Lecturer7_machine learning

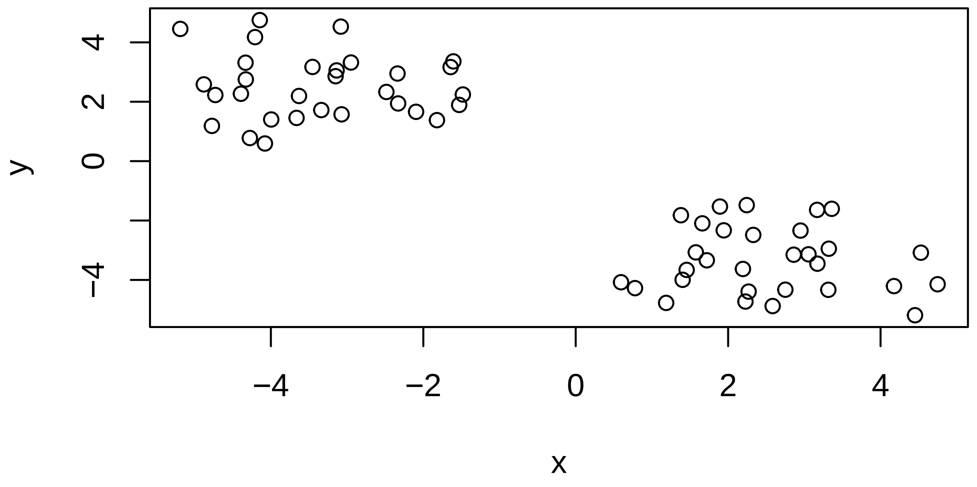
Longmei Zhang A17012012

We are going to learn how to apply different machine learning methods, beginning with clustering.

The goal is to find groups/clusters in the input dataset.

First, make up some data with clear groups. For this I will used the `rnorm()` function.

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

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



Use `kmeans()` function setting k to 2 and nstart = 20.

> Q. How many points are in each cluster?

there are 30 points in each cluster

```r
km <- kmeans(z, centers = 2)
km
```

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

Cluster means:
          x         y
1 -3.337664  2.509843
2  2.509843 -3.337664

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] 70.41314 70.41314
 (between_SS / total_SS =  87.9 %)

Available components:

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

> Q. What 'component' of your result object details - cluster size? - cluster assignment/membership? - cluster center?

results in kmeans object `km`

```r
km$size
```

```
[1] 30 30
```

```r
km$cluster
```

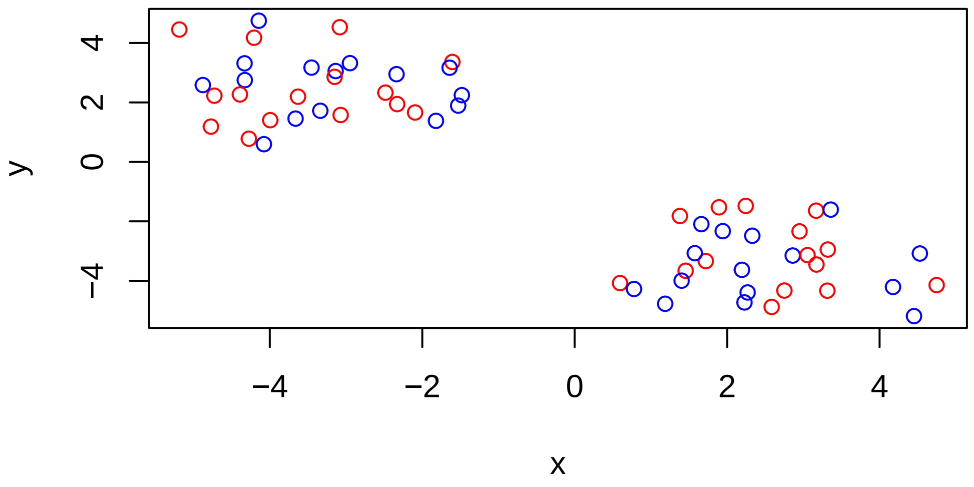
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
km$centers
```

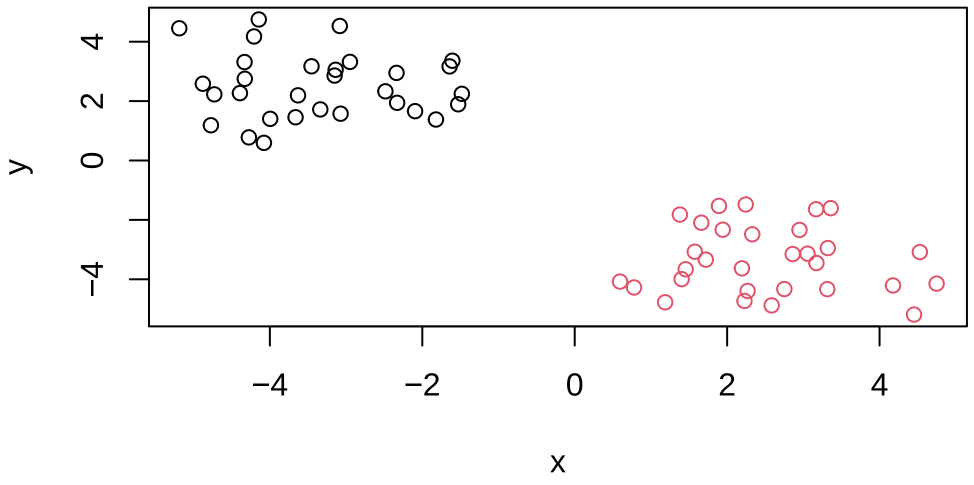
```
          x         y
1 -3.337664  2.509843
2  2.509843 -3.337664
```

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

