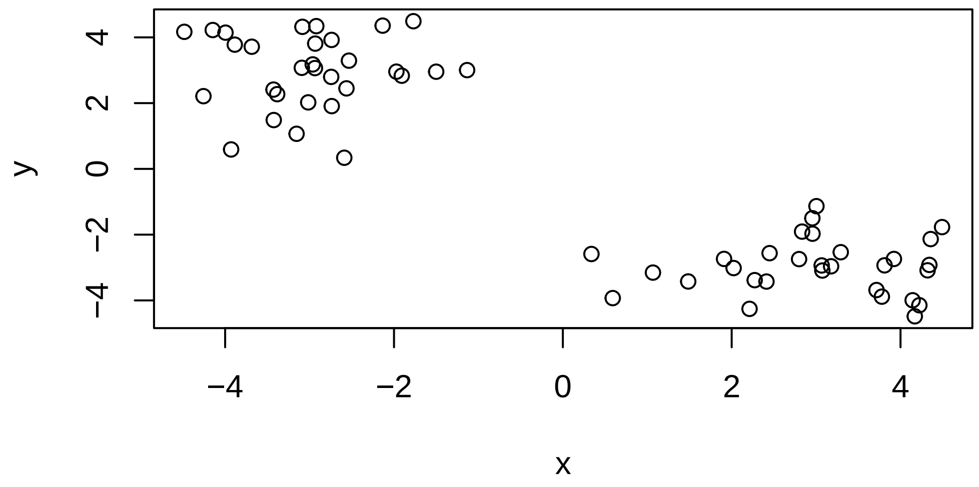
Class 7: Machine Learning

Izabelle Querubin

Example of K-means clustering

First step is to make up some data with a known structure, so we know what the answer should be.

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



Now we have some structured data in `x`. Let's see if k-means is able to identify the two groups.

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

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

Cluster means:
          x         y
1 -2.968109  2.973144
2  2.973144 -2.968109

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] 57.20792 57.20792
 (between_SS / total_SS =  90.2 %)

Available components:

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

Let's explore k:

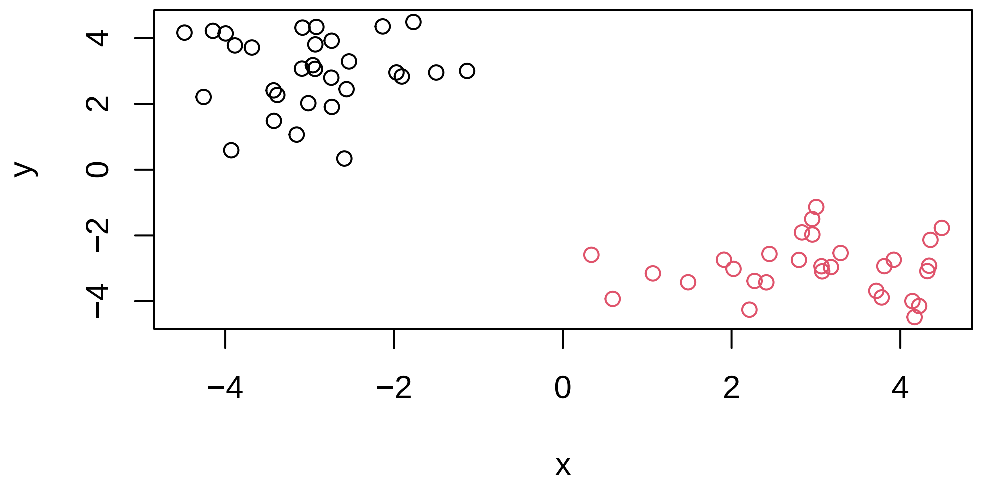
```r
k$size
```

```
[1] 30 30
```

```r
k$centers
```

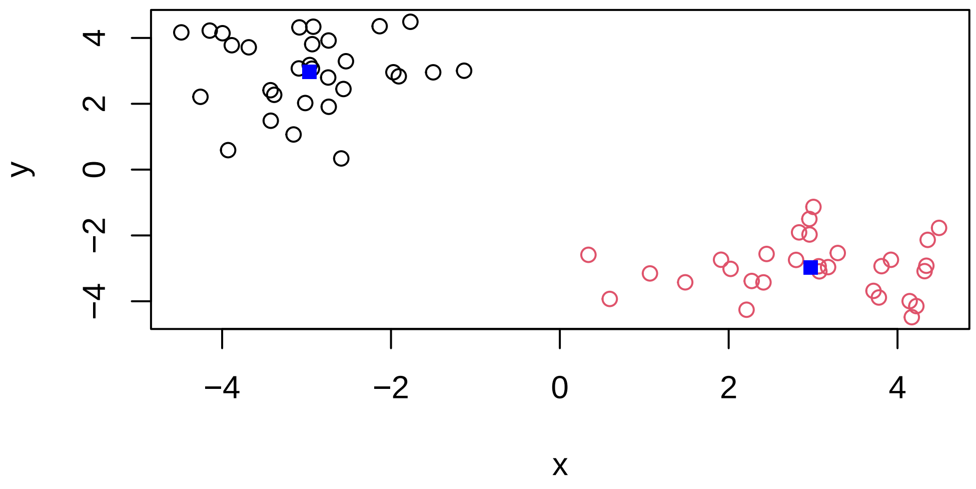
```
          x         y
1 -2.968109  2.973144
2  2.973144 -2.968109
```

```r
plot(x, col = k$cluster)
```

