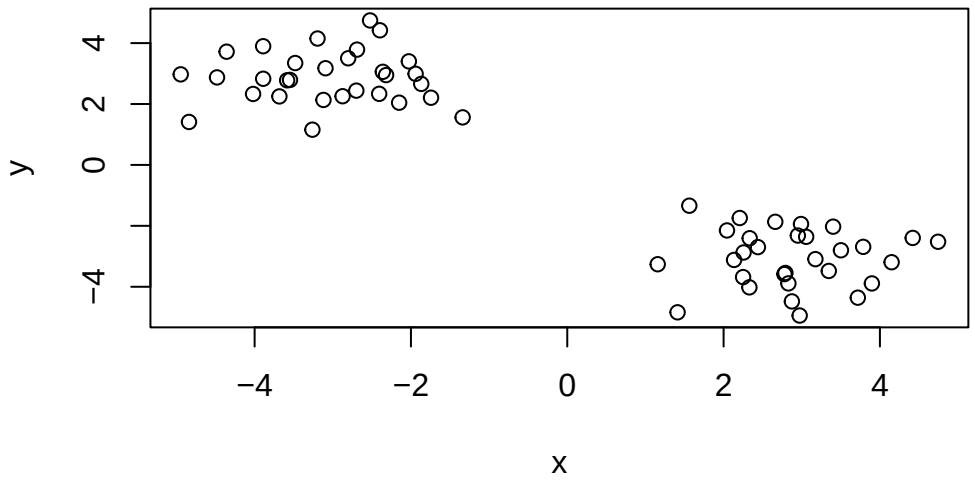
Machine Learning 1

Patrick Tran

First up kmeans()

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

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



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

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

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

Cluster means:
          x         y
1  2.872205 -3.050853
2 -3.050853  2.872205

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] 46.95606 46.95606
 (between_SS / total_SS =  91.8 %)

Available components:

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

Q. How many points are in each cluster?

```r
k$size
```

```
[1] 30 30
```

Q. How do we get to the cluster membership/assignment?

```r
k$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
```

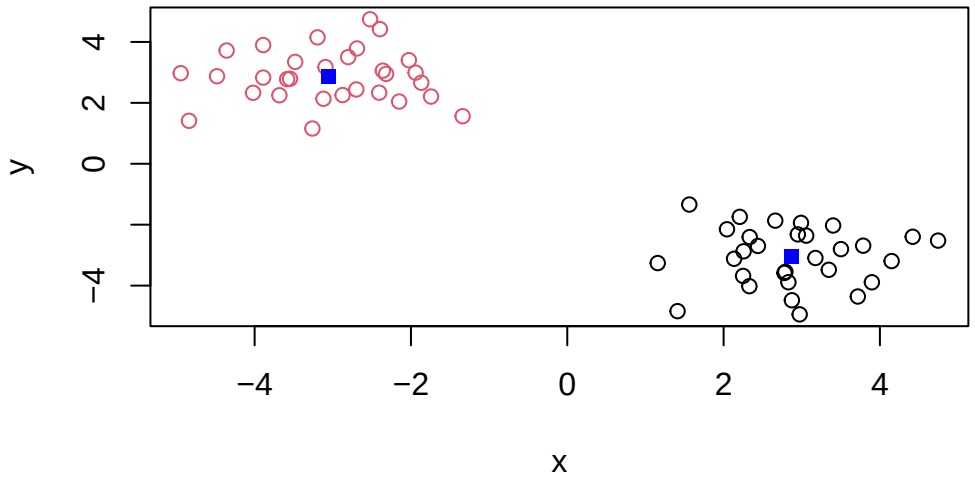
Q. What about cluster centers?

```r
k$centers
```

```
          x         y
1  2.872205 -3.050853
2 -3.050853  2.872205
```