```r
##R will recycle the shorter color vector to be the same length as the longer in z
plot(z, col = c("red", "blue"))
```

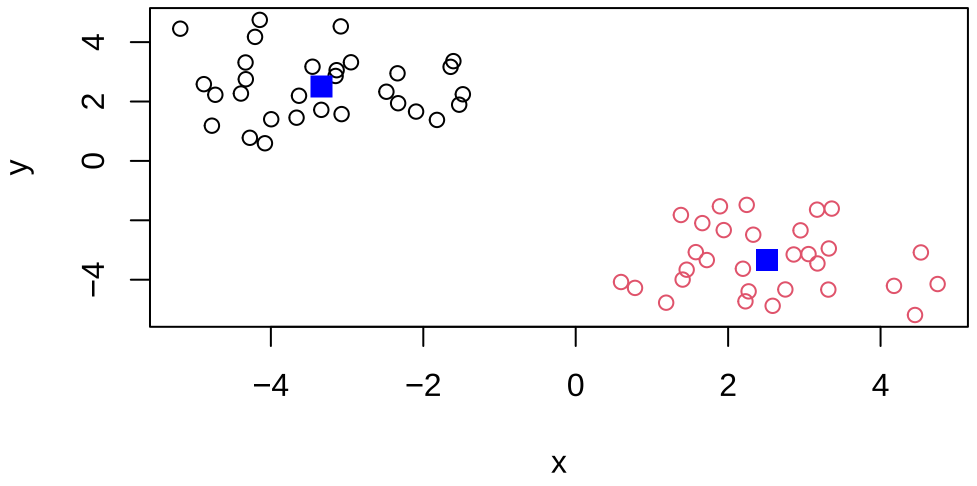


```r
plot(z, col = km$cluster)
```

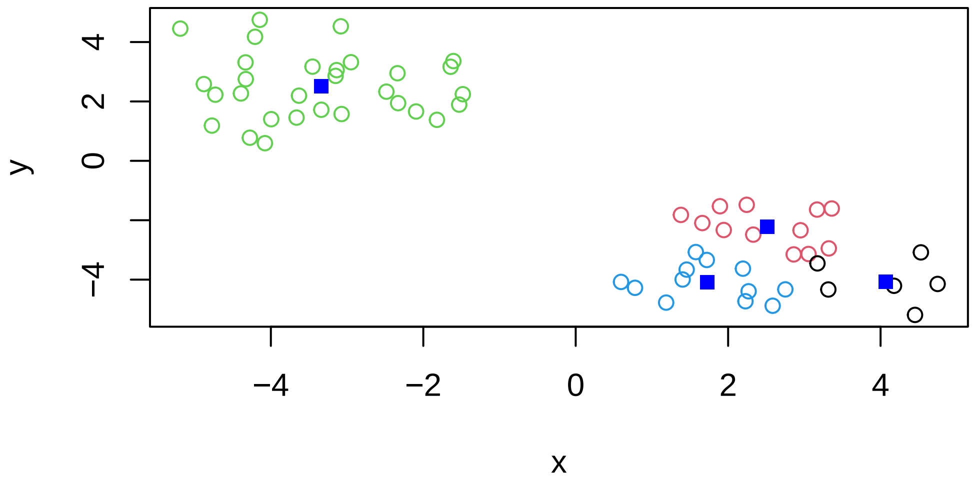


We can use the `points()` function to add new points to the existing plot (like the cluster centers)

```
plot(z, col = km$cluster)
points(km$centers, col  = "blue", pch = 15, cex = 1.5)
```



```
km2 <- kmeans(z, centers = 4)
plot(z, col = km2$cluster)
points(km2$centers, col = "blue", pch = 15)
```



Hierarchical Clustering

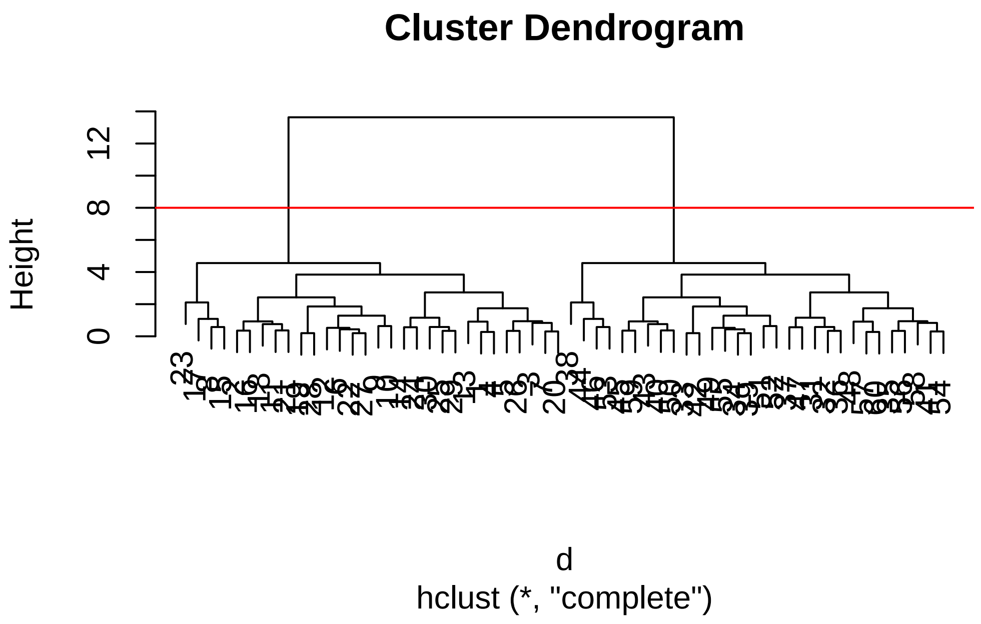
Lets take out same made-up data and test with hierarchical clustering

First we need a distance matrix of our data to be clustered

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

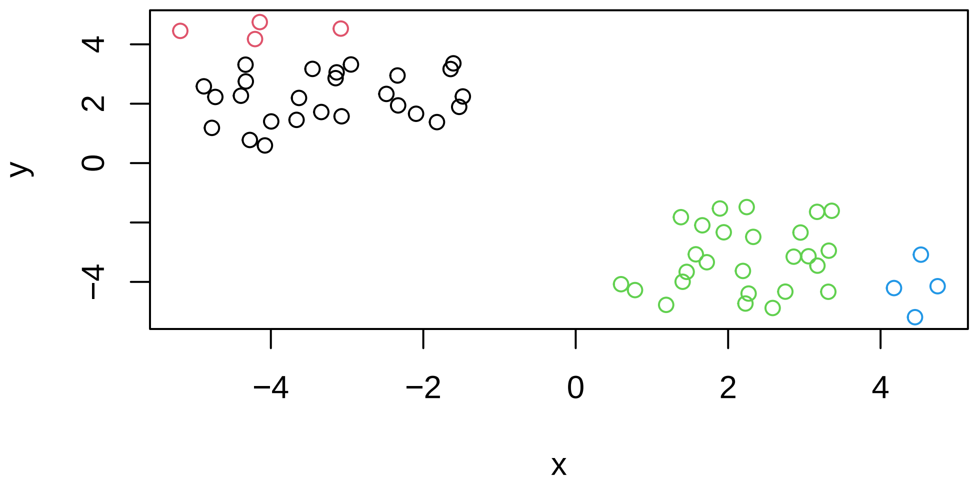
```
Call:
hclust(d = d)

