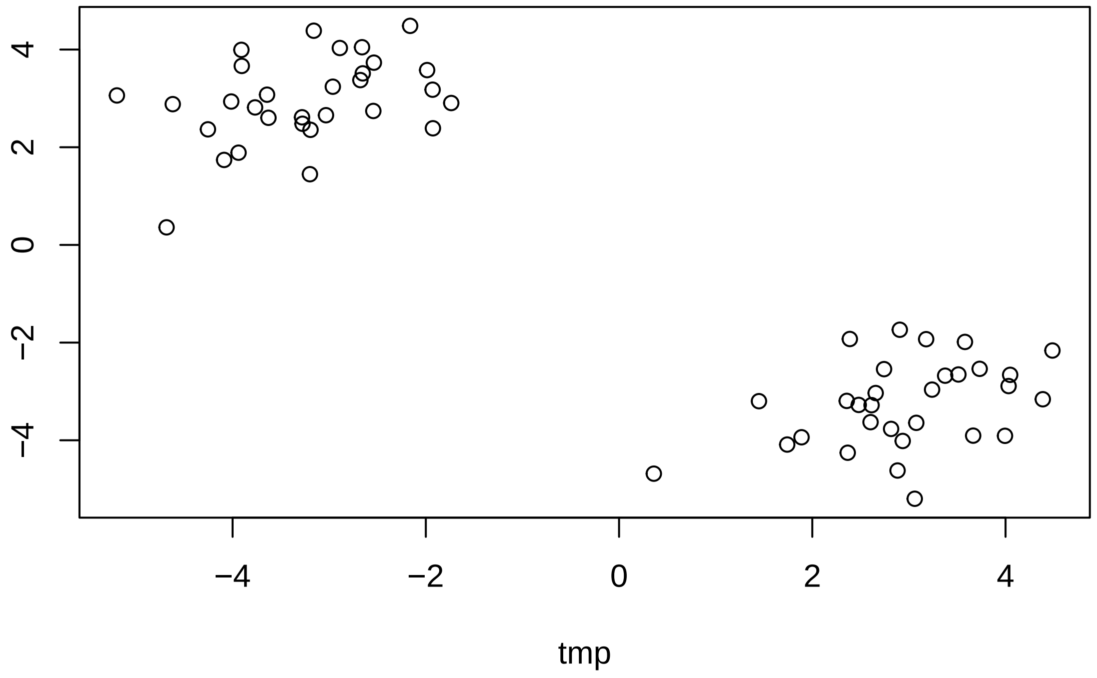
Lab07:Machine learning and PCA

Morgan Farrell

2/8/2022

##Clustering with kmeans() and hclust() We will begin by making up some data to cluster

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



Now we will run 'kmeans()'

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

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

```
##         tmp
## 1 -3.248552  2.951533
## 2  2.951533 -3.248552
##
## 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 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] 46.03384 46.03384
##  (between_SS / total_SS =  92.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

What size are each cluster?

Readout gives us 2 clusters each a size of 30, this makes sense because we told R to make us 2 clusters of 30

```
k$size
```

```
## [1] 30 30
```

cluster centers?

```
k$centers
```

```
##         tmp
## 1 -3.248552  2.951533
## 2  2.951533 -3.248552
```

Clustering vector?

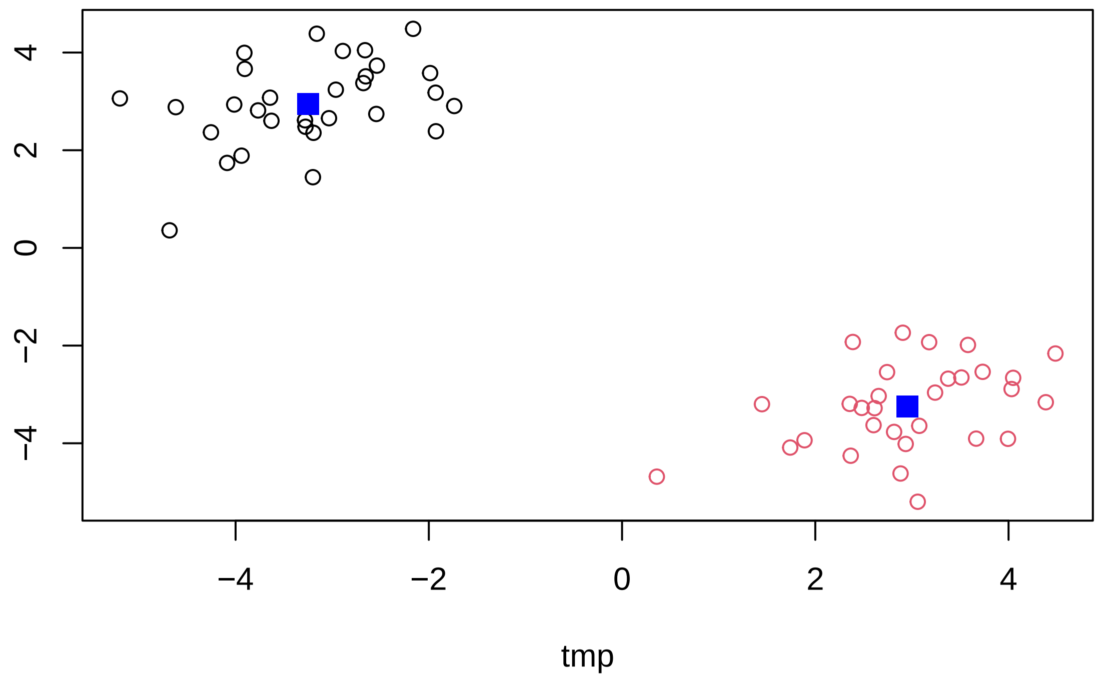
```
k$cluster
```