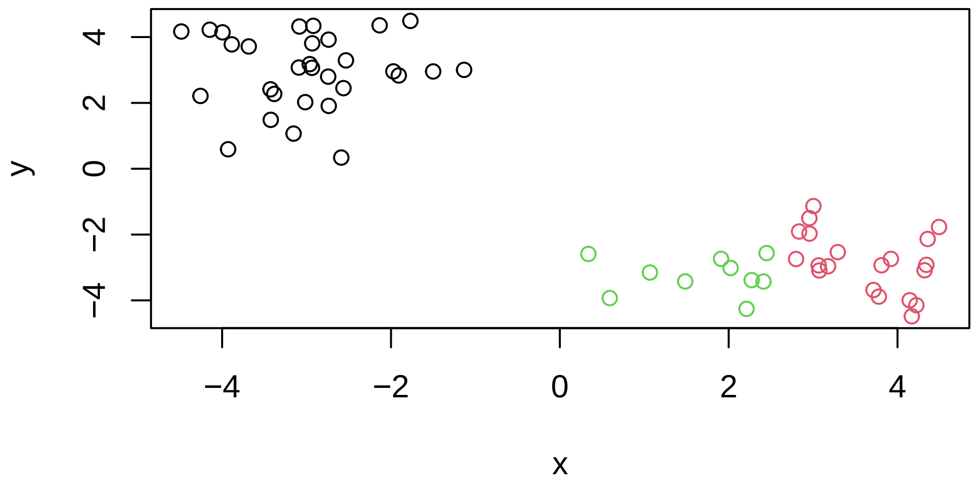


Now we can add the clusters centers:

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



```
k_3 <- kmeans(x, centers = 3, nstart = 20)
plot(x, col = k_3$cluster)
```



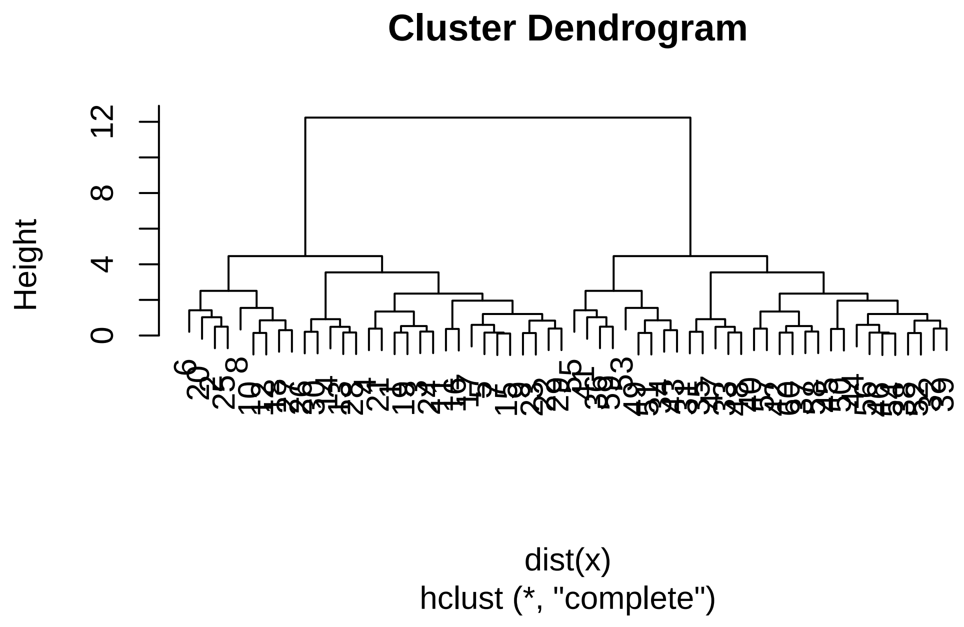
Example of Hierarchical Clustering

Let's use the same data as before, which we stored in `x`. We will use the `hclust()` function.

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

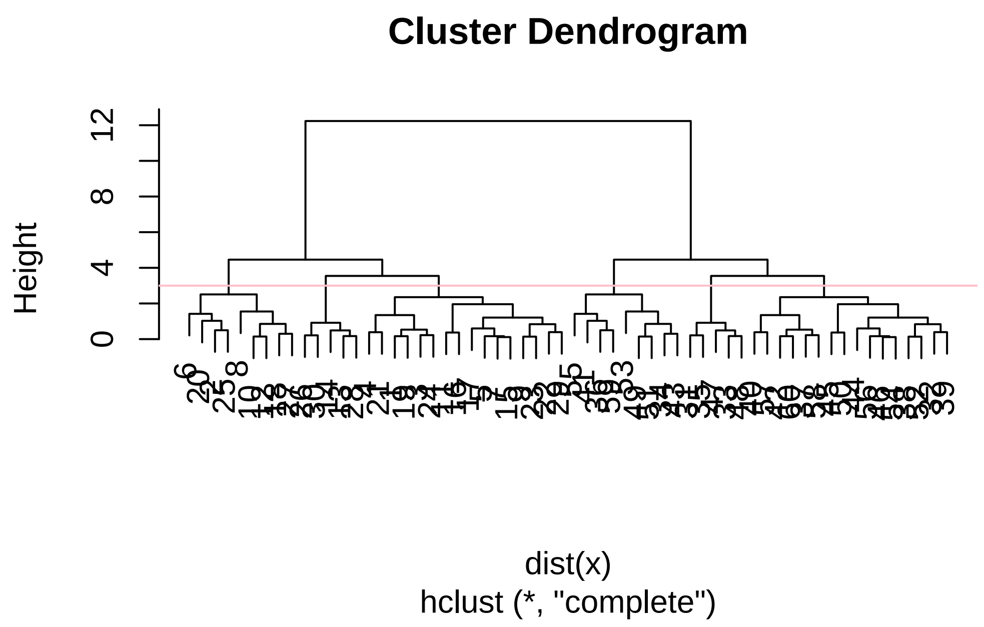
```
Call:
hclust(d = dist(x))

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

```
plot(clustering)
```



Let's add a horizontal line:

```
plot(clustering)
abline(h = 3, col = "pink")
```



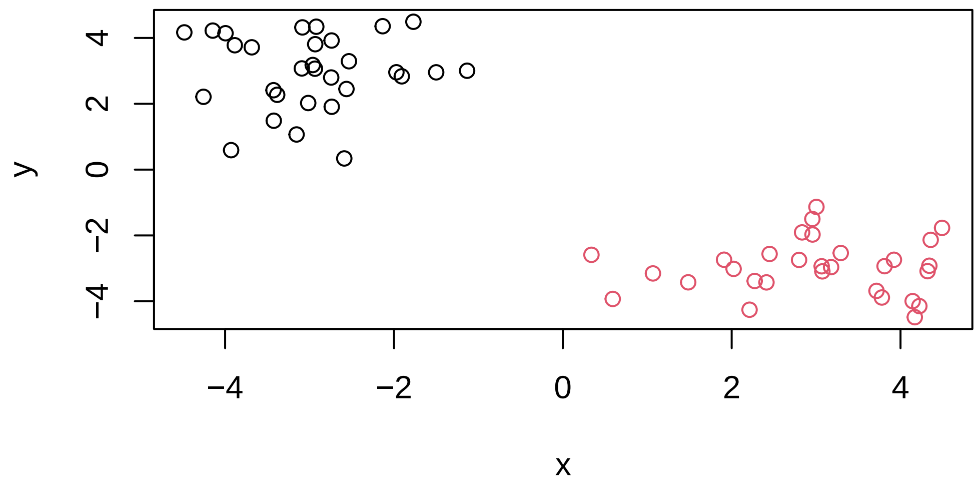
To get our results (i.e. membership vector), we need to "cut" the tree. The function for doing this is `cutree()`.

```
subgroups <- cutree(clustering, h = 10)
subgroups
```

```
 [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
```

Plotting this...

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



You can also “cut” your tree with the number of clusters you want:

```
cutree(clustering, k = 2)
```

```
 [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
```

Principal Component Analysis (PCA)

PCA of the UK Food

First was to read the data:

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

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

Rows: 17; Columns: 4

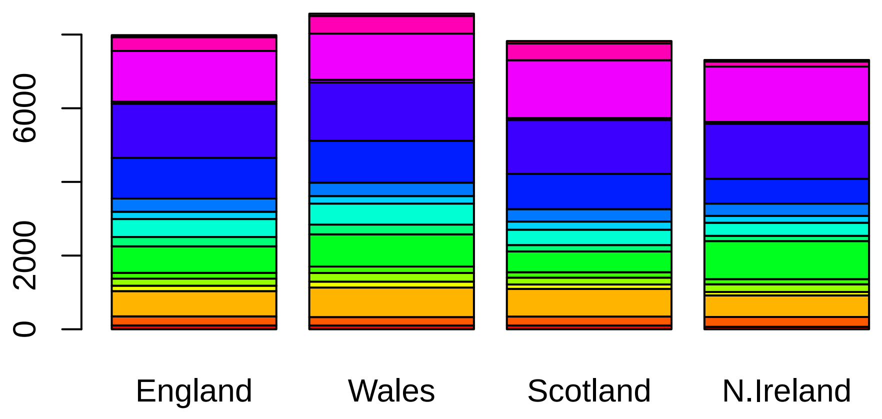
To answer this question, you could use the dim() function.

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?

Using `x <- read.csv(url, row.names=1)` is preferable and more robust since it is much simpler and more efficient.

Now we can generate some basic visualizations.

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

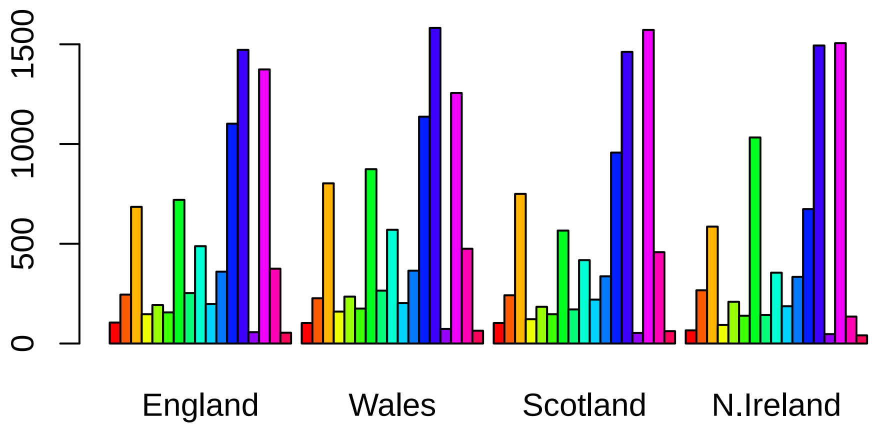


Q3. Changing what optional argument in the barplot() function results in the following plot?

`beside = TRUE` —TRUE will render the plot below, FALSE will render the plot above.

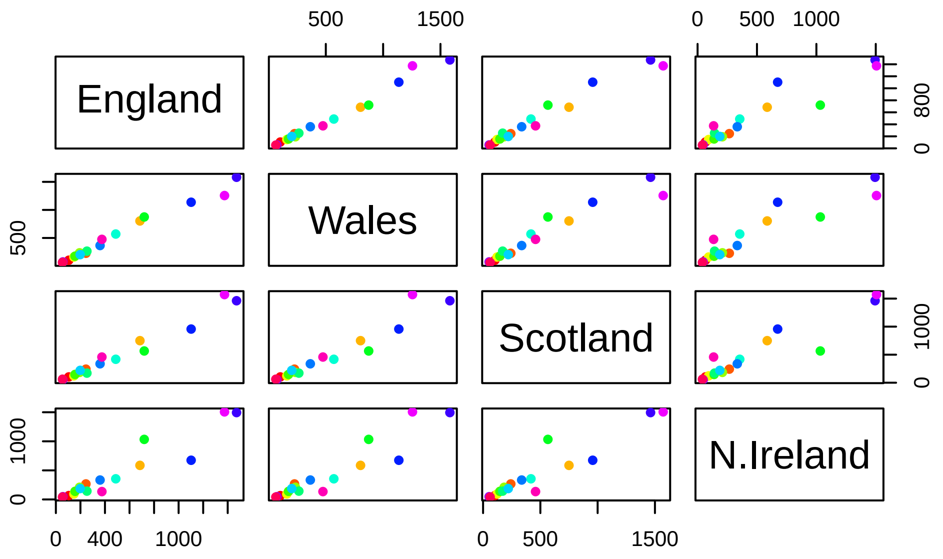
Let's refine our barplot.

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



Other visualizations that can be useful…