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

```
plot(hc)

## from this breaking point, data on different branches are from different clusters.
abline(h=8, col="red")
```



Getting clustering membership vector by cutting the tree with the `cutree()` function.

```
groups <- cutree(hc, h=4)
plot(z, col = groups)
```



PCA of UK food data

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

we can use `nrow()` and `ncol()` to find the dimension of data frame. There are 17 rows and 5 columns

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

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

```
##first approach: set the first column as row names and remove the first column
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
```

```r
dim(x)
```

```
[1] 17  4
```

```r
##second approach: read the data file again and set the row.names argument of read.csv() to
x <- read.csv(url, row.names=1)
head(x)
```

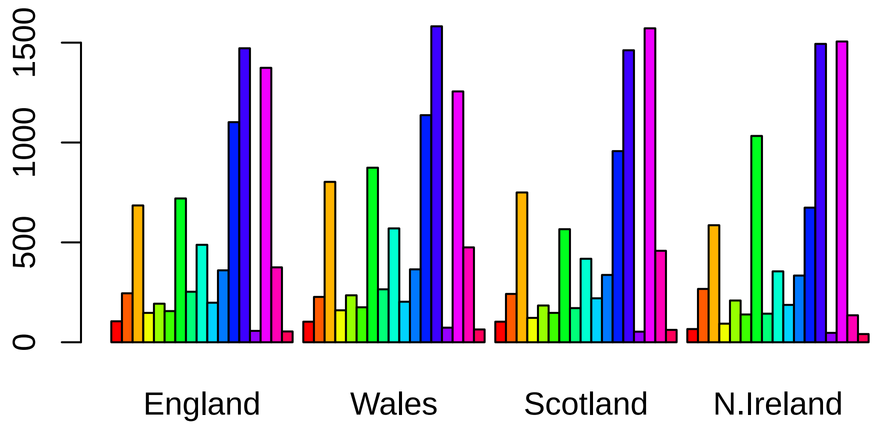
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars  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

> 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?

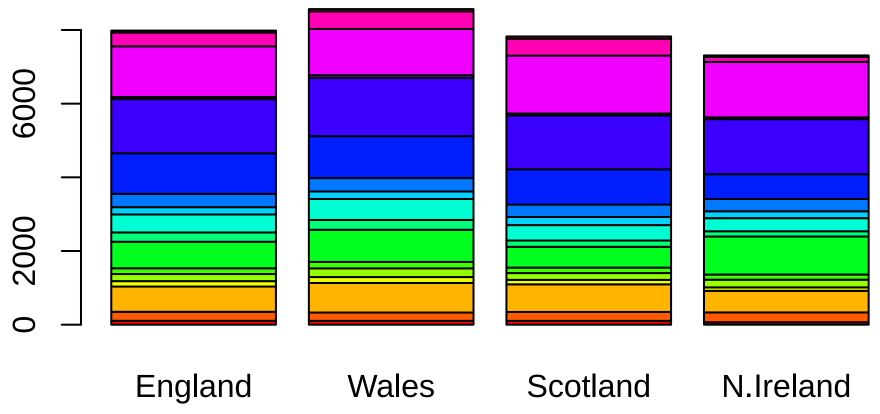
I prefer the second approach since it is easier. Also, if we use the first approach, the number of column would continue to be reduces if we run the code several times since the reduced data frame are reassigned to x, and the new x would be reduced again if we re-run the code.

Trying to visualize the data with different graph

```r
## not really helping
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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



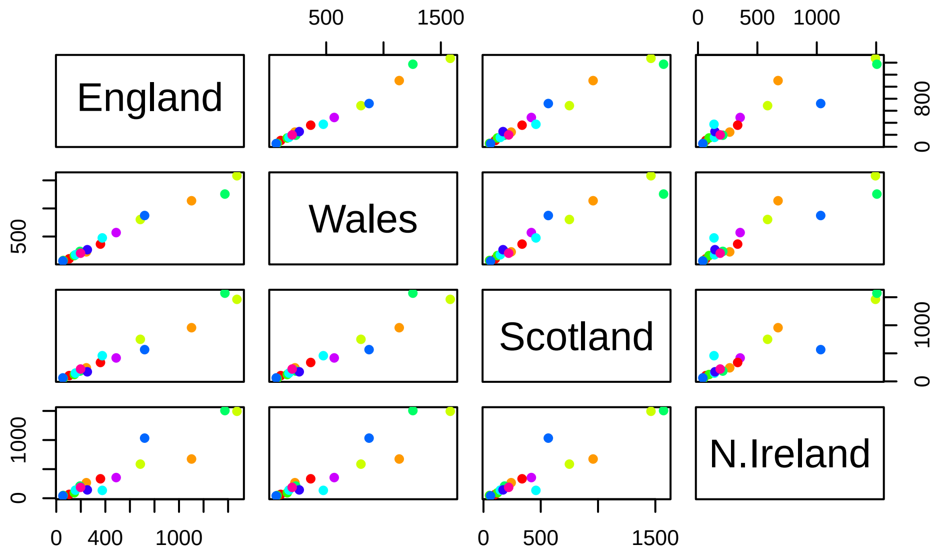
> Q3: Changing what optional argument in the above barplot() function results in the following plot?

changing the `beside` parameter to F would result in a stacked bar graph.

> 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?

If a given point lies on the diagonal, it means that the two countries expressed by this graph have similar data for this specific parameter. Its difficult to see structure and trend even in this small dataset. We would not work on bigger data with thousands of things we measure (dimension).

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



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

N. Ireland have lower consumption of cheese, fish, other meats, fresh fruits, vegetables, and alcoholic drinks and have higher consumption of potatoes.

PCA to the rescue

PCA works well when we are measuring a lot of things. Main function in base R to do PCA is called `prcomp()`.