```
##  [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 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
```

Clustering vector, indicates which values cluster in group 1 or group 2. The first 30 to 1 and the second 30 to 2

Plot our data with the clustering result

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



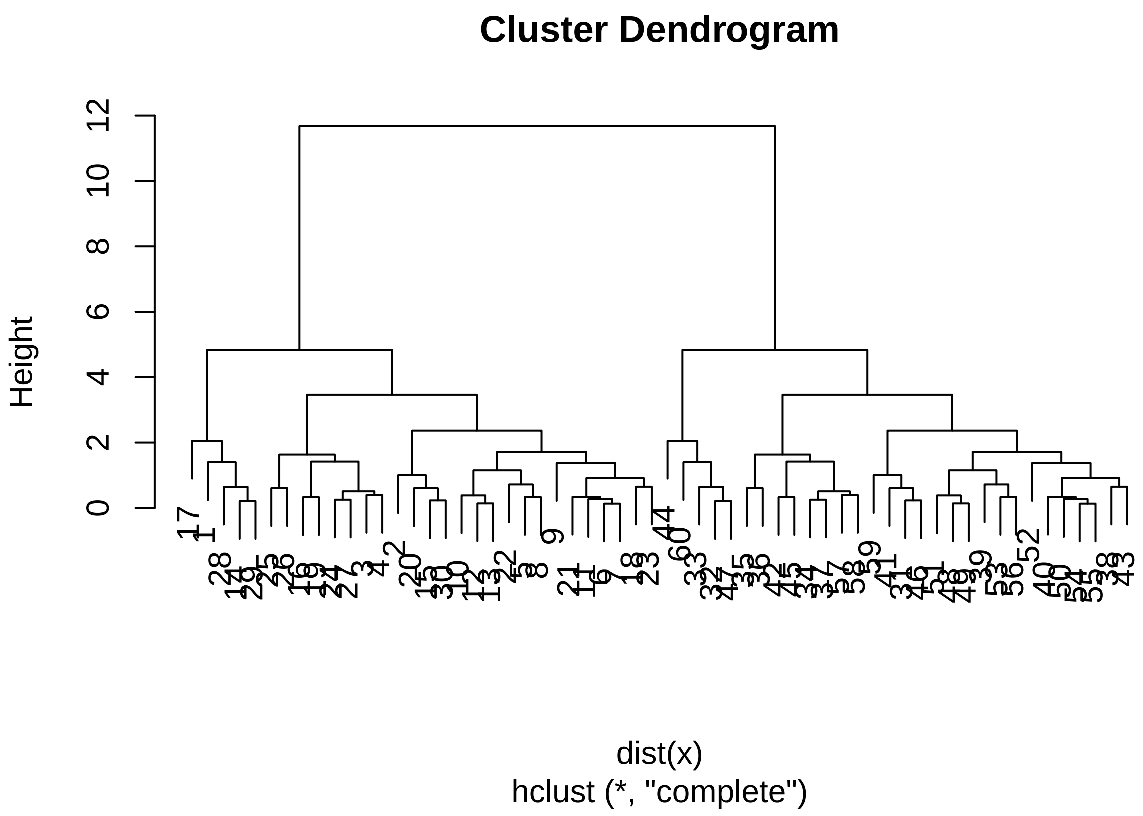
## Hierarchical clustering 'hclust()'

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

```
##
## Call:
## hclust(d = dist(x))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 60
```

Plot method for hclust()

```r
plot(hc)
```



The two groups show that the left has values 1-30 and the right have 31-60 again representing our two groups.

##Principal Component Analysis Data Practice Import data of UK foods

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

```
dim(x)
```

```
## [1] 17  5
```

Check your data by clicking on the x variable under data to see your data as a table or you can use 'head()' to preview the beginning

```
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
```

Need to fix the rownames to be the names not the numbers

```
# 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
```

Alternative row names approach to better control the table when importing

```
x <- read.csv(url, row.names=1)
head(x)
```

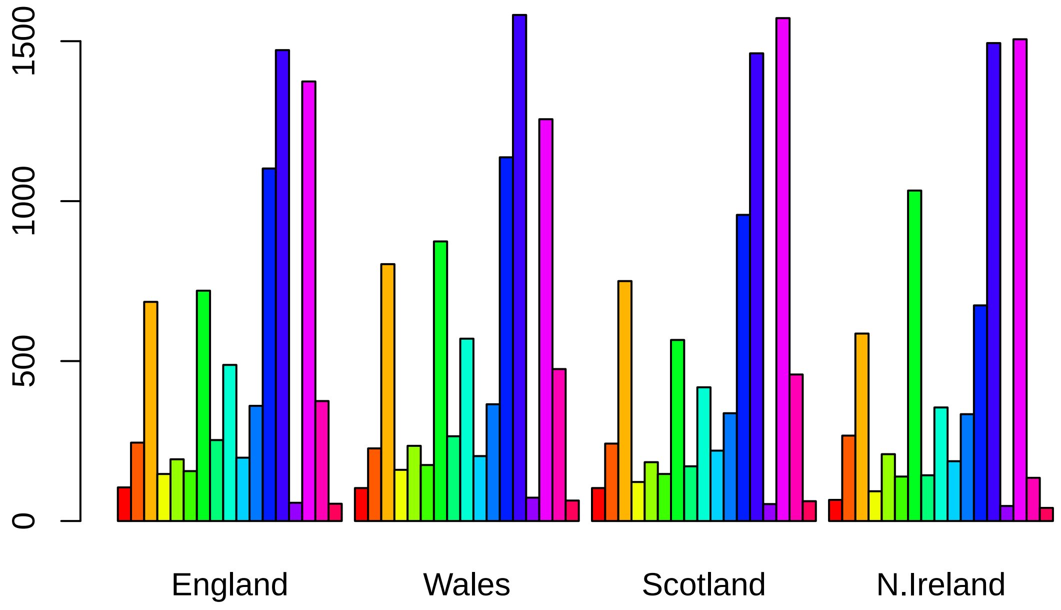
```
##                England Wales Scotland N.Ireland
## Cheese              105   103      103        66
## Carcass_meat        245   227      242       267
## Other_meat          685   803      750       586
## Fish                147   160      122        93
## Fats_and_oils       193   235      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've always used read.csv because you have the most control over your row names and column names

Next, visualize the data using a regular barplot

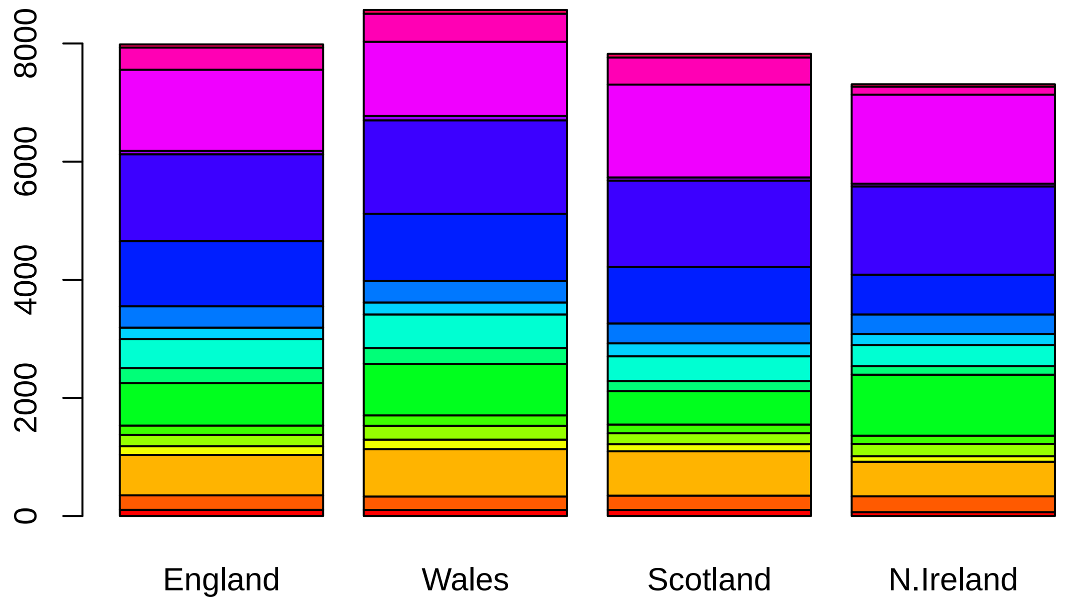
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