```r
pairs(x, col = rainbow(nrow(x)), pch = 16)
```



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?

These plots are meant to compare population means and determine how significantly different they are from one another. A point that lies on the diagonal suggests that the value of that variable is similar in both populations.

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

Most of the data is clustered towards the bottom left of the plot, indicating that N. Ireland consumes different amounts of the different foods compared to the other countries of the UK.

Let's apply PCA (principal components analysis). For that, we need to use the command `prcomp()`. This function expects the transpose of our data.

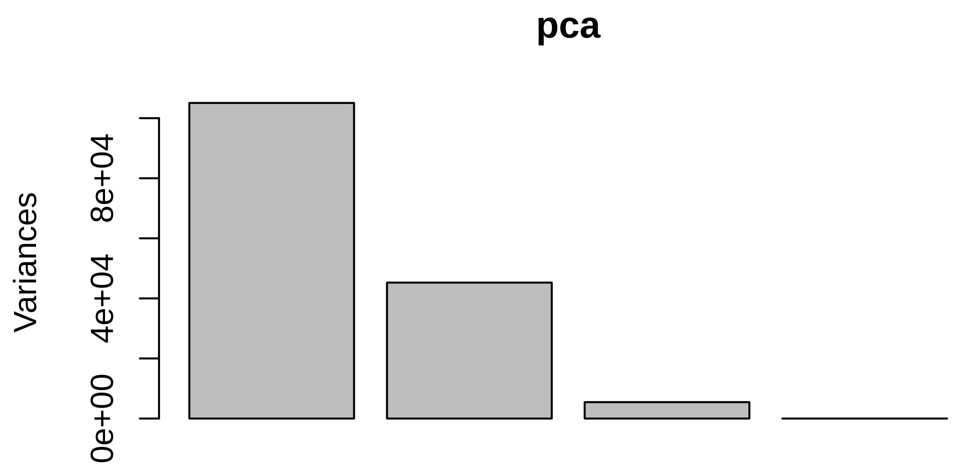
```
#transpose_matrix <-  t(s)
# pca <- prcomp(transpose_matrix)

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

```r
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
```

Let's plot the PCA results!

```r
plot(pca)
```



We need to access the results of the PCA analysis.

```r
attributes(pca)
```

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

$class
[1] "prcomp"
```

We can explore the pca$x dataframe:

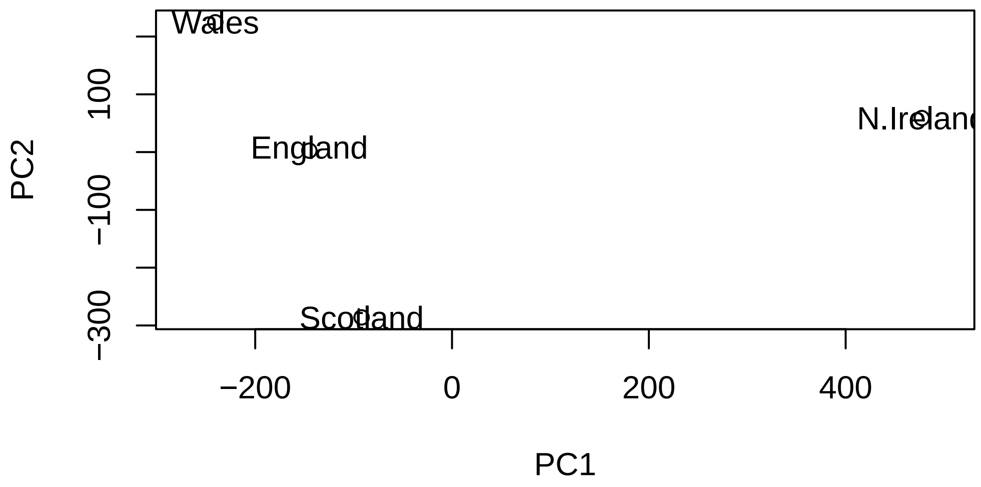
```
pca$x
```