```r
## transpose the table with samples(countries) as columns

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
```

Lets see what is inside this `pca` object that we produced from running `prcomp()`

```
attributes(pca)
```

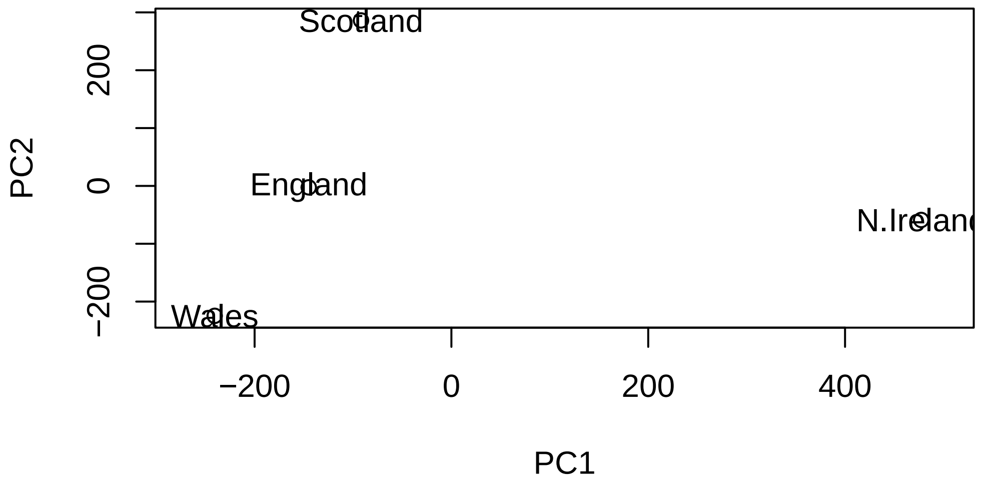
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -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
```

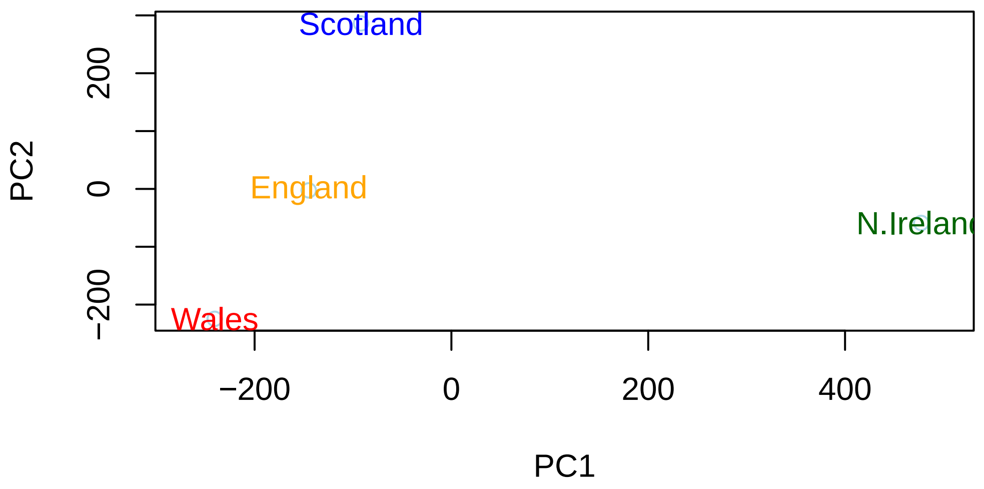
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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

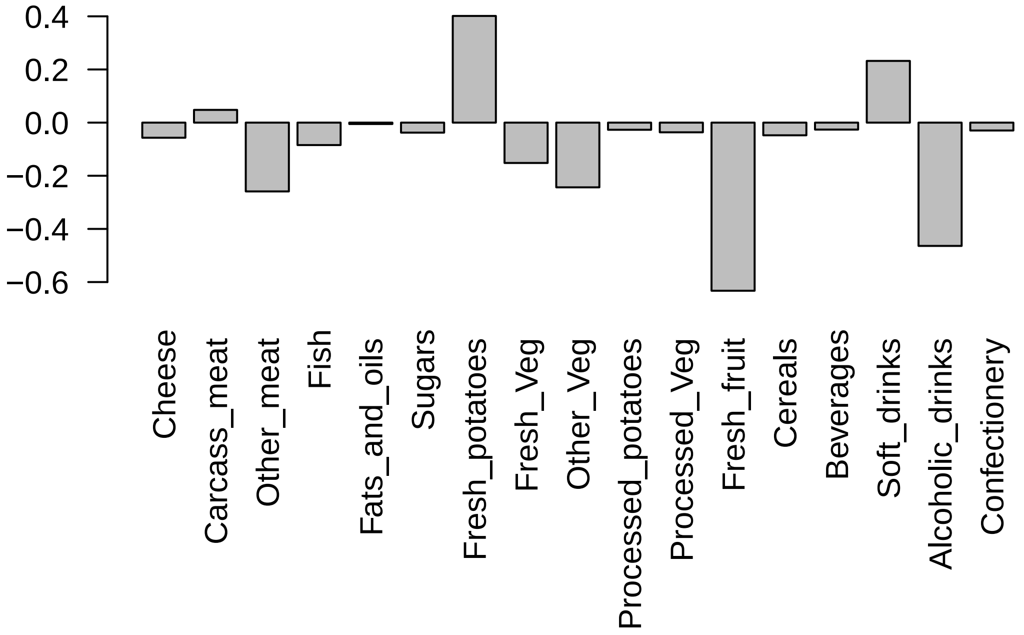
```r
plot(pca$x[, 1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), col = "lightblue")
text(pca$x[,1], pca$x[,2], colnames(x), col = c("orange",  "red", "blue","darkgreen"))
```



approximately 67% of the variance in the data is accounted for by the first principal component, and approximately 97% is accounted for in total by the first two principal components. In this case, we have therefore accounted for the vast majority of the variation in the data using only a two dimensional plot