Now 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 Hierarchial Clustering

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

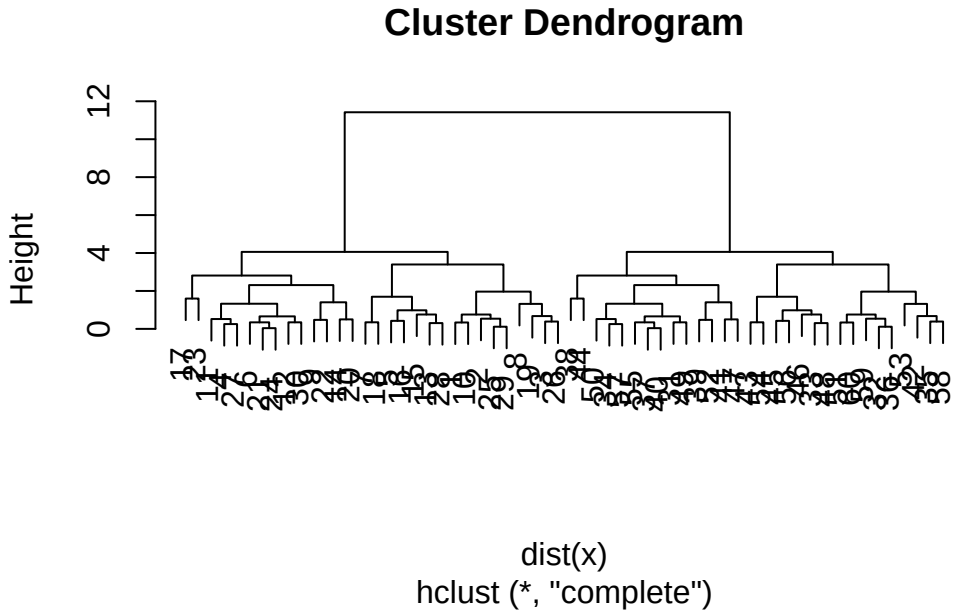
```
hc <- hclust(dist(x))
hc
```

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

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

Let's plot our hclust result

```
plot(hc)
```



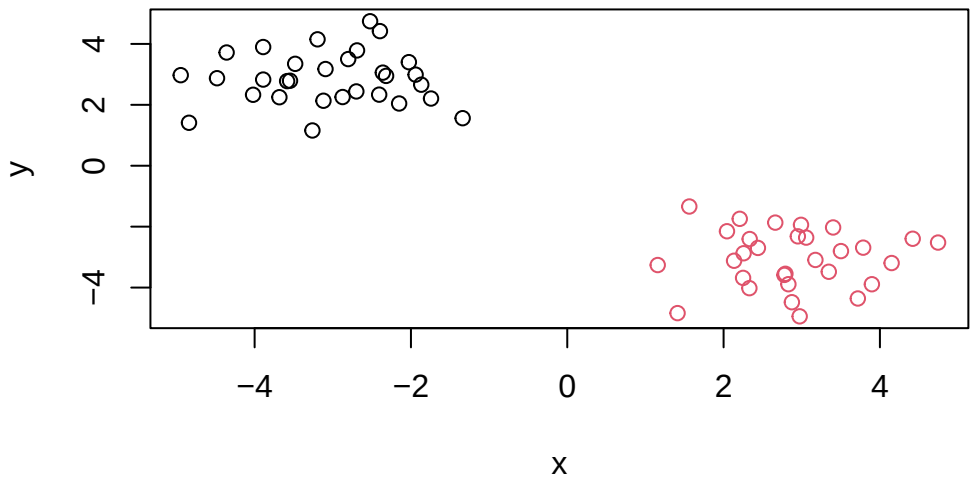
To get our cluster membership vector we need to "cut" the tree with `cutree()`

```
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 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 we plot our data with the hclust() results.

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



Principal Component Analysis(PCA)

PCA of UK food data

Read data from website and try a few visualizations.

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=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
```

Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

```
#Should be 17 rows and 5 columns before minus indexing.
dim(x)
```

```
[1] 17  4
```

```
ncol(x)
```

```
[1] 4
```

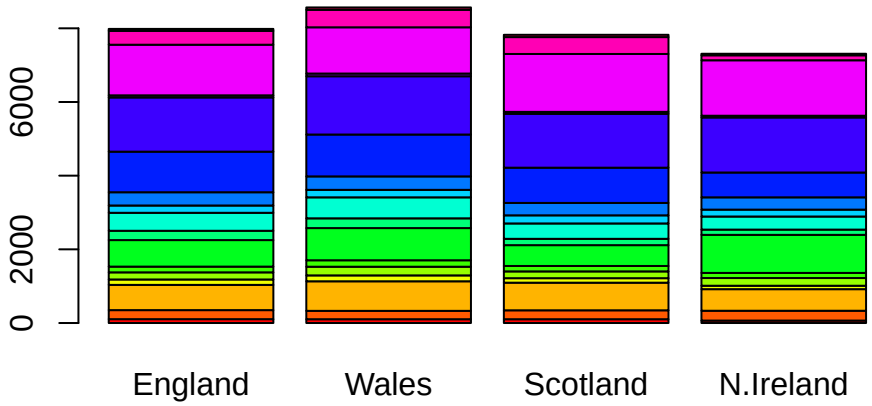
```
nrow(x)
```

```
[1] 17
```

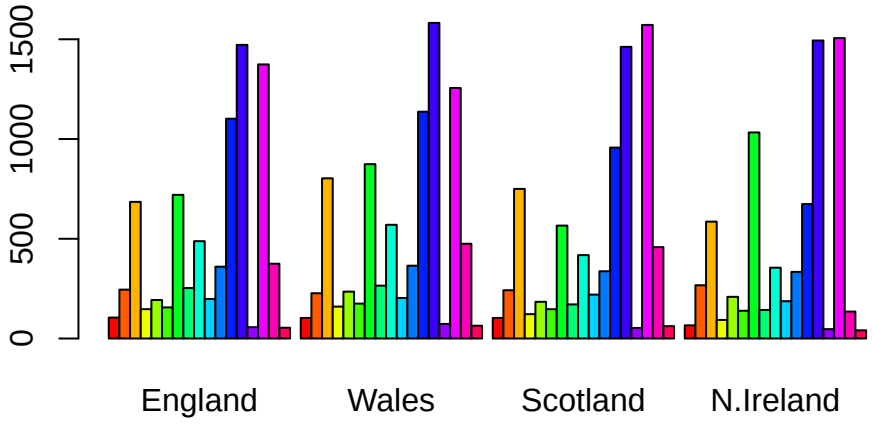
Q2. Which approach to solving the `row-names problem` mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