```
                PC1         PC2         PC3           PC4
England  -144.99315    2.532999 -105.768945  2.842865e-14
Wales    -240.52915  224.646925   56.475555  7.804382e-13
Scotland  -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland 477.39164   58.901862    4.877895  1.448078e-13
```

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

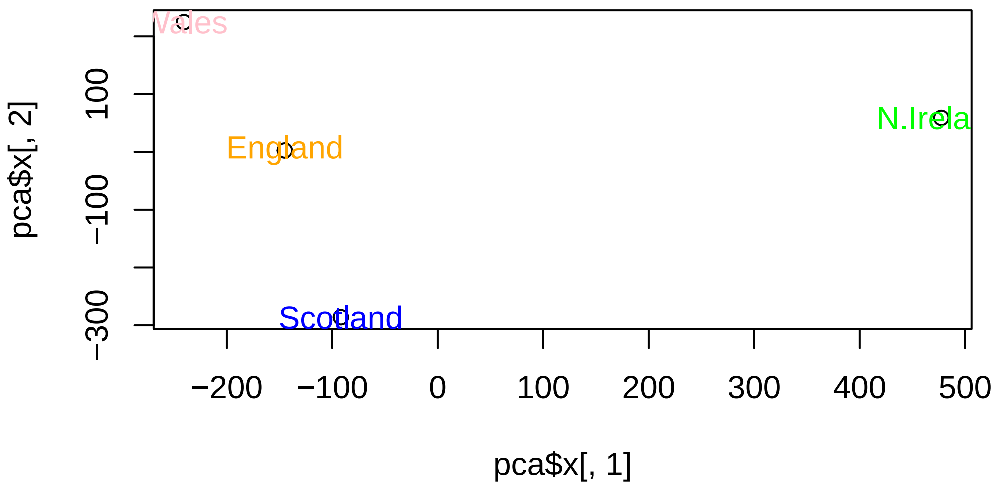
Plotting:

```
plot(x = pca$x[, 1], y = pca$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 the start of this document.

```r
plot(x = pca$x[, 1], y = pca$x[, 2])
colors_countries <- c('orange', 'pink', 'blue', 'green')
text(x = pca$x[, 1], y = pca$x[, 2], colnames(x), col = colors_countries)
```



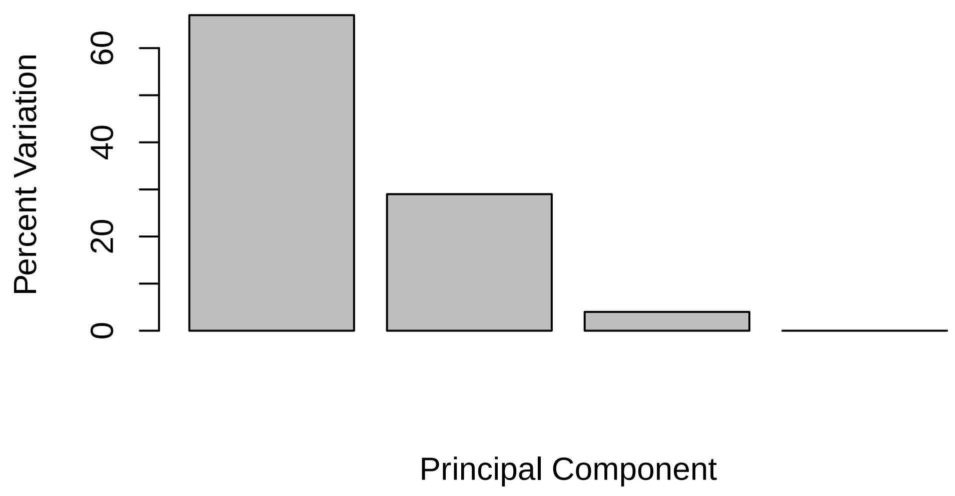
```r
v <- round(pca$sdev^2/sum(pca$sdev^2) * 100)
v
```

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

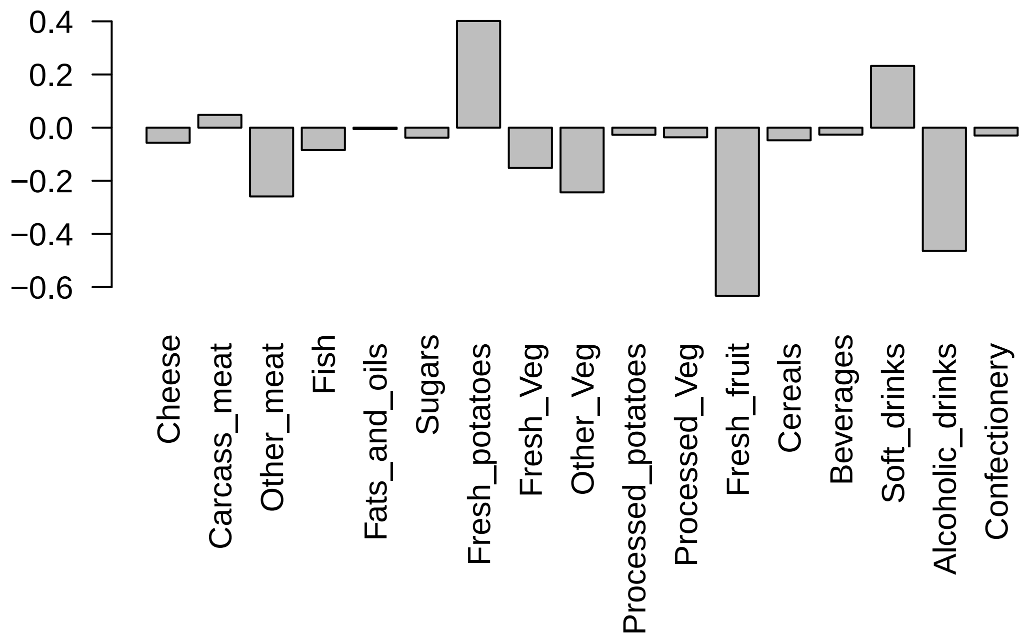
```r
## or the second row here...
z <- summary(pca)
z$importance
```

```
                             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
```

