Lab 07

Bomin Xie

Class 07 Lab - Hands on with PCA

K-means introduction

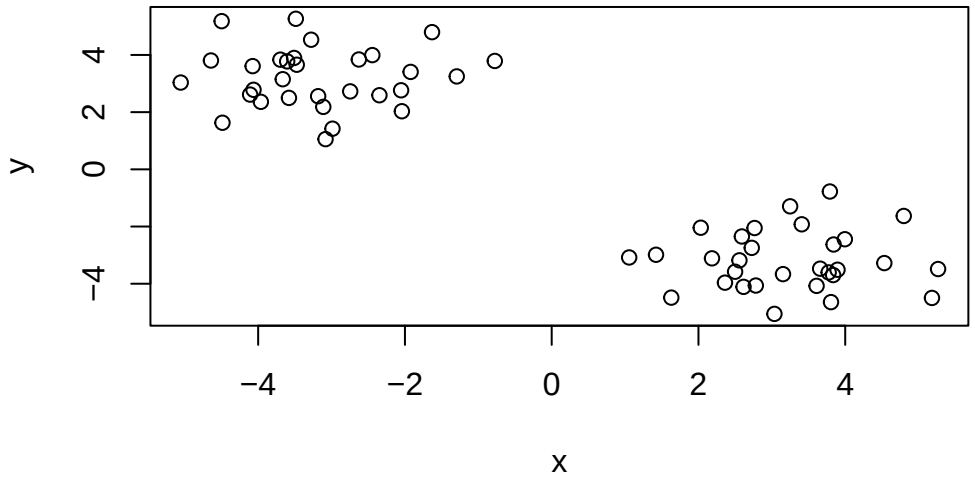
Demo using `kmeans()` function:

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

```
               x          y
 [1,] -2.3474961  2.5912493
 [2,] -3.2782508  4.5325165
 [3,] -4.4866748  1.6299398
 [4,] -2.0417117  2.0314303
 [5,] -3.4741257  3.6584457
 [6,] -5.0554016  3.0351633
 [7,] -3.9632364  2.3609818
 [8,] -3.4868229  5.2652836
 [9,] -1.9224321  3.4074212
[10,] -2.7457517  2.7262547
[11,] -3.5107349  3.8949765
[12,] -3.0816654  1.0558087
[13,] -3.1828069  2.5581082
[14,] -3.5806397  2.4991637
[15,] -3.6970575  3.8353318
[16,] -4.6435242  3.8060476
[17,] -4.4974623  5.1819306
[18,] -2.6283353  3.8416522
[19,] -1.6306688  4.7976336
[20,] -2.0522703  2.7648634
```

```
[21,] -3.6649957  3.1520641
[22,] -1.2937593  3.2492157
[23,] -3.1125500  2.1842230
[24,] -3.6047514  3.7742947
[25,] -4.1096495  2.6141985
[26,] -4.0751851  3.6097164
[27,] -0.7753824  3.7908369
[28,] -2.9870914  1.4215629
[29,] -2.4443364  3.9949799
[30,] -4.0633389  2.7817720
[31,]  2.7817720 -4.0633389
[32,]  3.9949799 -2.4443364
[33,]  1.4215629 -2.9870914
[34,]  3.7908369 -0.7753824
[35,]  3.6097164 -4.0751851
[36,]  2.6141985 -4.1096495
[37,]  3.7742947 -3.6047514
[38,]  2.1842230 -3.1125500
[39,]  3.2492157 -1.2937593
[40,]  3.1520641 -3.6649957
[41,]  2.7648634 -2.0522703
[42,]  4.7976336 -1.6306688
[43,]  3.8416522 -2.6283353
[44,]  5.1819306 -4.4974623
[45,]  3.8060476 -4.6435242
[46,]  3.8353318 -3.6970575
[47,]  2.4991637 -3.5806397
[48,]  2.5581082 -3.1828069
[49,]  1.0558087 -3.0816654
[50,]  3.8949765 -3.5107349
[51,]  2.7262547 -2.7457517
[52,]  3.4074212 -1.9224321
[53,]  5.2652836 -3.4868229
[54,]  2.3609818 -3.9632364
[55,]  3.0351633 -5.0554016
[56,]  3.6584457 -3.4741257
[57,]  2.0314303 -2.0417117
[58,]  1.6299398 -4.4866748
[59,]  4.5325165 -3.2782508
[60,]  2.5912493 -2.3474961
```

```
plot(x)
```



Now apply kmeans to see how it works with these data:

```
k = kmeans(x, center=2, nstart = 20)
k
```

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

Cluster means:
          x         y
1 -3.181270  3.201569
2  3.201569 -3.181270

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 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

Within cluster sum of squares by cluster:
[1] 62.45604 62.45604
 (between_SS / total_SS =  90.7 %)
```

```
Available components:

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

```
k$size # gives how many points in each cluster
```

```
[1] 30 30
```

```
k$cluster # gives how we get to the cluster membership/assignment
```