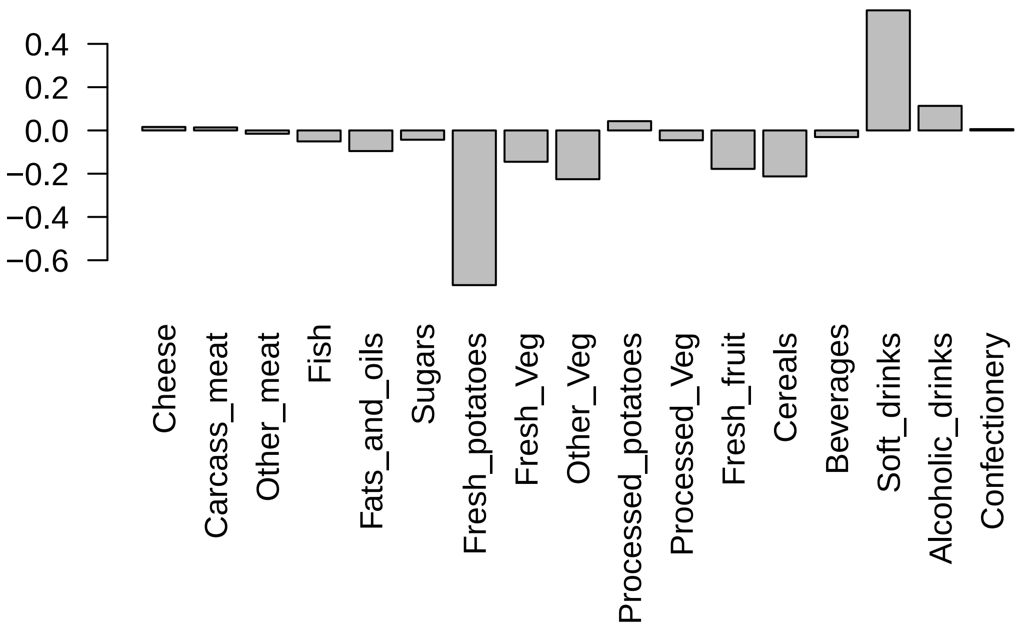
Considering the variable loading: the influence of each variable upon the principle components

```r
## biplot(): 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?

```r
## biplot() for PC2
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```

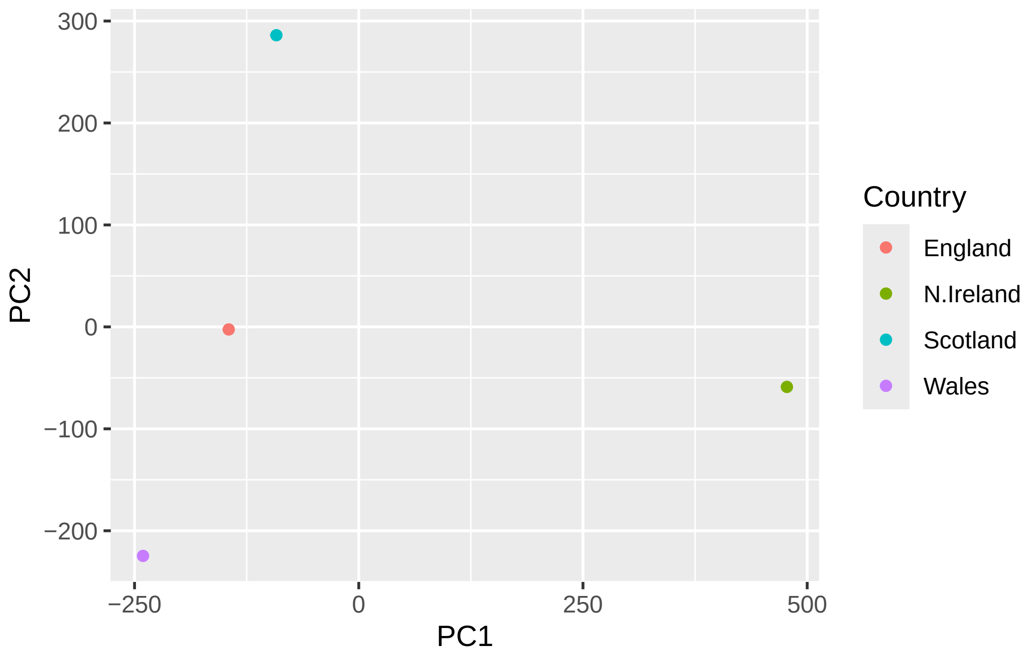


Fresh potatoes and soft drinks are featured prominantely in PC2. It tells us that soft drinks, which has the highest positive score, pushed Scotland to the top of the plot. Fresh potatoes, which has high negative score, pushed countries to the lower side of the plot.

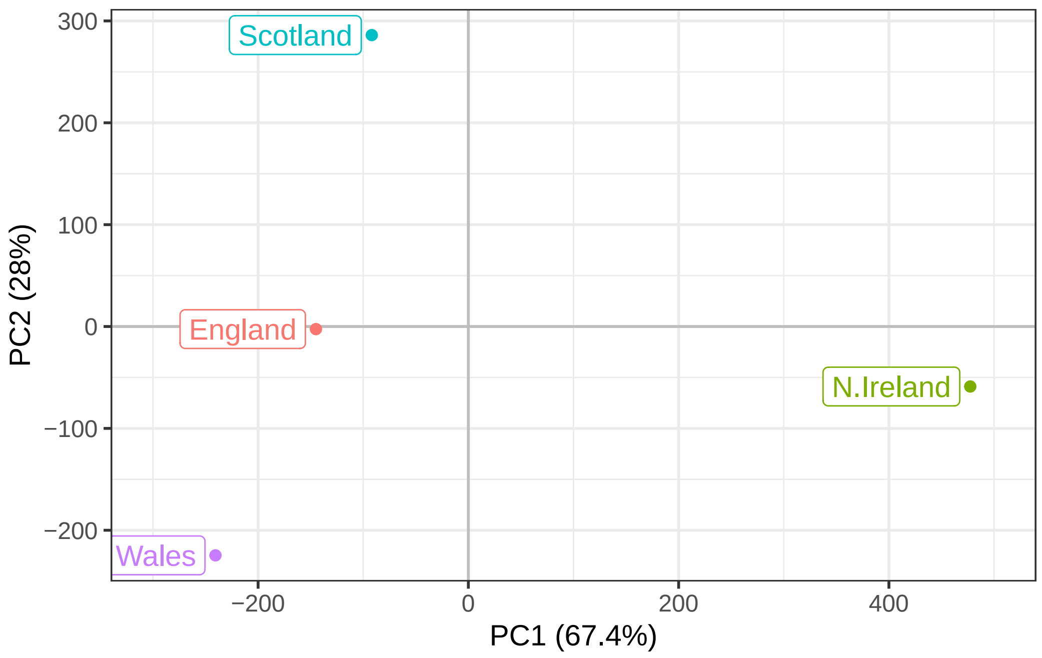
Using ggplot for these figures

we will need to take whatever it is we want to plot and convert it to a data.frame with the as.data.frame() function. Then to make our plotting life easier we will also add the country labels as a column (called "Country") to this data frame with the rownames_to_column() function from the tibble package (you might need to install this):

```r
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()
```