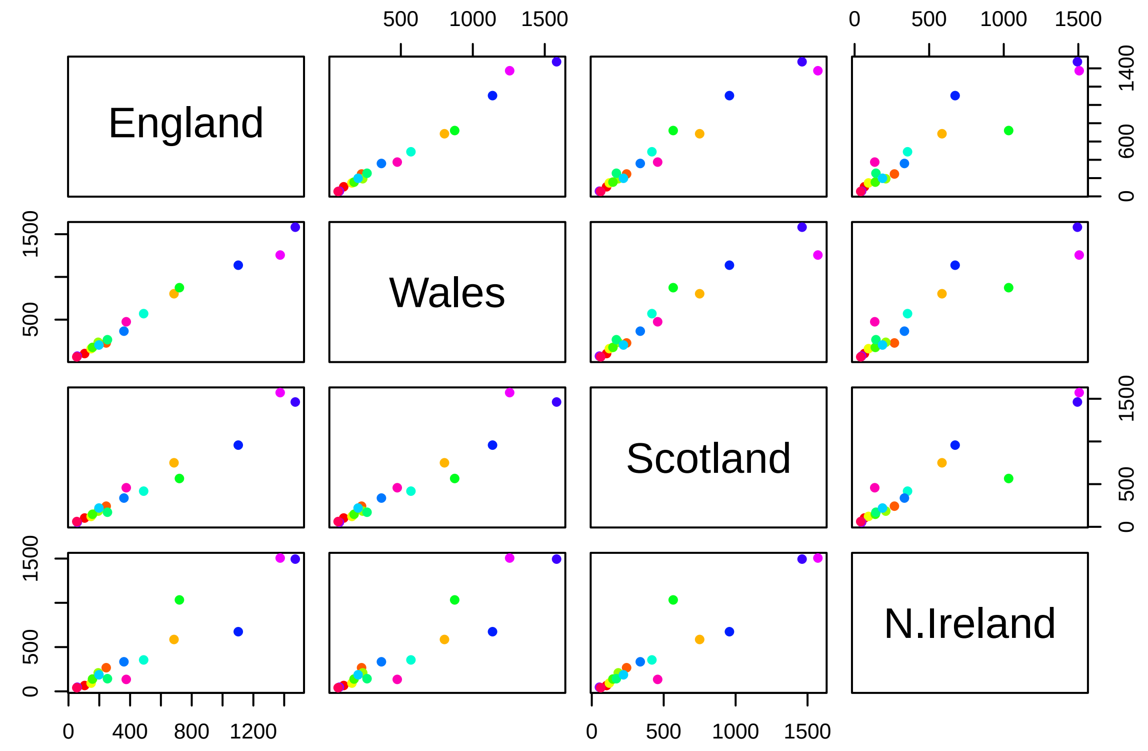
First look at the help page for barplot() Set the color to 'nrow()' giving you a color for each of your food categories If you set beside= TRUE it will break up the row categories to put them side-by-side vs stacked

```
?barplot()
mycols <- rainbow(nrow(x)) #set the rows to different colors of the rainbow store as a vector for that
barplot(as.matrix(x), col=mycols)#use your color vector
```



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=mycols, pch=16)
```



The countries are written down the diagonal The first plot is England vs Wales Each plot are comparing two countries if they have similar values you would see a straight line indicating there is no difference or the value is equal

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

N. Ireland and the other countries have a lot of differences in food consumption since they do not have graphs with straight lines. For example, the blue food group is consumed more in England, but the green food group is consumed more in Ireland.

PCA to Interpret Data

Next we will use PCA to interpret this data and see if it is more clear to figure out the trends

We are going to use the 'prcom()' function which expects the observations to be rows and the variables to be columns. So we need to transpose the data.

```
t(x) #t() can transpose the data frame to fit the expectations of the prcomp()
```

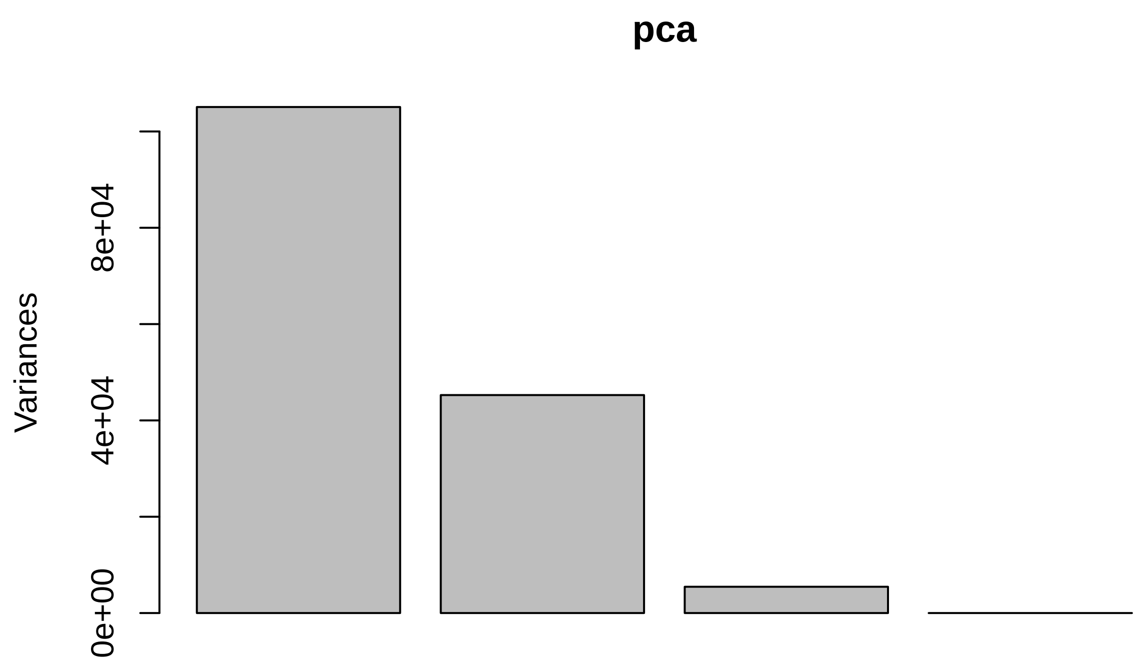
```
##             Cheese Carcass_meat  Other_meat Fish Fats_and_oils  Sugars
## England        105          245         685  147           193     156
## Wales          103          227         803  160           235     175
## Scotland       103          242         750  122           184     147
## N.Ireland       66          267         586   93           209     139
##           Fresh_potatoes  Fresh_Veg  Other_Veg  Processed_potatoes
## England              720        253        488                 198
## Wales                874        265        570                 203
## Scotland             566        171        418                 220
## N.Ireland           1033        143        355                 187
##           Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
## England             360         1102     1472         57        1374
## Wales               365         1137     1582         73        1256
## Scotland            337          957     1462         53        1572
## N.Ireland           334          674     1494         47        1506
##           Alcoholic_drinks  Confectionery
## England                375             54
## Wales                  475             64
## Scotland               458             62
## N.Ireland              135             41
```

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

Notice PC1 will describe 67.44% of the variance of this data set. The PC2 is 29% of data. PC1 +PC2= 96.5% explain most of the variance of the data