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

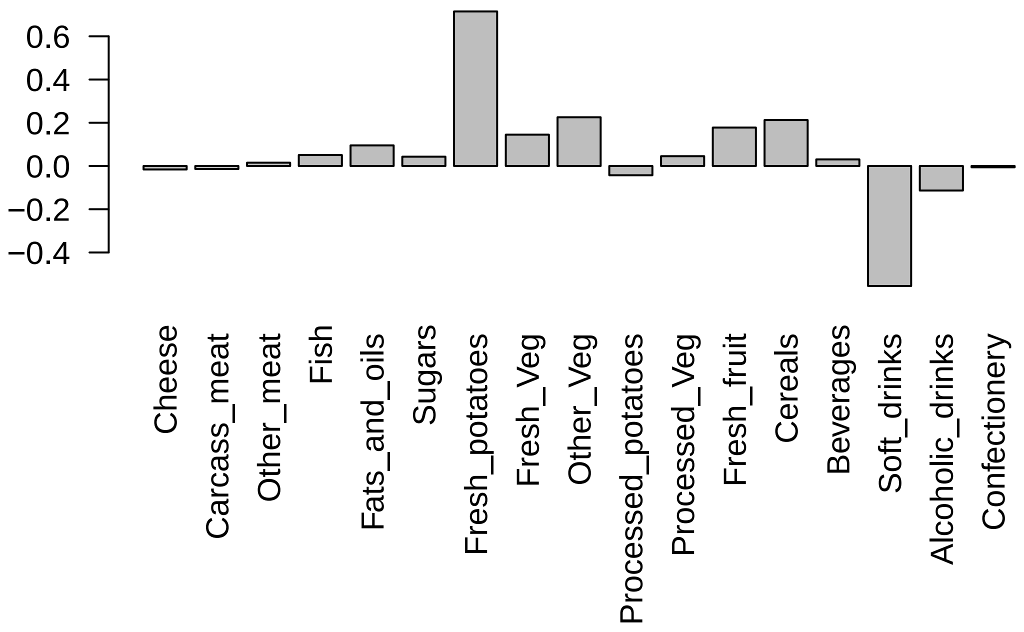


```r
## Let's focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot(pca$rotation[,1], las = 2)
```



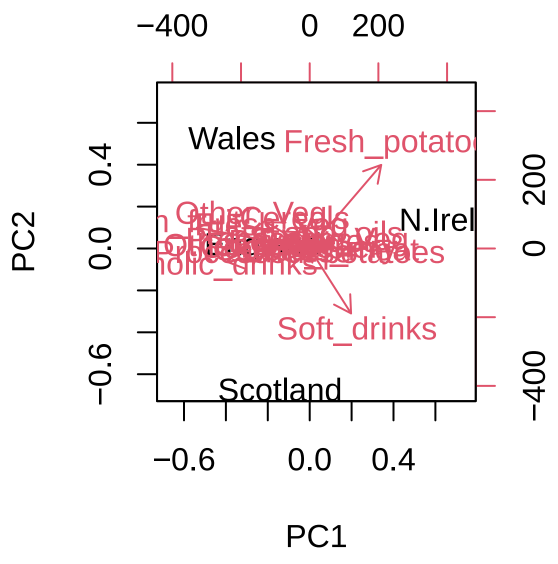
Q9. Generate a similar 'loadings plot' for PC2. What two food groups feature prominently and what does PC2 mainly tell us about?

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



The food with the largest positive loading score is mainly `Fresh_potatoes` and the food with the highest negative score is mainly `Soft_drinks`. PC2 tells us about the second principle component, which is the secondary trend or pattern that is orthogonal to PC1.

```r
biplot(pca)
```



PCA of a RNA-Seq Dataset

```
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?

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

```
[1] 100  10
```

I have 100 genes and 10 samples.

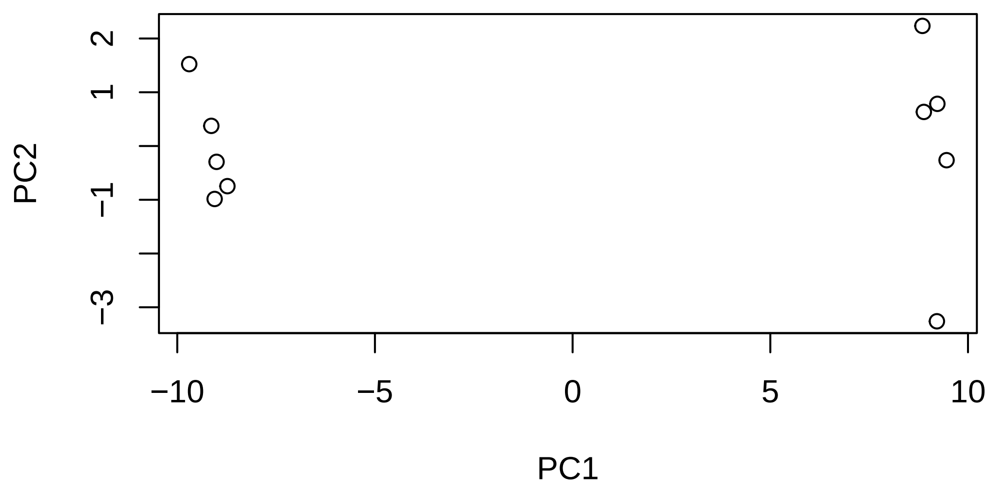
Let's apply PCA:

```
pca_rna = prcomp(t(rna.data), scale = TRUE)
summary(pca_rna)
```

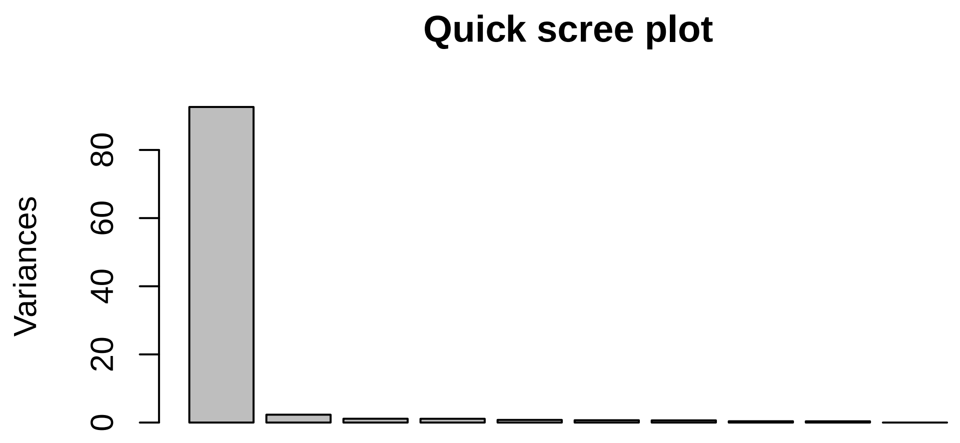
```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     9.6237 1.5198 1.05787 1.05203 0.88062 0.82545 0.80111
Proportion of Variance 0.9262 0.0231 0.01119 0.01107 0.00775 0.00681 0.00642
Cumulative Proportion  0.9262 0.9493 0.96045 0.97152 0.97928 0.98609 0.99251
                           PC8     PC9      PC10
Standard deviation     0.62065 0.60342 3.348e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

Let's plot the principle components 1 and 2.

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



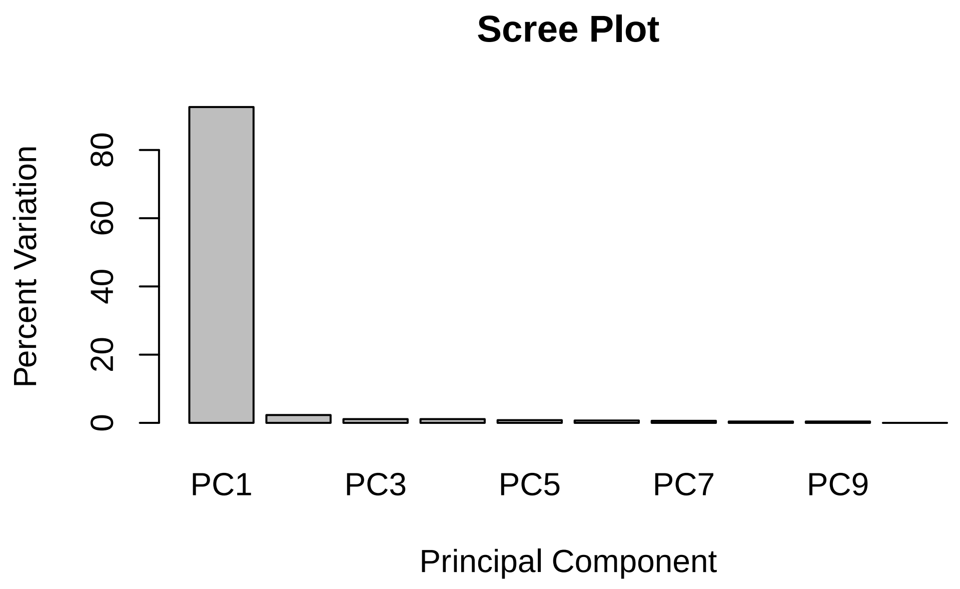
```r
plot(pca_rna, main = "Quick scree plot")
```



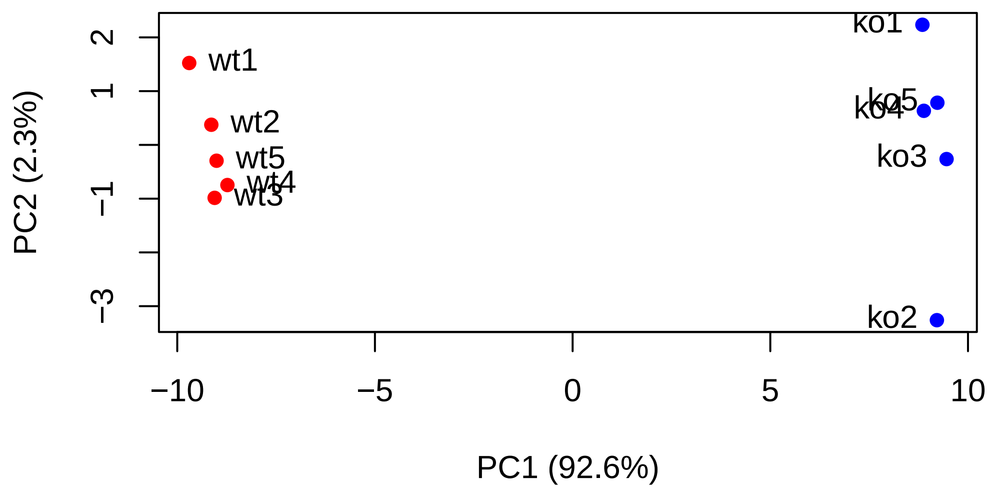
```r
## Variance captured per PC
pca.var <- pca_rna$sdev^2

## 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
```