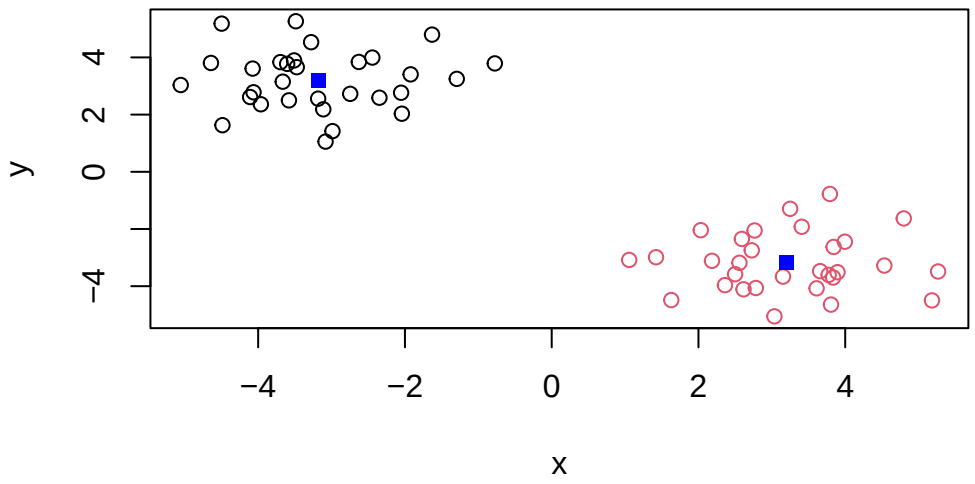
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
k$centers # gives us what are the cluster centers
```

```
          x         y
1 -3.181270  3.201569
2  3.201569 -3.181270
```

Plot the kmeans result based on the main results:

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



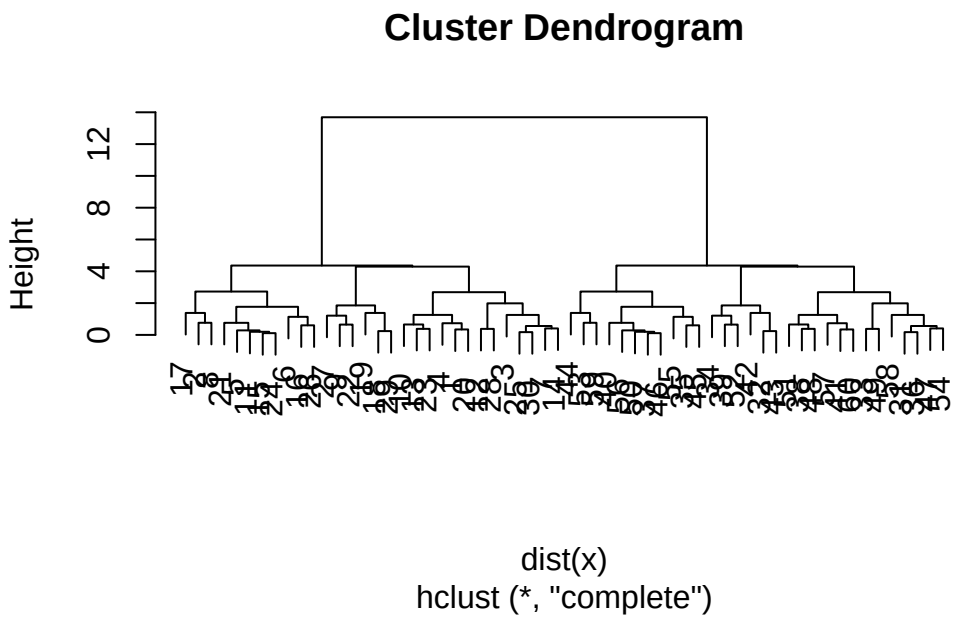
Hierarchical clustering hclust()

We will then cluster the same data with `hclust()`.

```
hc = hclust(dist(x)) # Need a distance matrix as an input (dissimilarity)
hc
```

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

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

```
plot(hc)
```



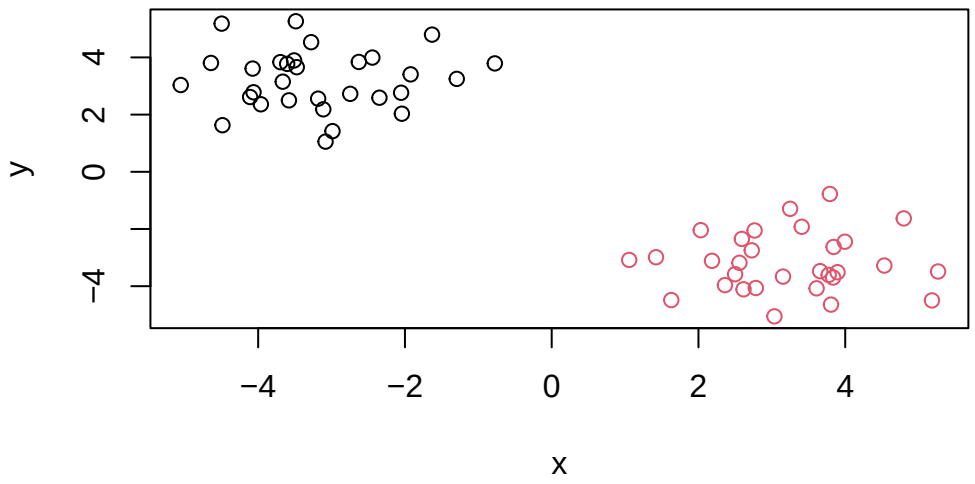
To get the cluster membership vector, we need to "cut" the tree:

```
group1 = cutree(hc, h=8)
group1
```

```
 [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
```

We could plot our data with the result.