```
plot(pca) #Only plots % of explained variance
```



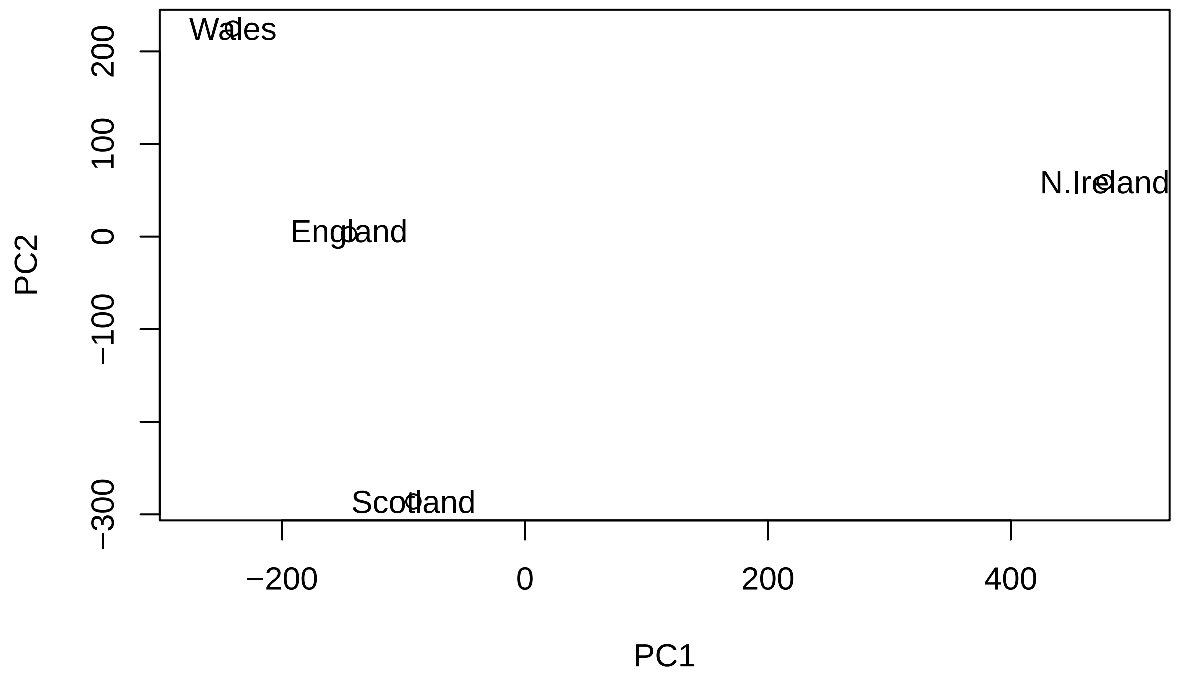
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
attributes(pca)
```

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

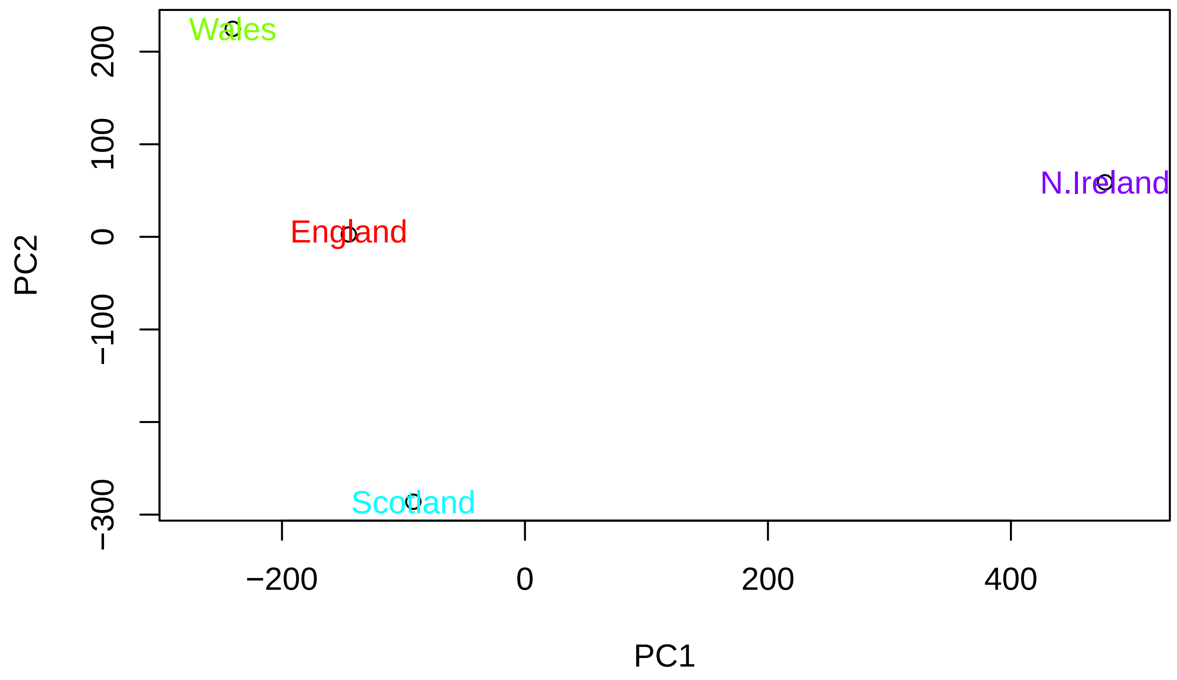
PCA plot or PCA score plot is PC1 vs PC2

```
# Plot PC1 vs PC2
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.