I prefer to use the argument setting `row.names=1`. It is more robust and you can run the block multiple times. If you use `x <- x[,-1]` repeatedly, it will delete columns.

```
cols <- rainbow(nrow(x))
barplot(as.matrix(x), col=cols)
```



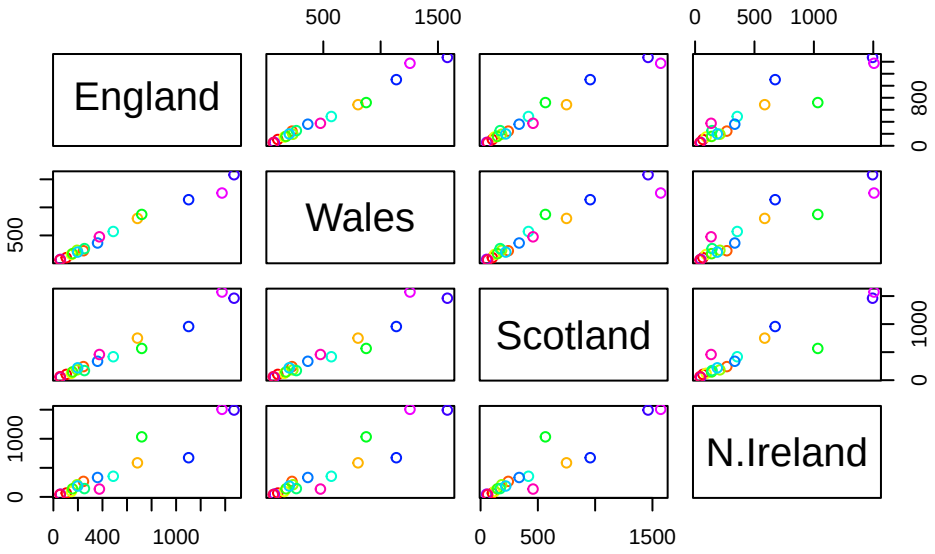

```
barplot(as.matrix(x), col=cols, beside = TRUE)
```



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

Changing it to `beside=FALSE` in the barplot() code.

```
pairs(x, col=cols)
```



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?

This plot shows all possible pairs of countries against each other. This is a matrix of plots. If a given point lies on the diagonal for a given plot then it means that the countries are similar.

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

N. Ireland differs in values compared to the other countries of the UK. The points on the plot do not follow the diagonal line pattern. We observe less of a trend.

PCA to the rescue!! The main base R PCA function is called `prcomp()` and we will need to give it the transpose of our input data!

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

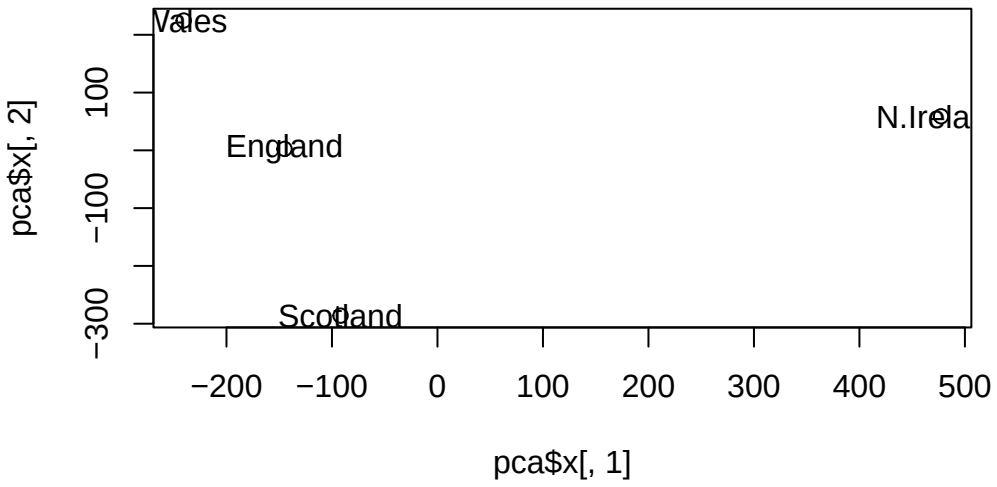
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