```
plot(x, col=group1)
```



PCA of UK food data

Import data and general visualization

Import data from website and try few visualization:

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

Examine the number of rows and columns:

```
dim(x)
```

```
[1] 17  5
```

Q1: Based on the result, there are 17 rows and 5 columns in the new dataframe "x".

Preview a portion of the data:

```
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
```

Correct the error of including the country as part of the data:

```
# Note how the minus indexing works
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
```

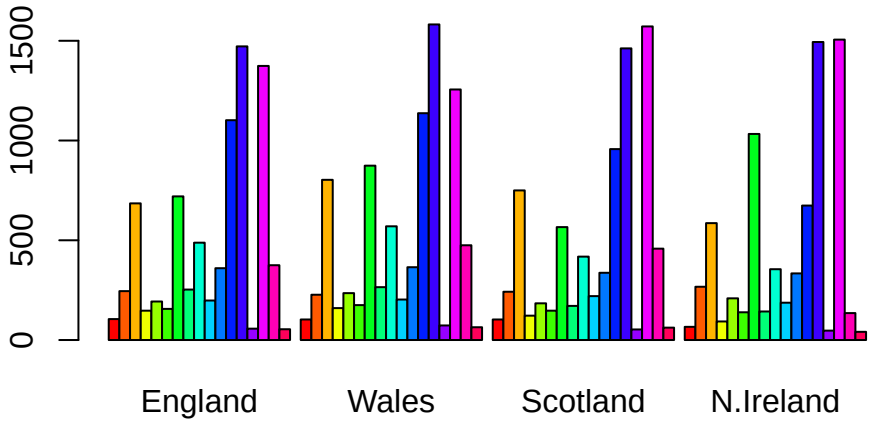
```
dim(x)
```

```
[1] 17  4
```

Q2: An alternative approach of setting `row.names` as the first column would be more convenient if the structure of the data frame is known. Meanwhile, the deletion of first column might cause trouble of deleting all data if performed multiple times.

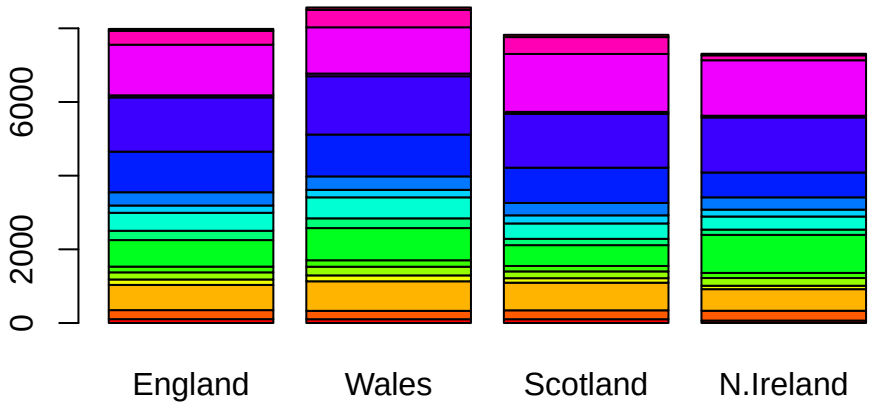
Plot the table in the form of barplots:

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



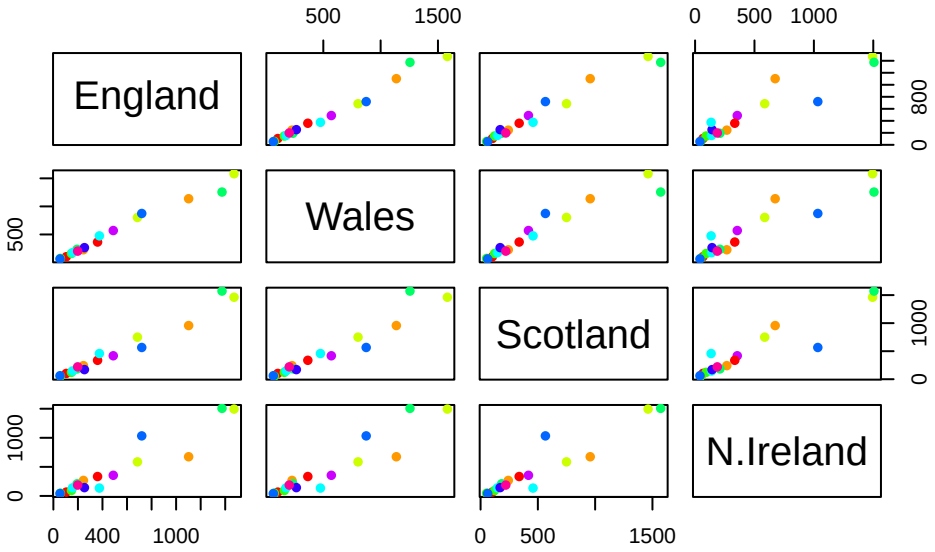
Q3: If we would like to change the barplot to a stacked barplot, we could change the argument `beside` as FALSE:

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



Q5: a generation of all pairwise plots might help distinguish the differences between different regions. The plot is a pair-wise comparison on different countries of UK. The diagonal value indicates the similarities between the pair of regions (i.e. England vs. Wales, or England vs. N. Ireland), and if the values of a certain factor in two different regions are similar between the two regions, the point is closer to the diagonal line. (The figure is depicted below.)

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



Q6. The main difference between N.Ireland and other countries of UK in this dataset is the amount of Fresh potatoes, as depicted as the blue dot in the figure.

PCA to the rescue

Using R `prcomp()` function to generate a PCA. Note the observation should be in rows and variables should be in columns (which we need to transpose the original dataframe `x`)