```
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=rainbow(4))
```



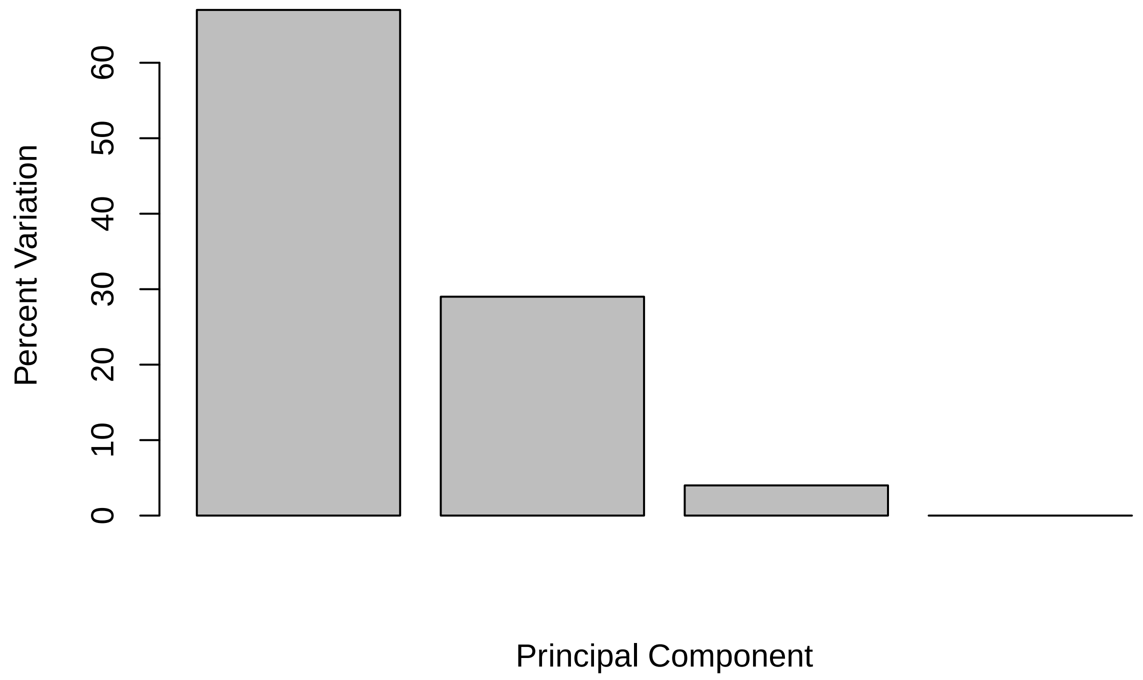
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

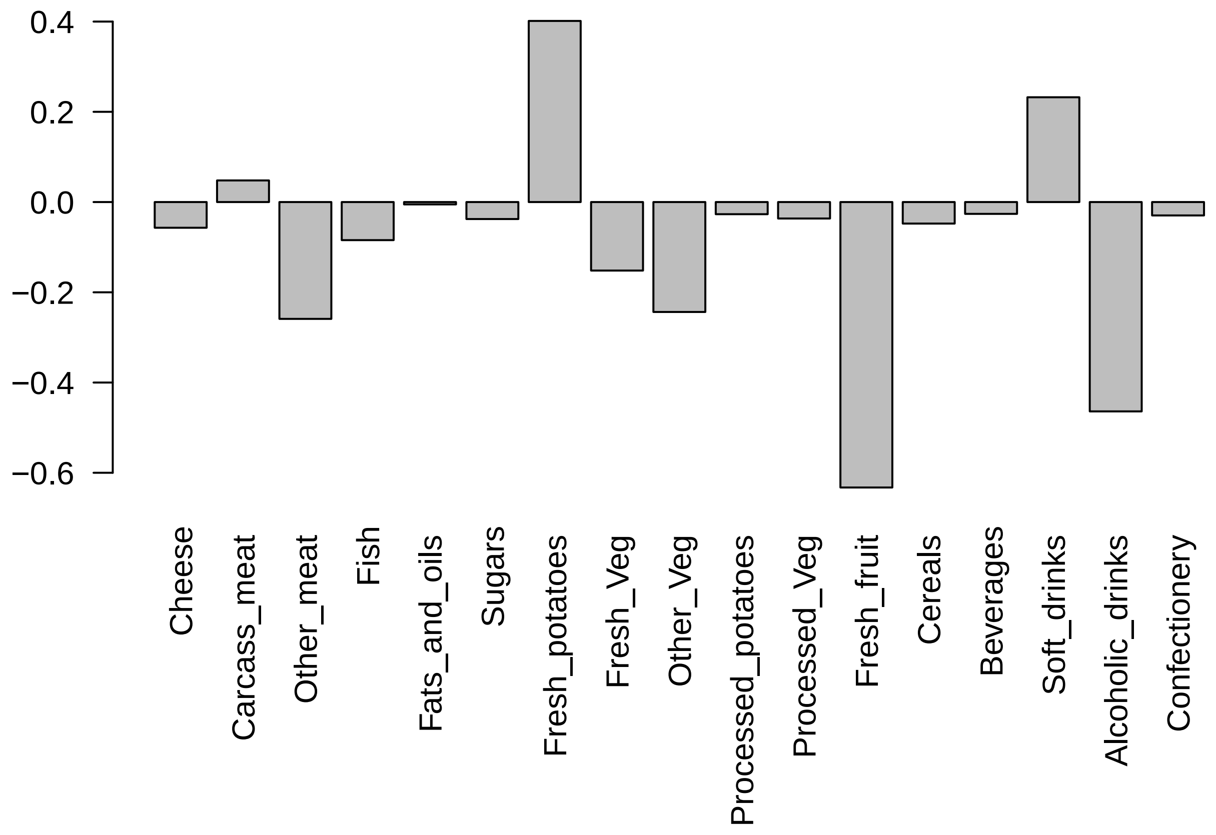
```
z <- summary(pca)
z$importance
```

```
##                              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")
```

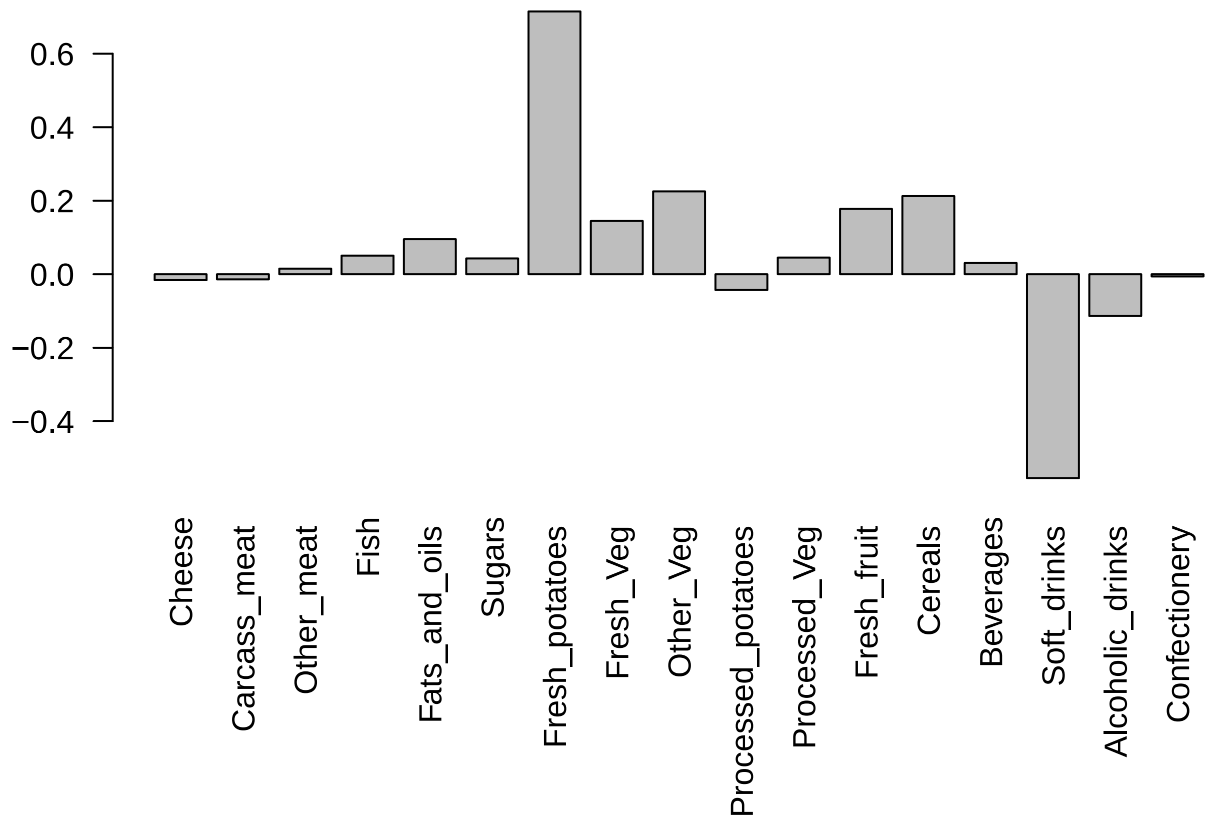