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

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

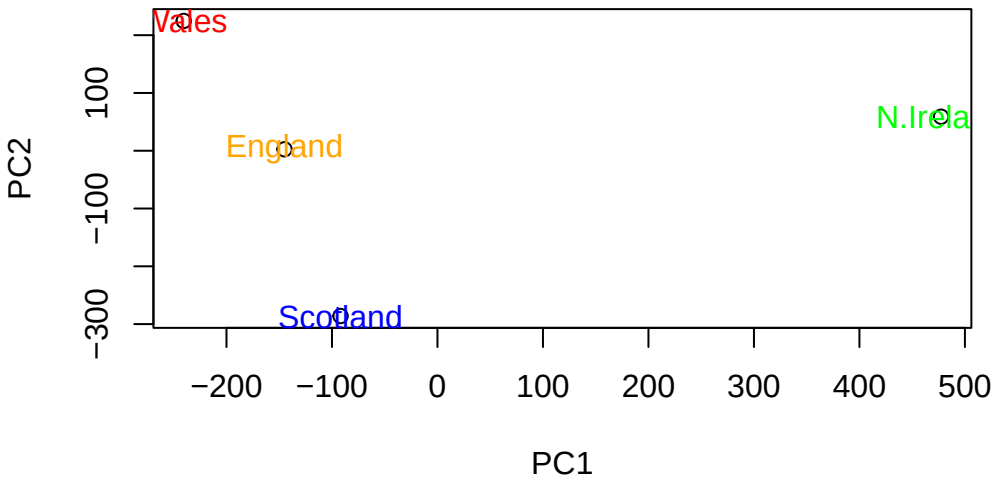
```
plot(pca$x[,1], pca$x[,2])
text(pca$x[,1], pca$x[,2], colnames(x))
```



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.

Color up the plot

```
country_cols <- c("orange", "red", "blue", "green")
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
text(pca$x[,1], pca$x[,2], colnames(x), col=country_cols)
```



