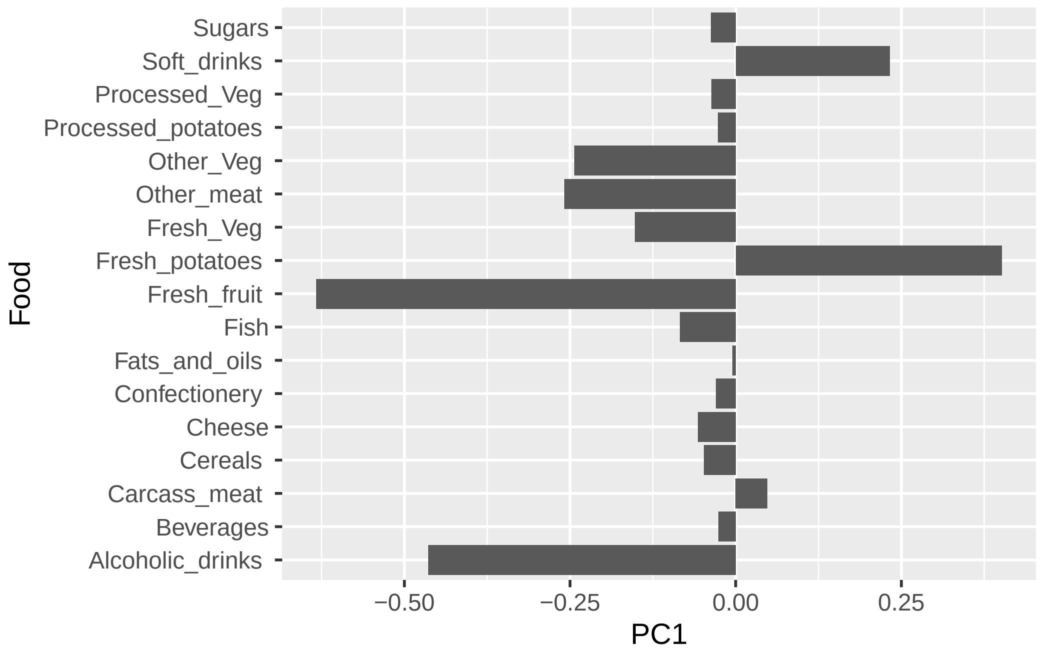
```
library(ggplot2)

df <- as.data.frame(pca$x)
df_lab <- tibble::rownames_to_column(df, "Country")

# Our first basic plot
ggplot(df_lab) +
  aes(PC1, PC2, col=Country) +
  geom_point()
```



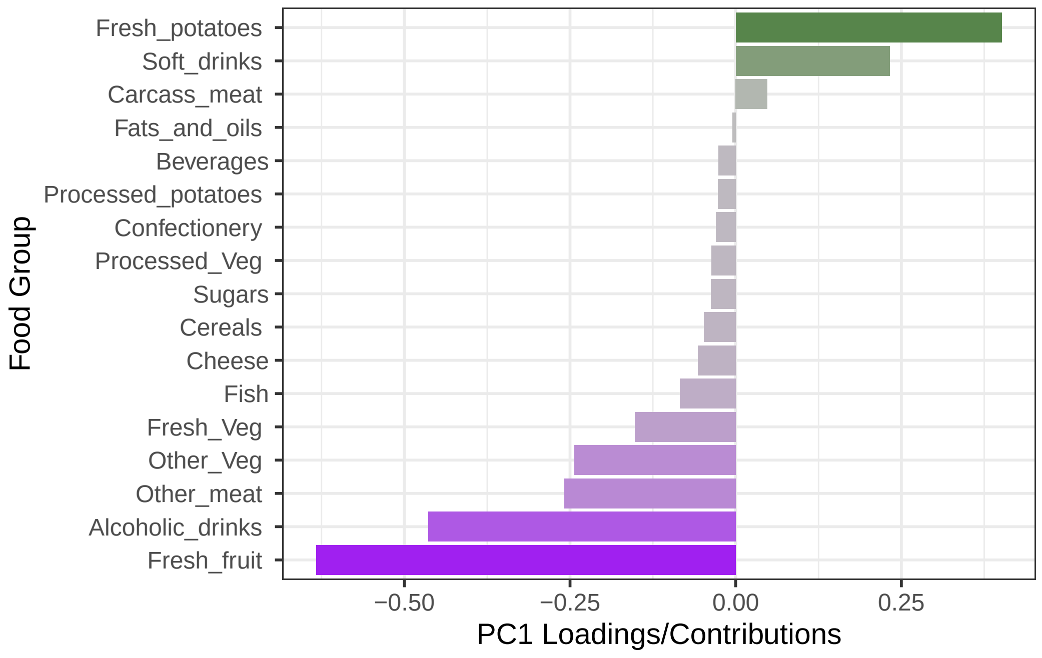
```r
ggplot(df_lab) +
  aes(PC1, PC2, col=Country, label=Country) +
  geom_hline(yintercept = 0, col="gray") +
  geom_vline(xintercept = 0, col="gray") +
  geom_point(show.legend = FALSE) +
  geom_label(hjust=1, nudge_x = -10, show.legend = FALSE) +
  expand_limits(x = c(-300,500)) +
  xlab("PC1 (67.4%)") +
  ylab("PC2 (28%)") +
  theme_bw()
```



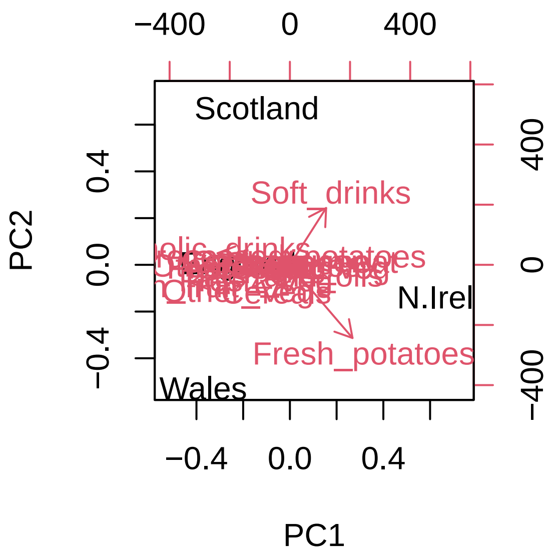
```r
ld <- as.data.frame(pca$rotation)
ld_lab <- tibble::rownames_to_column(ld, "Food")

ggplot(ld_lab) +
  aes(PC1, Food) +
  geom_col()
```



```r
ggplot(ld_lab) +
  aes(PC1, reorder(Food, PC1), bg=PC1) +
  geom_col() +
  xlab("PC1 Loadings/Contributions") +
  ylab("Food Group") +
  scale_fill_gradient2(low="purple", mid="gray", high="darkgreen", guide=NULL) +
  theme_bw()
```



```r
## The inbuilt biplot() can be useful for small datasets
biplot(pca)
```


Scotland
Soft_drinks
N.Irel
Fresh_potatoes
Wales
PC1
PC2