```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



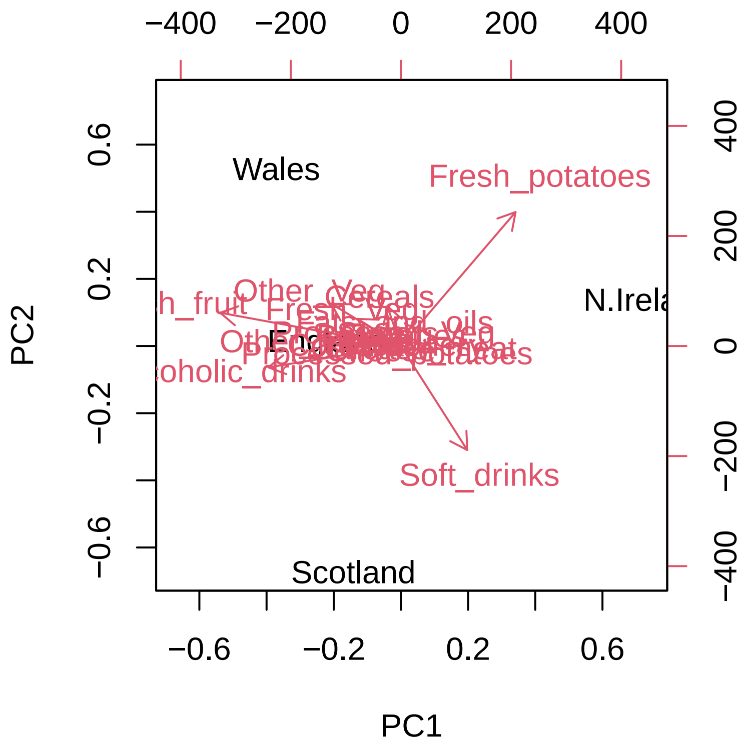
>Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

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



The PC1 vs PC2 plot show differences in several food groups mainly, Vegetables, fresh fruit, soft drinks and alcoholic beverages. These categories change the most between PC1 and PC2. Ireland eats more potatoes and drink more soft drinks. PC2 shows us that the rest of UK drink more alcohol and eat more fresh fruit.

```
## 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?

If you click on the rna.data file you can see that there are 100 genes and 5 knockouts along with 5 wild type

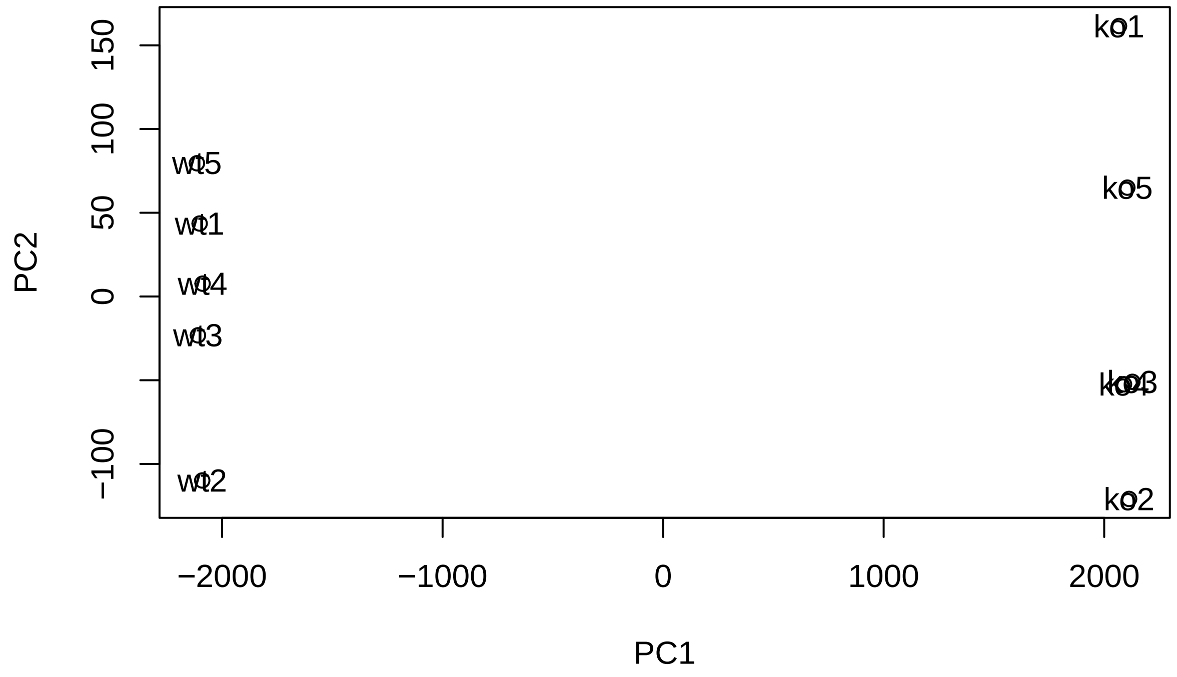
```r
pca <- prcomp(t(rna.data))
summary(pca)
```

```
## 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
```

Make a PCA score plot

```
## Simple un polished plot of pc1 and pc2
plot(pca$x[,1:2], xlab="PC1", ylab="PC2")
text(pca$x[,1:2], labels=colnames(rna.data))
```

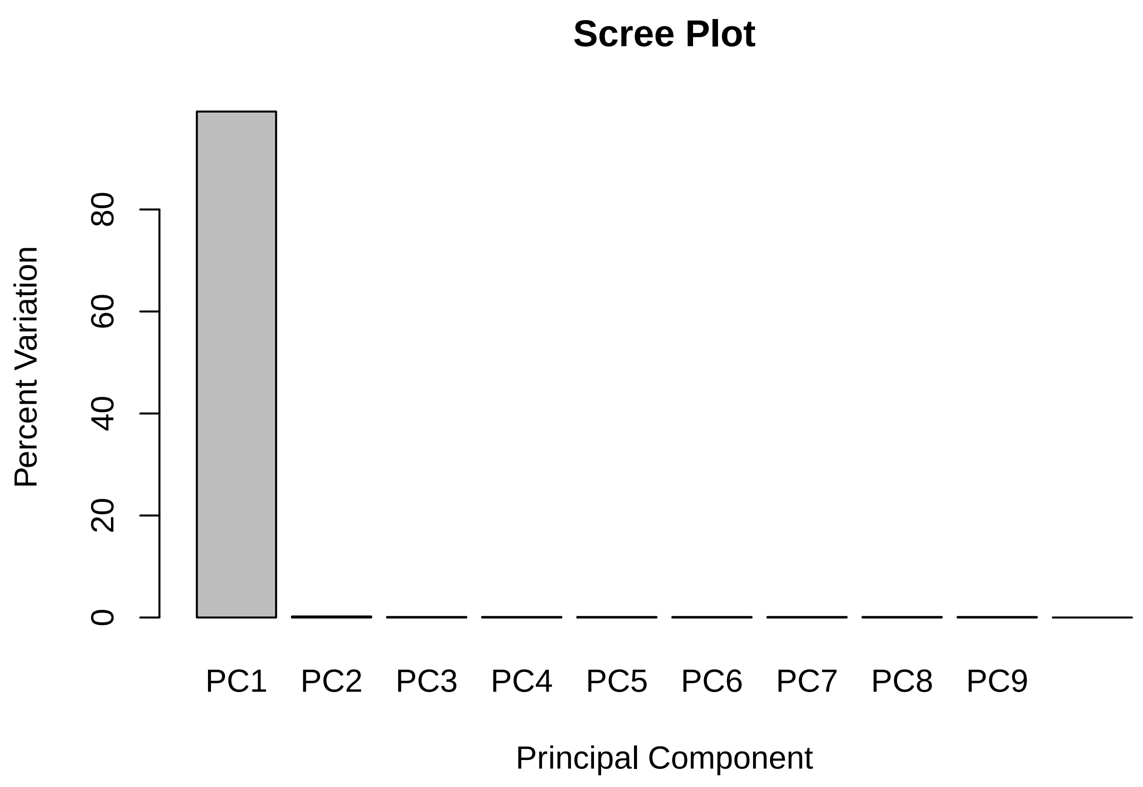