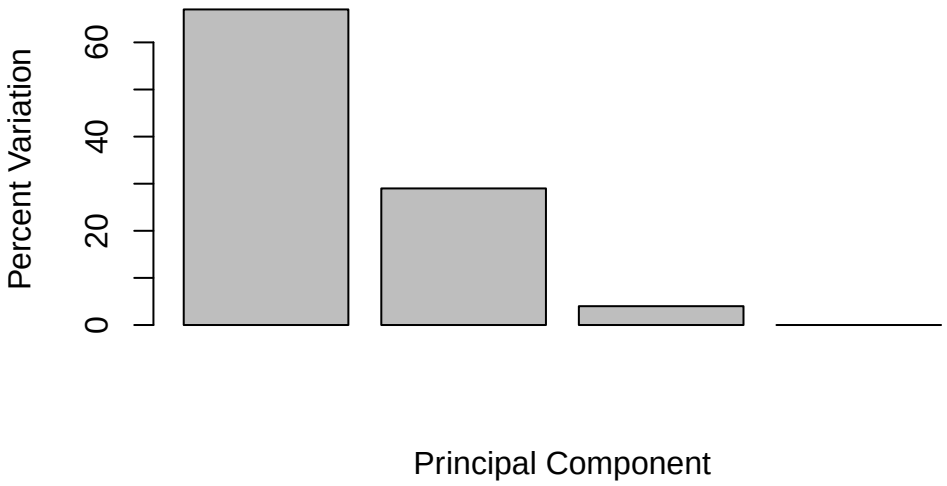

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

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

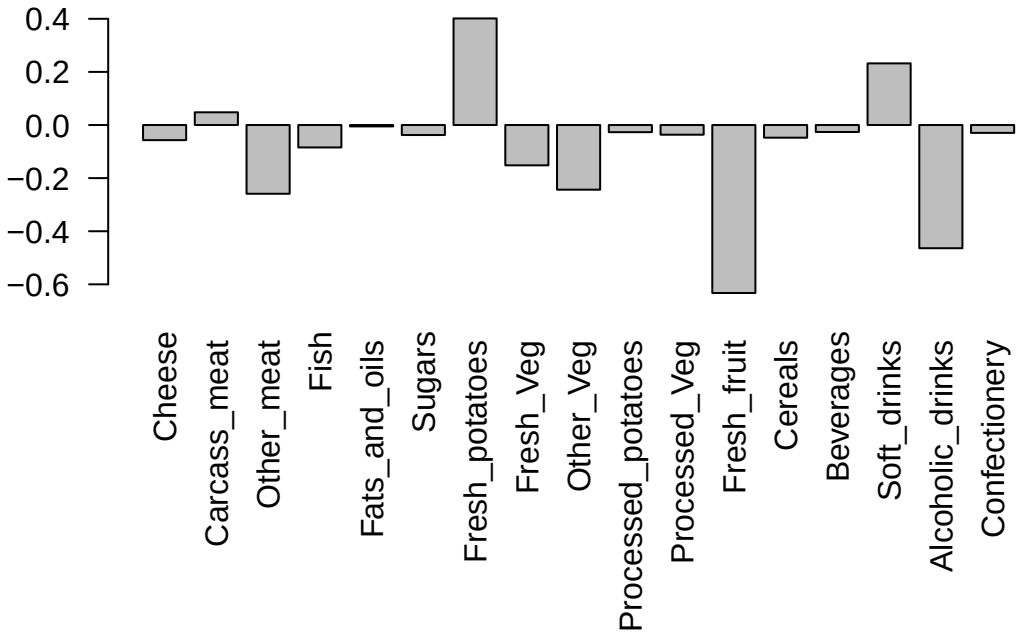
```
                             PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

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



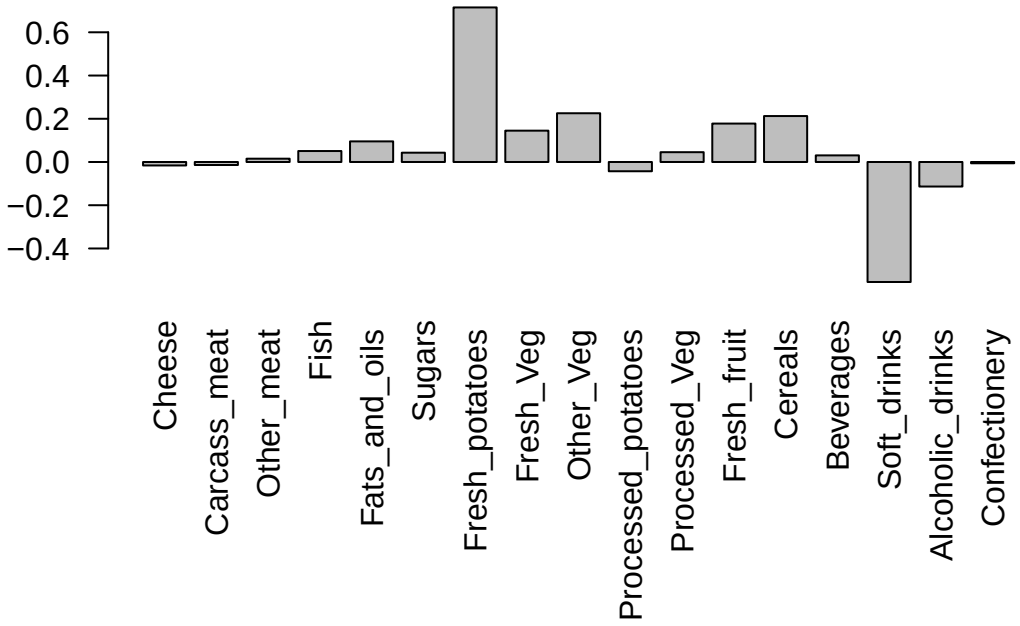
Digging Deeper

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



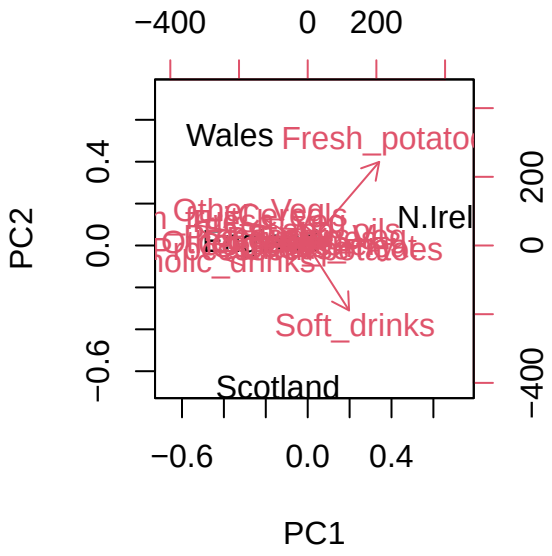
Q9. Generate a similar 'loading plot' for PC2. What two food groups feature prominantely and what does PC2 mainly tell us about?

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



The 2 prominent groups are `Fresh_potatoes` and `Soft_drinks`. Fresh potatoes push to right positive side of plot. Soft drinks push to left side of plot.

```
biplot(pca)
```



PCA of RNA-Seq data

Read in data from website

```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
       wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
gene1  439 458  408  429 420  90  88  86  90  93
gene2  219 200  204  210 187 427 423 434 433 426
gene3 1006 989 1030 1017 973 252 237 238 226 210
gene4  783 792  829  856 760 849 856 835 885 894
gene5  181 249  204  244 225 277 305 272 270 279
gene6  460 502  491  491 493 612 594 577 618 638
```

Q10. How many genes are samples are in this data set?