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

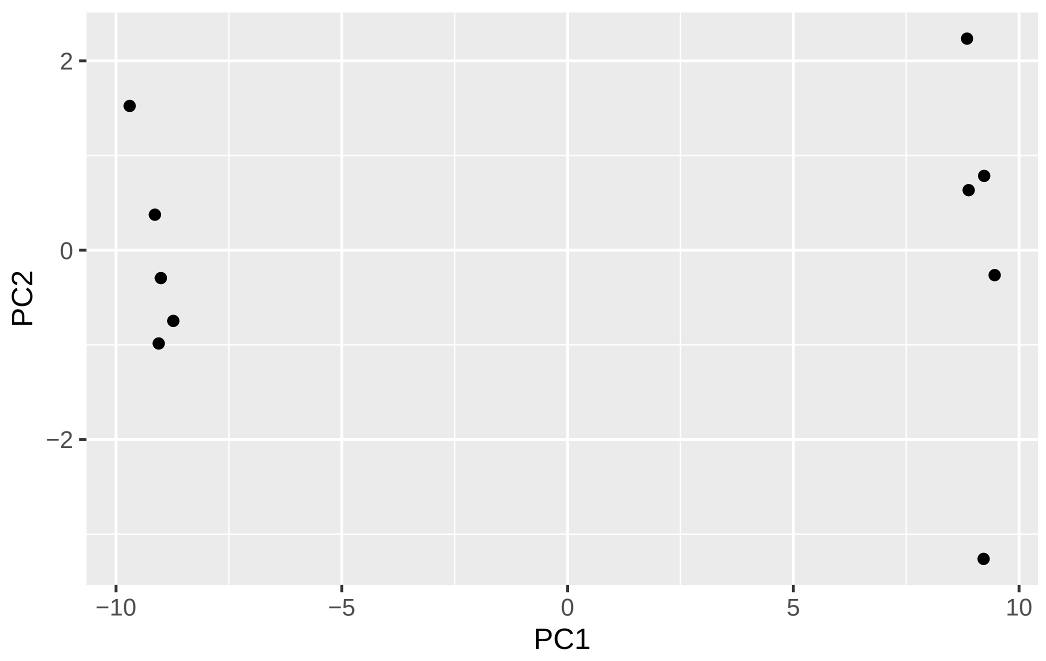


```
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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

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

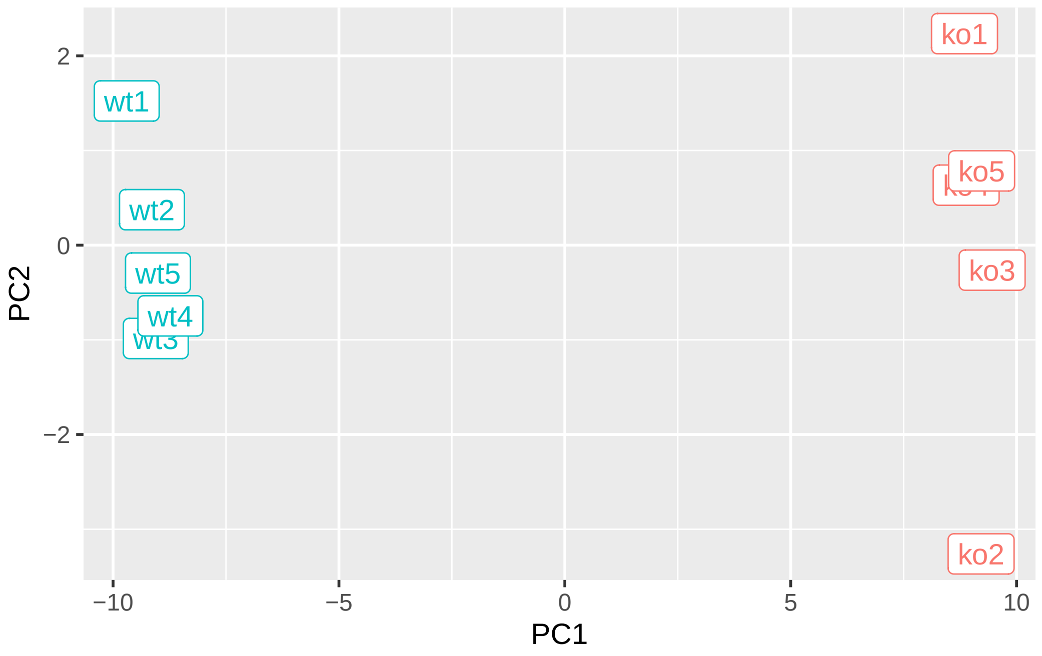


Using ggplot

```
library(ggplot2)

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

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



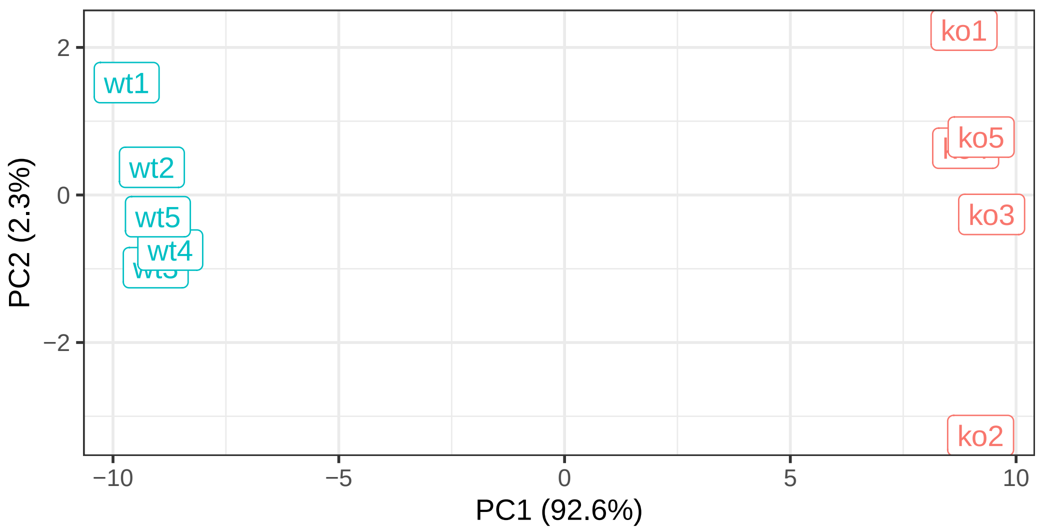
```r
# Add a 'wt' and 'ko' "condition" column
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data), 1, 2)

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 cleanly separates wild-tyoe 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 cleanly separates wild−tyoe from knock−out samples



Class example data