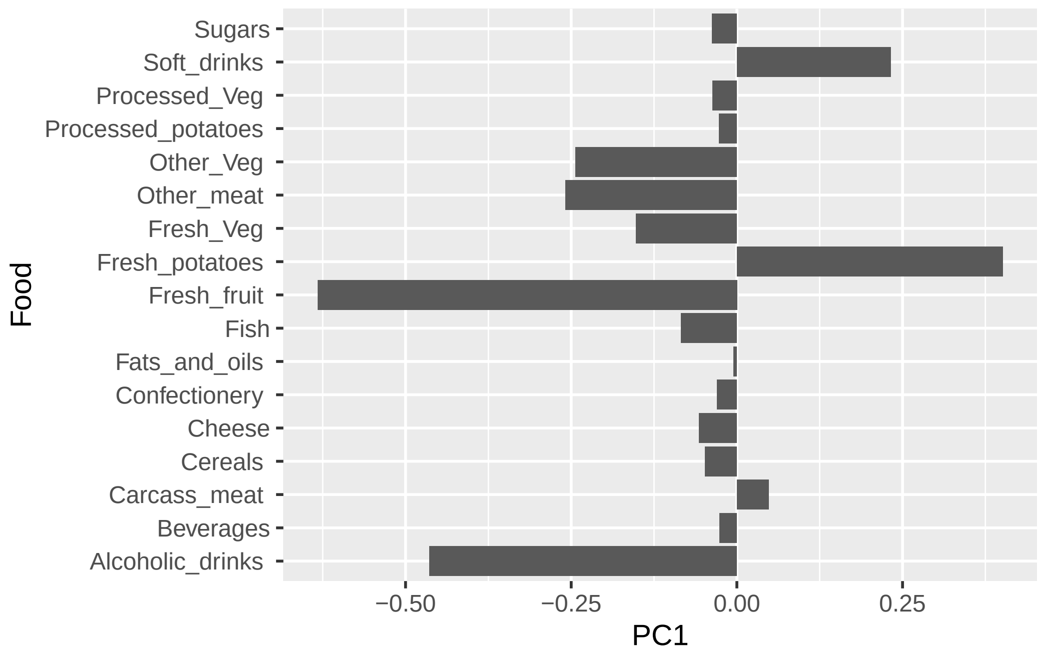


```
#much better looking plot
ggplot(df_lab) +
  aes(PC1, PC2, col=Country, label=Country) +
  geom_hline(yintercept = 0, col="gray") +
  geom_vline(xintercept = 0, col="gray") +
  geom_point(show.legend = FALSE) +
  geom_label(hjust=1, nudge_x = -10, show.legend = FALSE) +
  expand_limits(x = c(-300,500))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



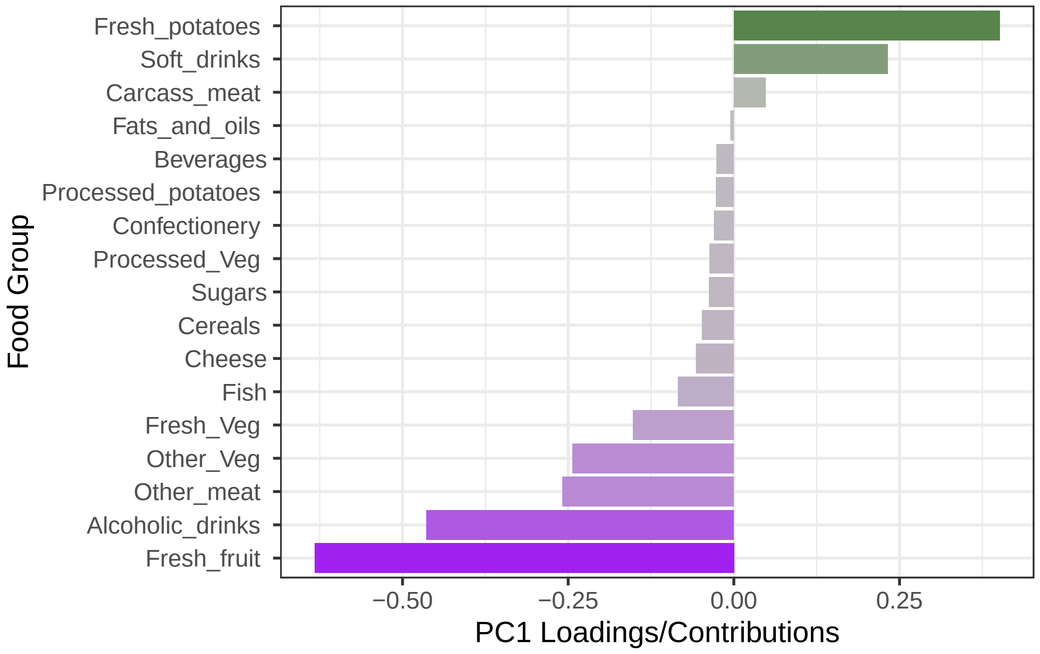
plot loadings/PC contributions figures. This data is stored in the pca$rotation object that we convert to a data frame, add the useful row names as a new column and then plot and customize with additional ggplot layers.

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

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



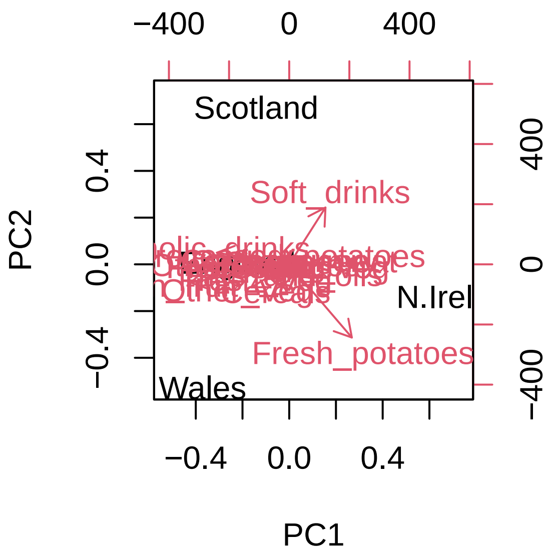
We can now add some additional features to the plot, such as reordering the y axis by the PC1 loadings and selecting a color scale and our prefered theme layer.