```
dim(rna.data)
```

```
[1] 100  10
```

```
#100 genes, 10 samples
```

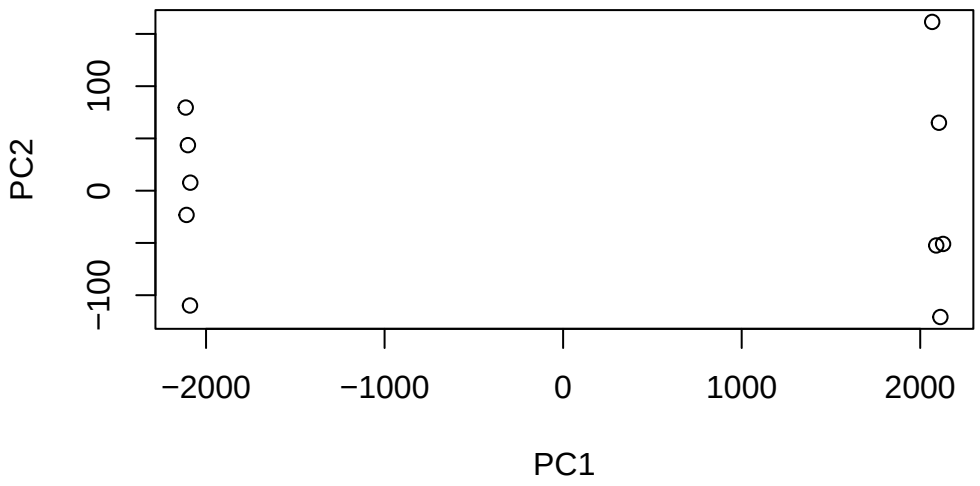
There is a nice summary of how well PCA is doing

```
pca <- prcomp(t(rna.data))
summary(pca)
```

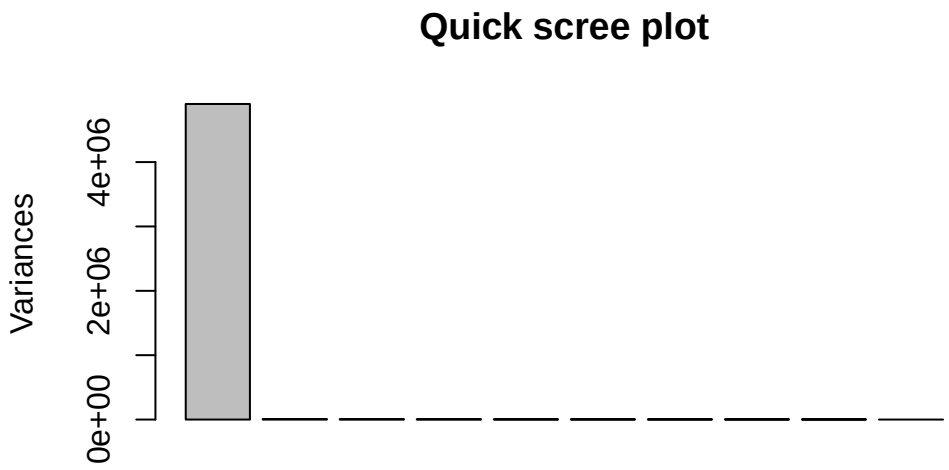
```
Importance of components:
                            PC1      PC2      PC3      PC4      PC5      PC6
Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
Cumulative Proportion     0.9917  0.9933  0.99471  0.99593  0.99691  0.99784
                           PC7      PC8      PC9      PC10
Standard deviation     65.29428 59.90981 53.20803 3.142e-13
Proportion of Variance  0.00086  0.00073  0.00057 0.000e+00
Cumulative Proportion   0.99870  0.99943  1.00000 1.000e+00
```

Do our PCA plot of this RNA-Seq data

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```



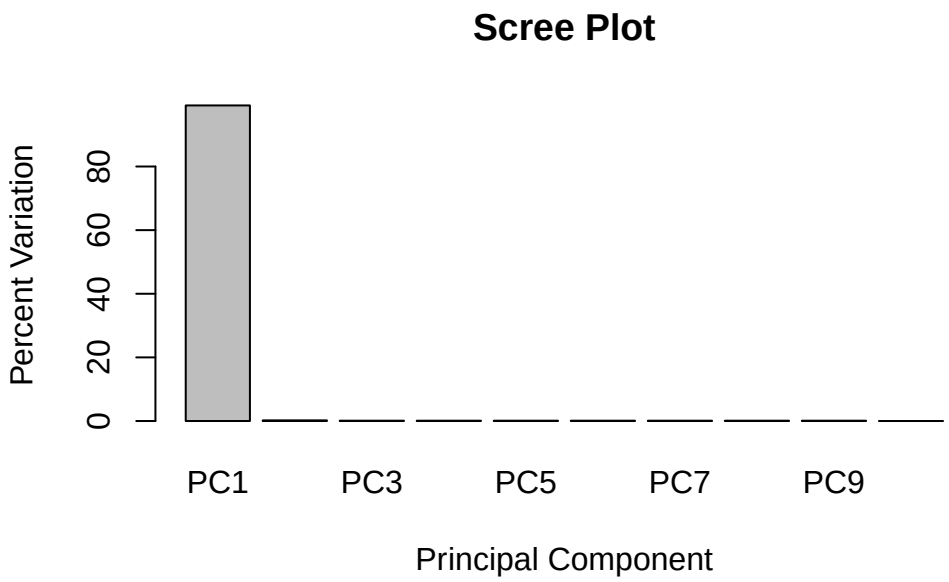
```r
plot(pca, main="Quick scree plot")
```



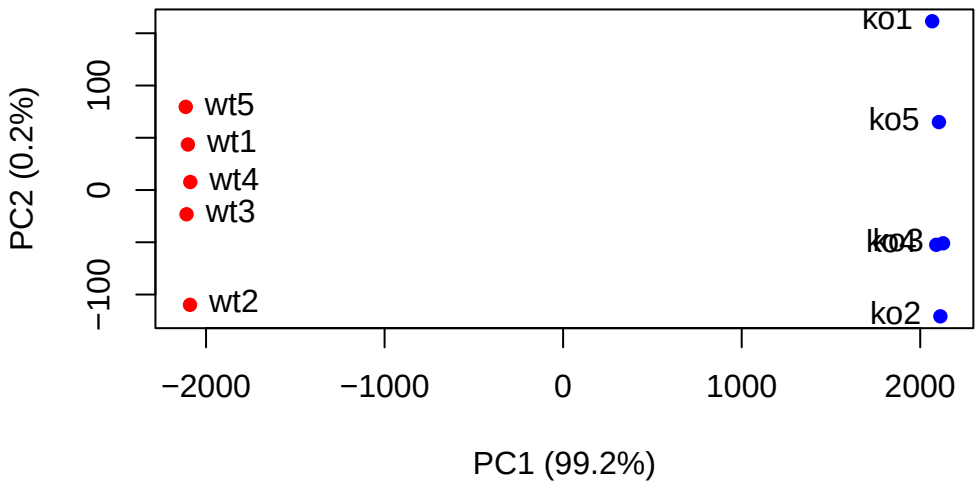