```
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
```

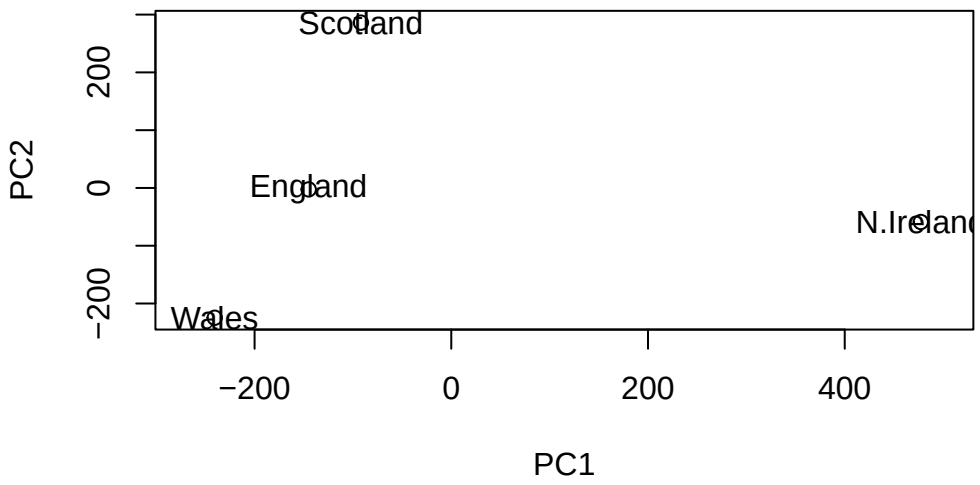
To make the new PCA plot we access `pca$x`:

```
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
```

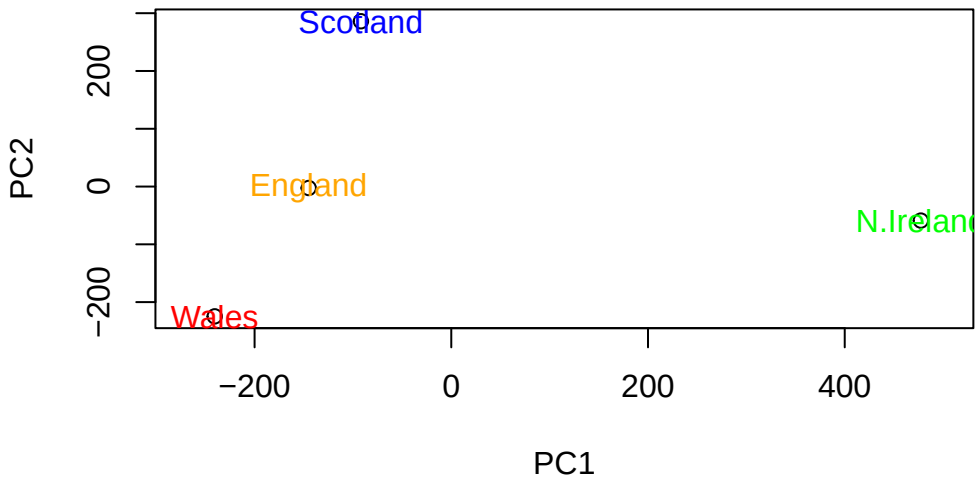
Q7:

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



To add more color, we could add a color vector to text:

```
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","green"))
```

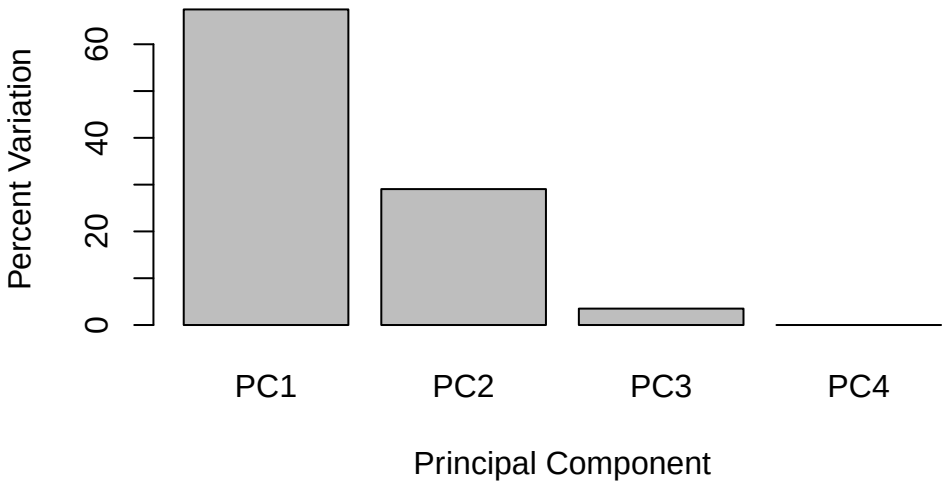


Using scree plot to determine the variation in the original data of each PC:

```
pca_sum = summary(pca)
pca_sum$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
```

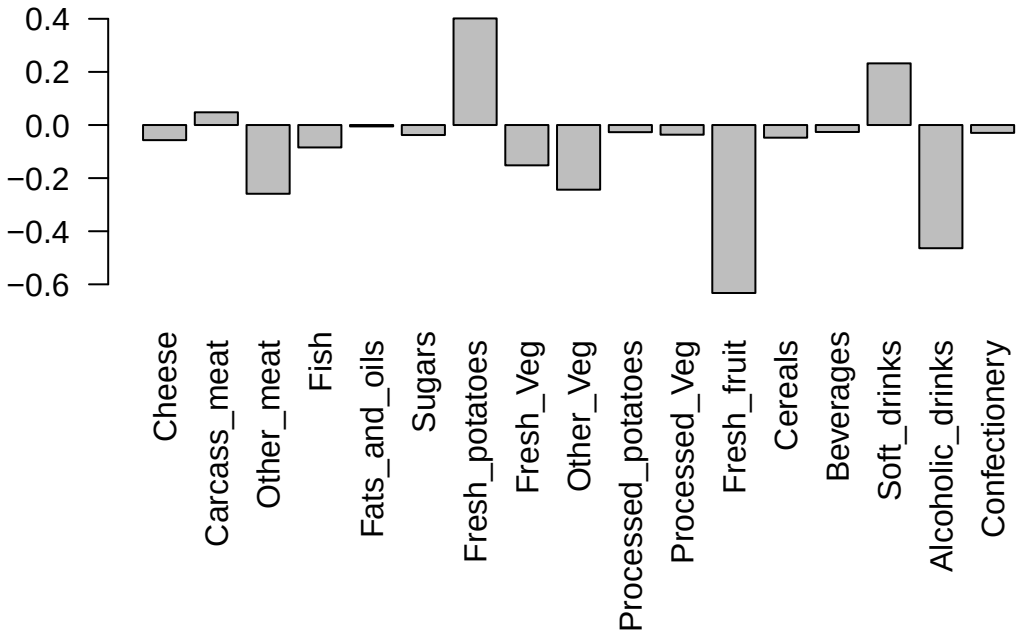
```
barplot(pca_sum$importance[2,] * 100, xlab="Principal Component", ylab="Percent Variation"
```



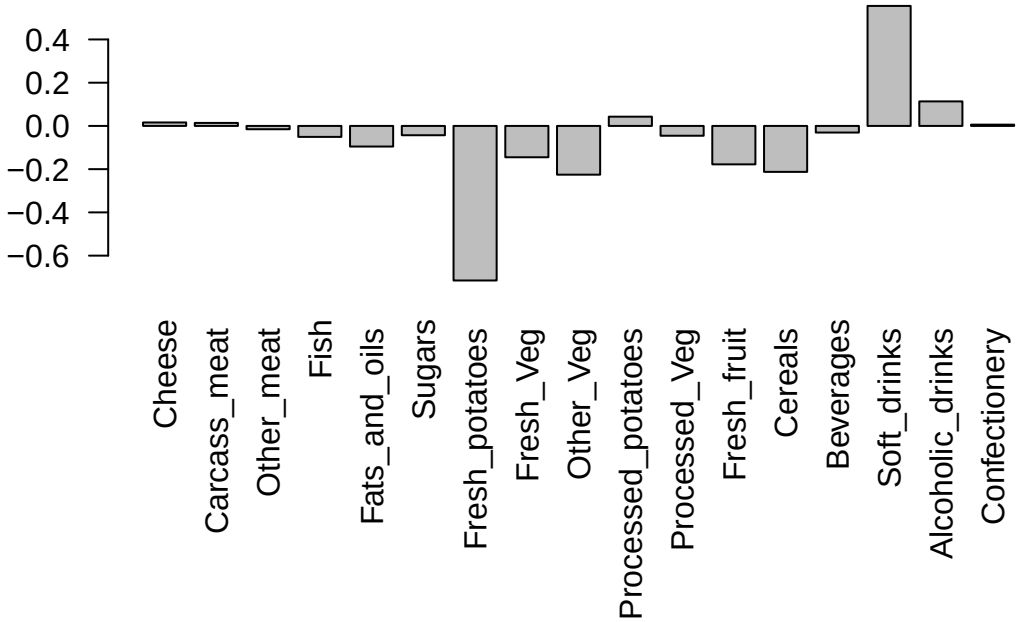
Variable loadings

To examine the influence of each original variables on each PCA component, we are using `rotation`:

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