```
## Variance captured per PC
pca.var <- pca$sdev^2

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

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

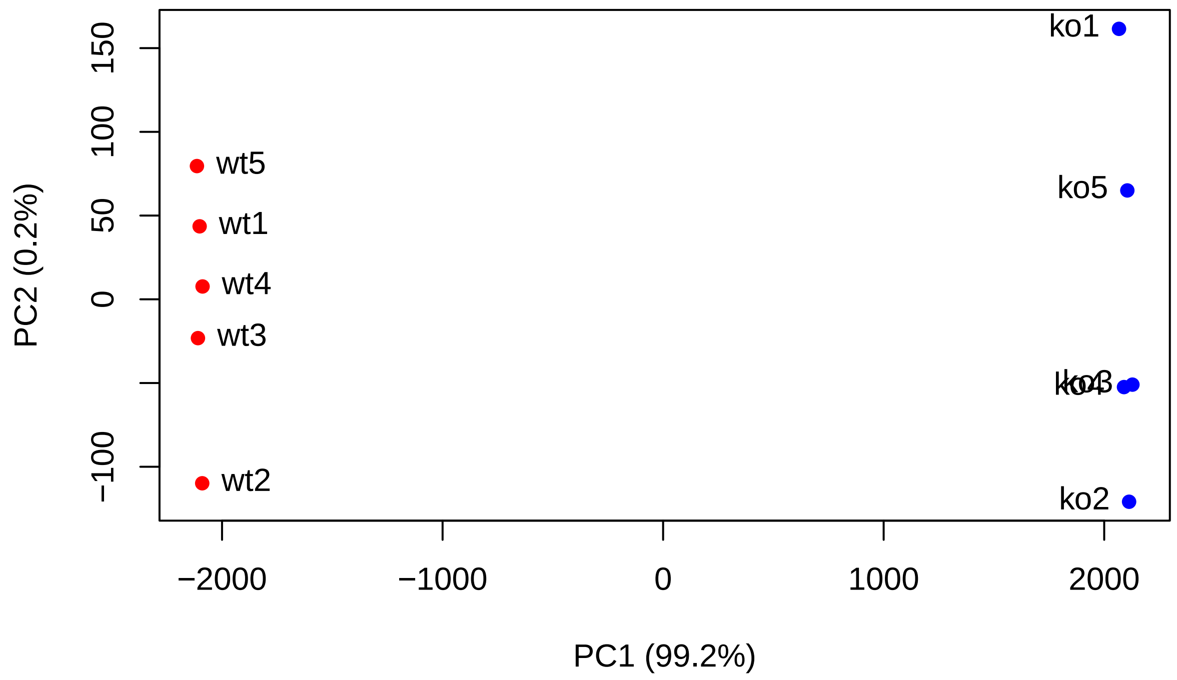
```r
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



```r
## 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)))
```

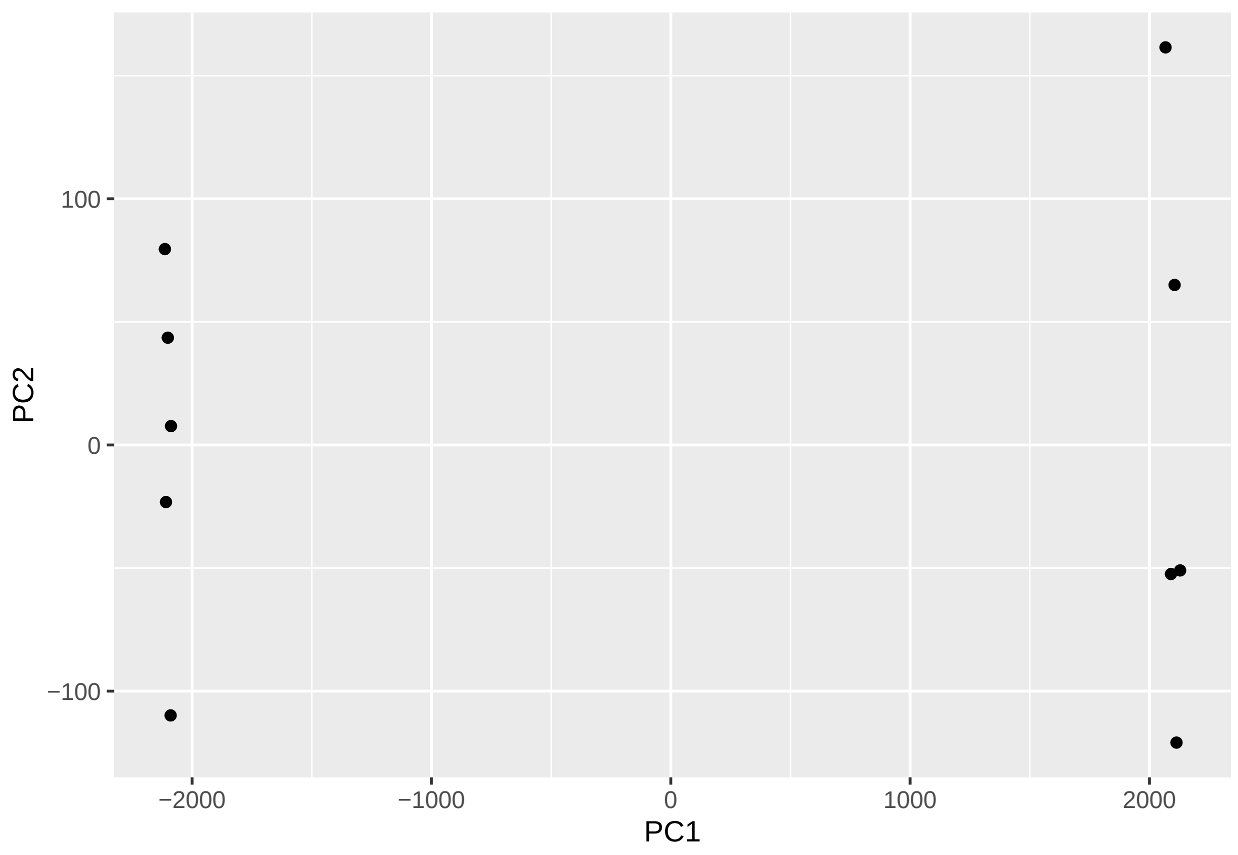