```
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()
```



Another way to see this information together with the main PCA plot is in a so-called biplot:

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



PCA of RNA-seq data

```r
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
```

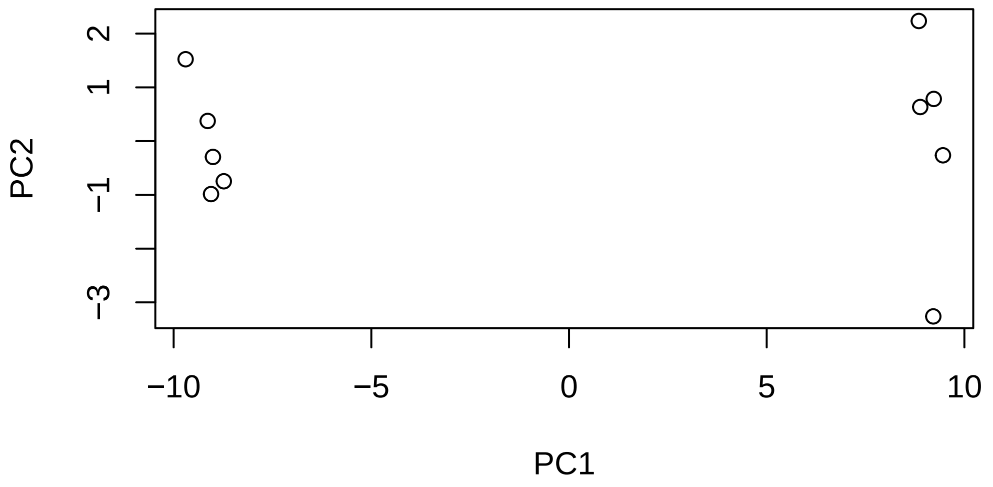
Q10: How many genes and samples are in this data set?

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

```
[1] 100  10
```

There are 100 genes and 10 samples

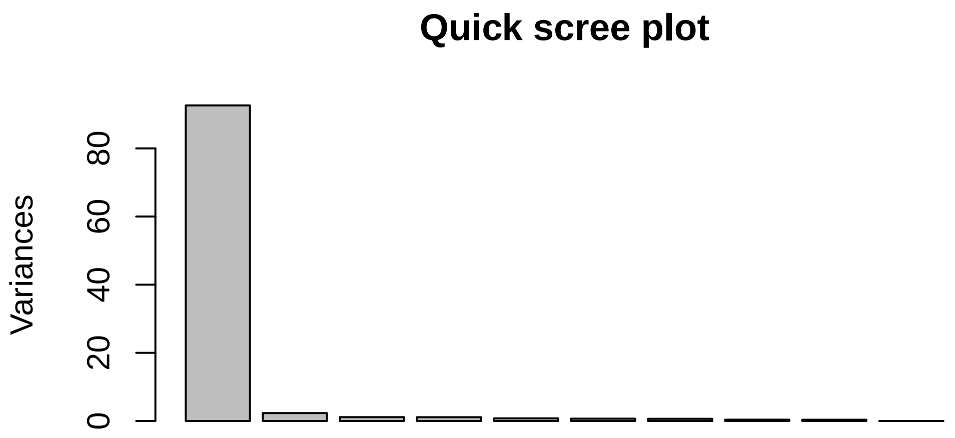
```r
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

## Simple un polished plot of pc1 and pc2
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 3.345e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

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

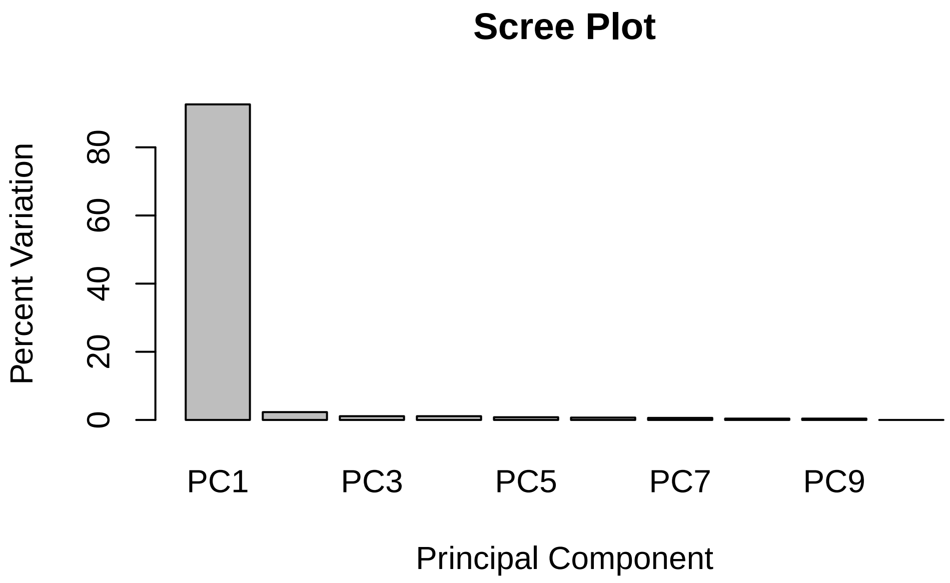


Let's make the above scree plot ourselves and in so doing explore the object returned from prcomp() a little further. We can use the square of pca$sdev, which stands for "standard deviation", to calculate how much variation in the original data each PC accounts for:

```
## Variance captured per PC
pca.var <- pca$sdev^2

## Percent variance is often more informative to look at
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.var.per
```