```r
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.var.per
```

```
[1] 99.2  0.2  0.1  0.1  0.1  0.1  0.1  0.1  0.1  0.0
```

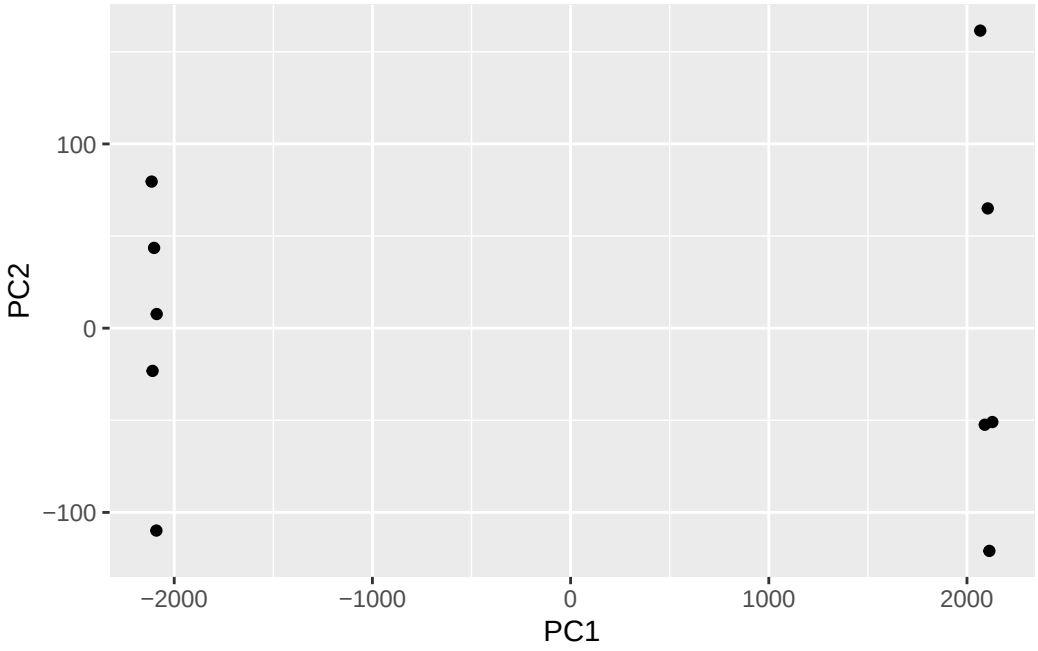
```r
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