```
barplot( pca$rotation[,2], las=2 )
```



Q9: Based on the plotting, the two food groups feature prominently are fresh potatoes and soft drinks. The soft drink consumption pushed other countries below the Wales and the consumption of Fresh potatoes allowed the separation of those countries on the figure.

PCA of RNA-seq data

Pull down the data of RNA-seq 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
```

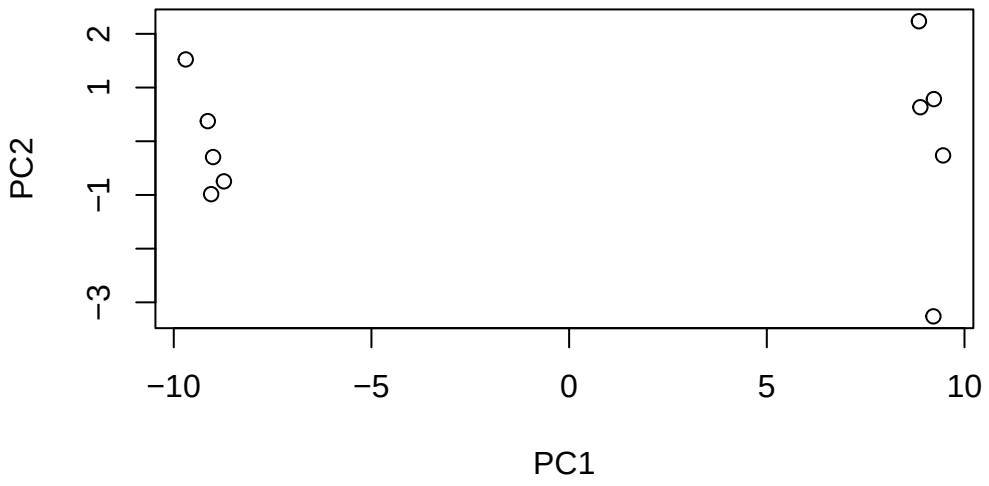
```
dim(rna.data)
```

```
[1] 100  10
```

Q10: According to the output, there are 100 genes and 10 samples in the data set.

Perform PCA on the dataset:

```
pca <- prcomp(t(rna.data), scale=TRUE)
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```

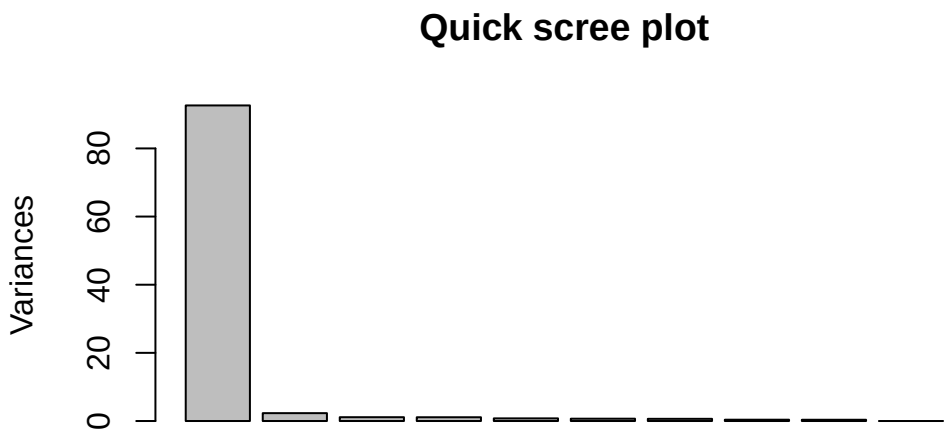



```
summary(pca)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     9.6237 1.5198 1.05787 1.05203 0.88062 0.82545 0.80111
Proportion of Variance 0.9262 0.0231 0.01119 0.01107 0.00775 0.00681 0.00642
Cumulative Proportion  0.9262 0.9493 0.96045 0.97152 0.97928 0.98609 0.99251
                           PC8     PC9      PC10
Standard deviation     0.62065 0.60342 3.347e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

Examine the variance per PC:

```
plot(pca, main="Quick scree plot", xlab="Importance of components")
```