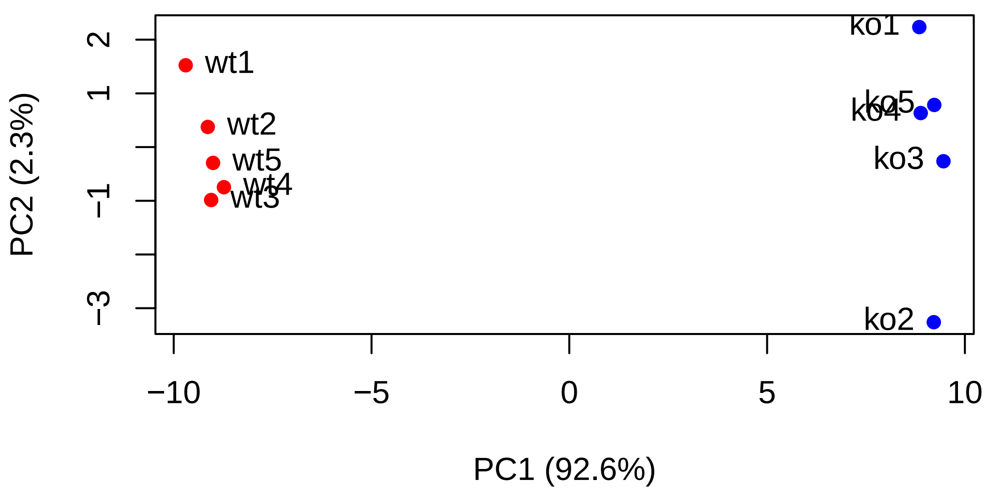
```
 [1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

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



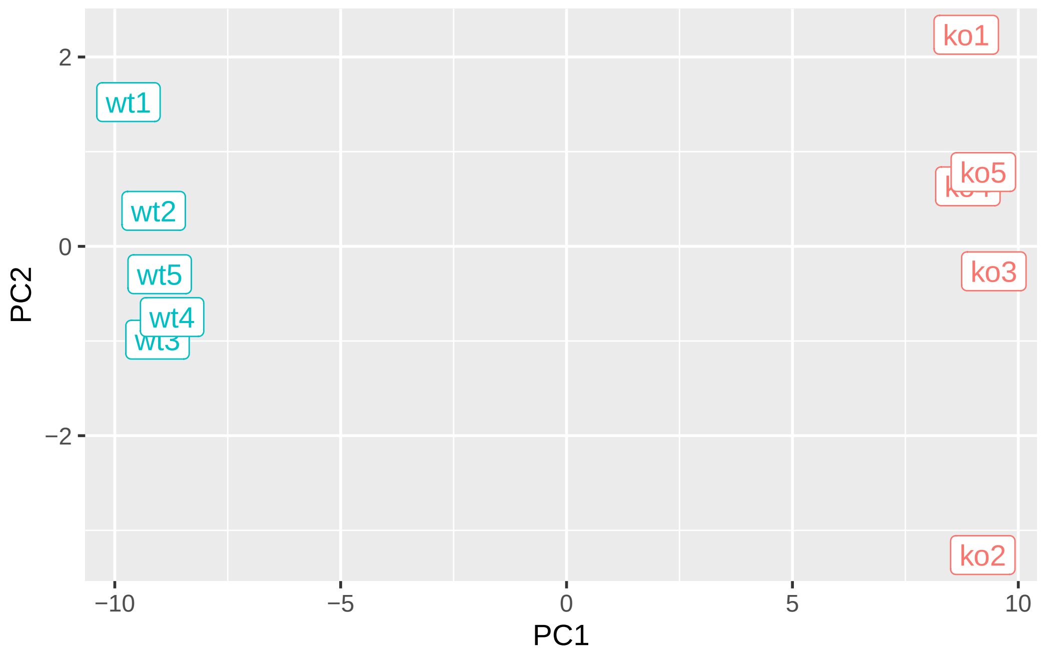
Making a more useful PCA plot

```
## A vector of colors for wt and ko samples
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)))
```



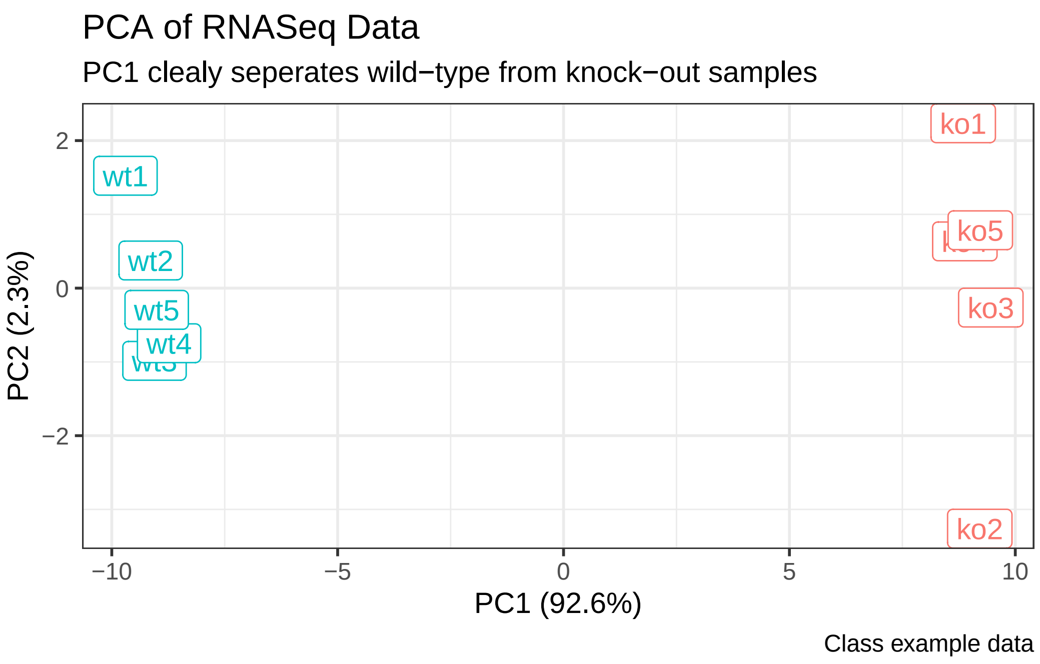
making a ggplot:

```
# Add a 'wt' and 'ko' "condition" column
df <- as.data.frame(pca$x)
df$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
```



Add some details

```
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
PC2 (2.3%)
PC1 (92.6%)
Class example data
wt1
wt2
wt5
wt4
wt3
ko1
ko5
ko4
ko3
ko2
2
0
−2
−10
−5
0
5
10