```
library(ggplot2)
```

```
## Warning in register(): Can't find generic `scale_type` in package ggplot2 to
## register S3 method.
```

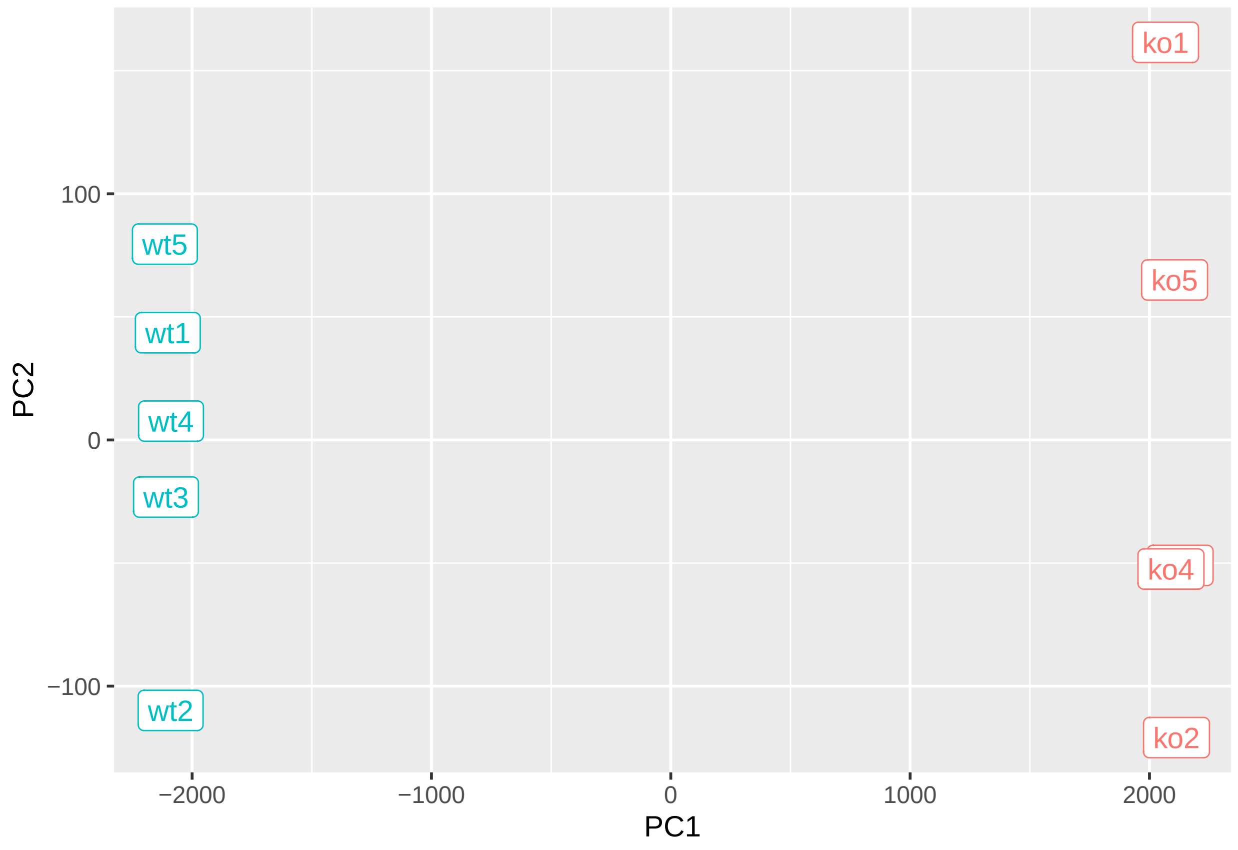
```
df <- as.data.frame(pca$x)

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

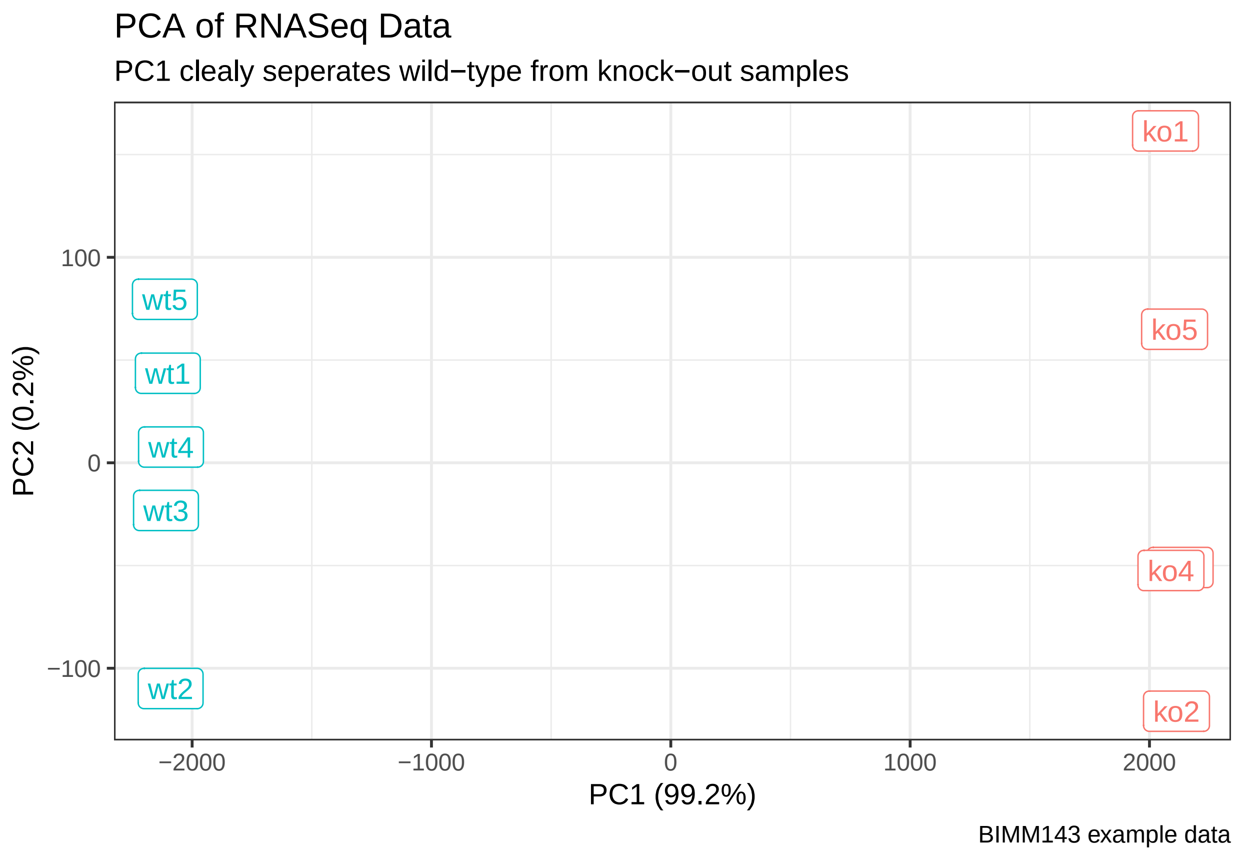


```
# Add a 'wt' and 'ko' "condition" column
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)

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="BIMM143 example data") +
     theme_bw()
```



```r
loading_scores <- pca$rotation[,1]

## Find the top 10 measurements (genes) that contribute
## most to PC1 in either direction (+ or -)
gene_scores <- abs(loading_scores)
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

## show the names of the top 10 genes
top_10_genes <- names(gene_score_ranked[1:10])
top_10_genes
```

```
##  [1] "gene98" "gene45" "gene10" "gene21" "gene48" "gene50" "gene18" "gene62"
##  [9] "gene3"  "gene60"
```