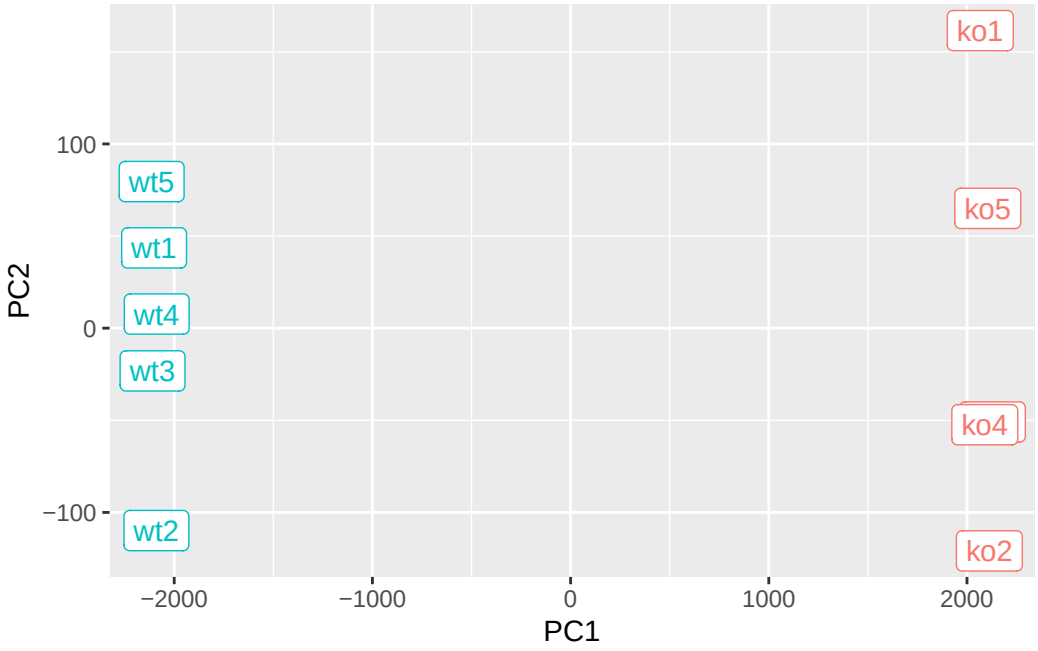

```
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

plot(pca$x[,1], pca$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca.var.per[2], "%)"))

text(pca$x[,1], pca$x[,2], labels = colnames(rna.data), pos=c(rep(4,5), rep(2,5)))
```



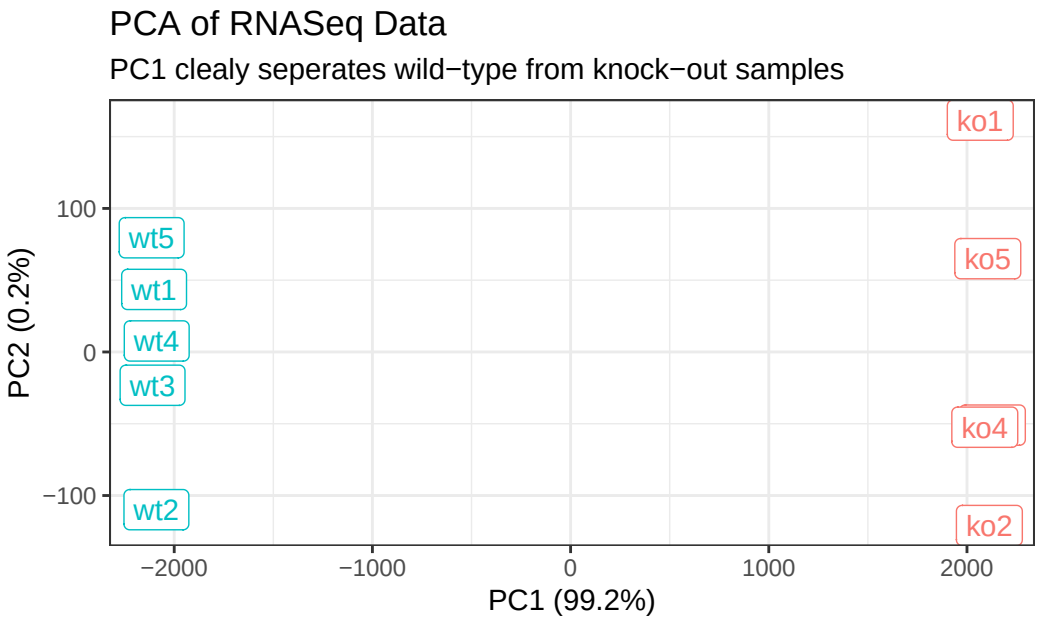

```
library(ggplot2)

df <- as.data.frame(pca$x)

ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```




```
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)

p <- ggplot(df) +
        aes(PC1, PC2, label=samples, col=condition) +
        geom_label(show.legend = FALSE)
p
```




```
p + labs(title="PCA of RNASeq Data",
       subtitle = "PC1 clealy seperates wild-type from knock-out samples",
       x=paste0("PC1 (", pca.var.per[1], "%)"),
       y=paste0("PC2 (", pca.var.per[2], "%)"),
       caption="Class example data") +
     theme_bw()
```


PCA of RNASeq Data
PC1 clealy seperates wild−type from knock−out samples
ko1
ko5
ko4
ko2
wt5
wt1
wt4
wt3
wt2
100
0
−100
PC2 (0.2%)
−2000
−1000
0
1000
2000
PC1 (99.2%)
Class example data