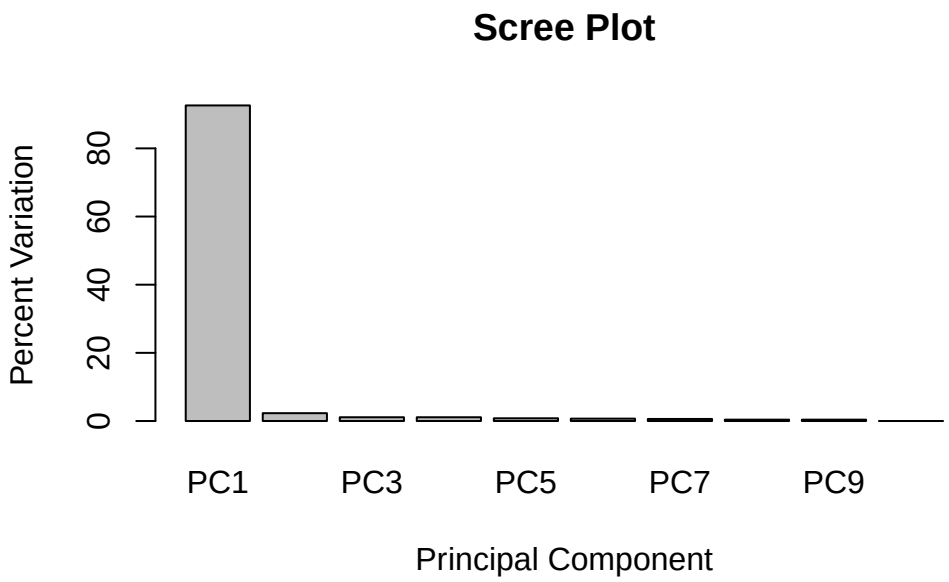


Importance of components

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

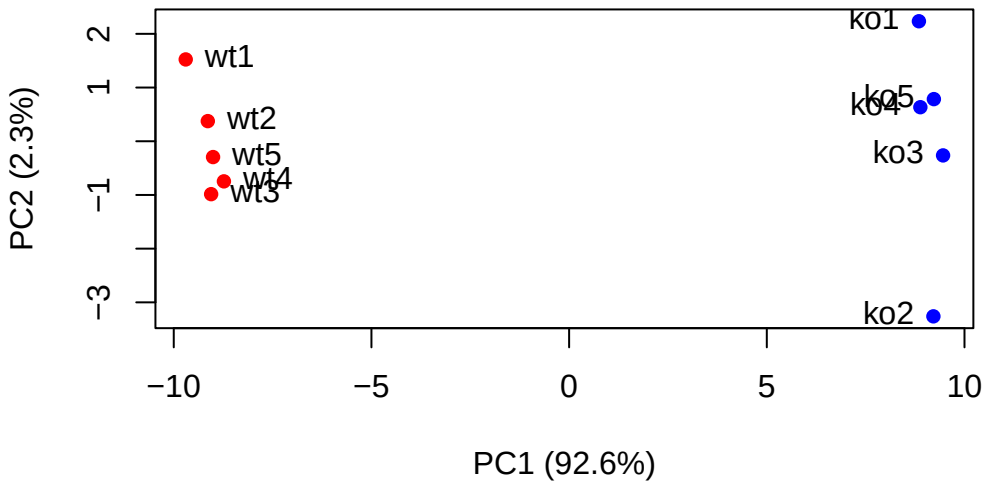
```
[1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

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



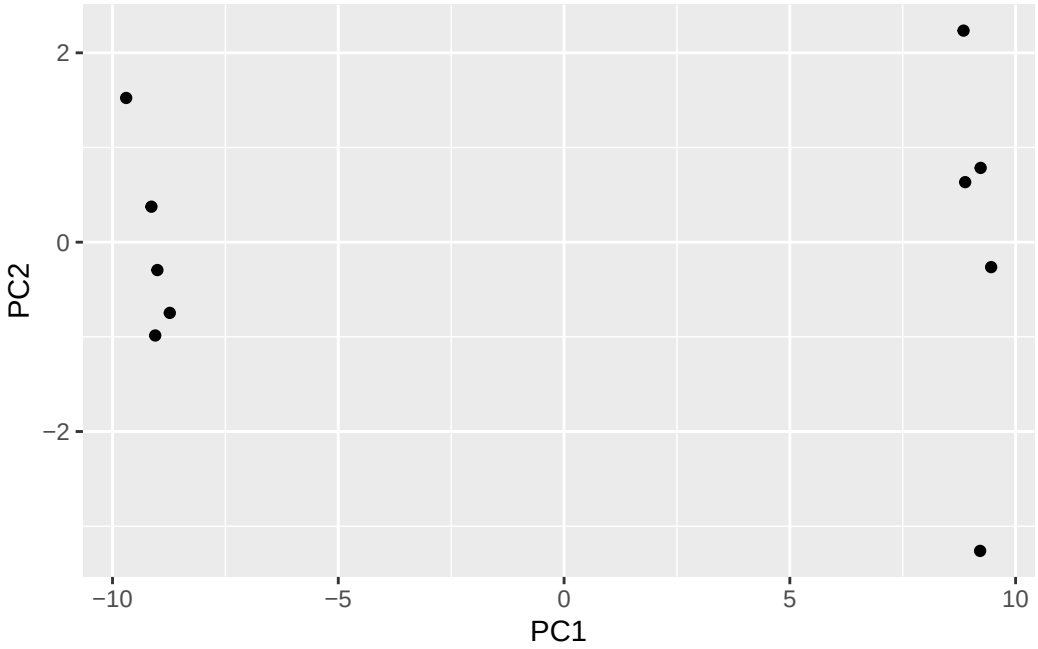
Adjust the main PCA plot, adding notations and names of each sample:

```
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)))
```

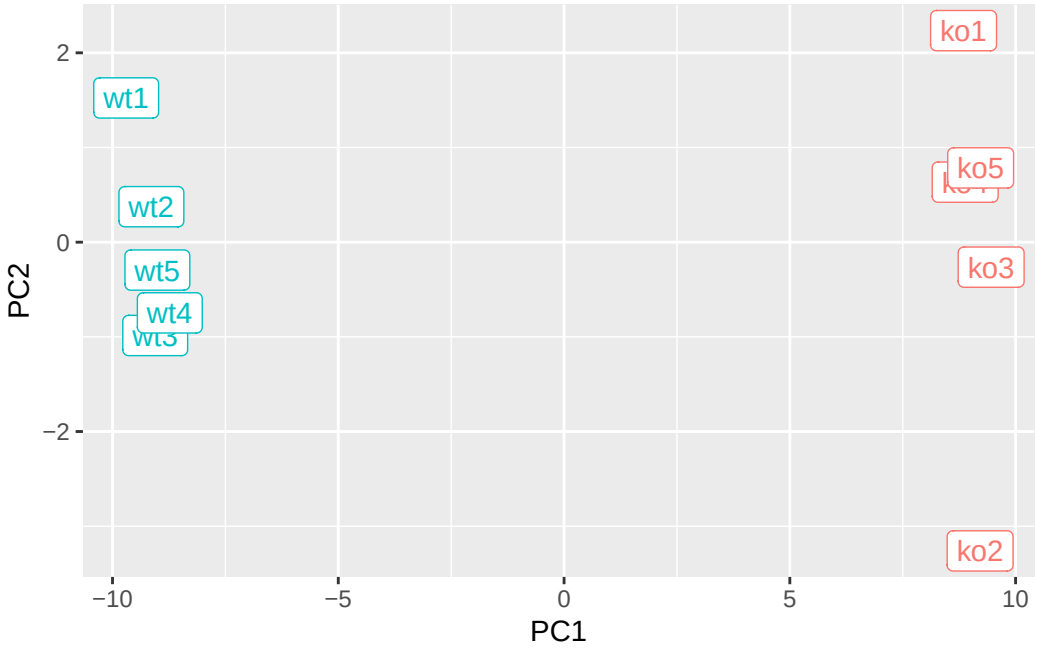


Using ggplot2 package to facilitate the plotting

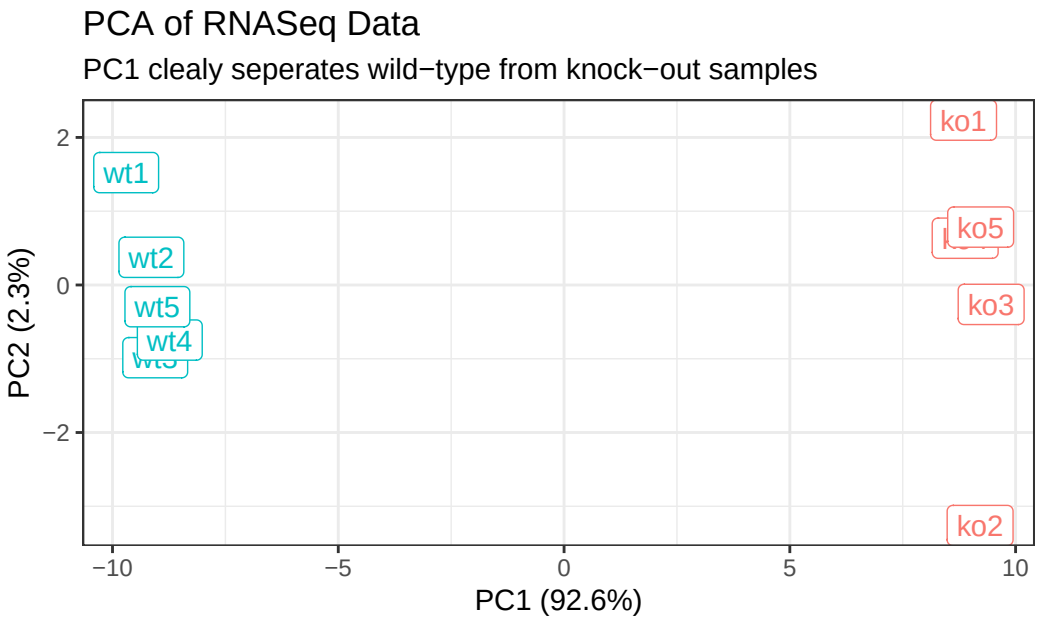
```
library(ggplot2)

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

# Our first basic plot
ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```




```
# Add a 'wt' and 'ko' "condition" column
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
wt1
ko5
wt2
wt5
ko3
wt4
ko2
2
0
−2
PC2 (2.3%)
−10
−5
0
5
10
PC1 (92.6%)
Class example data