Class 07 R: Machine Learning

Katelyn Brown PID: A15891811

2/8/2022

Clustering Methods

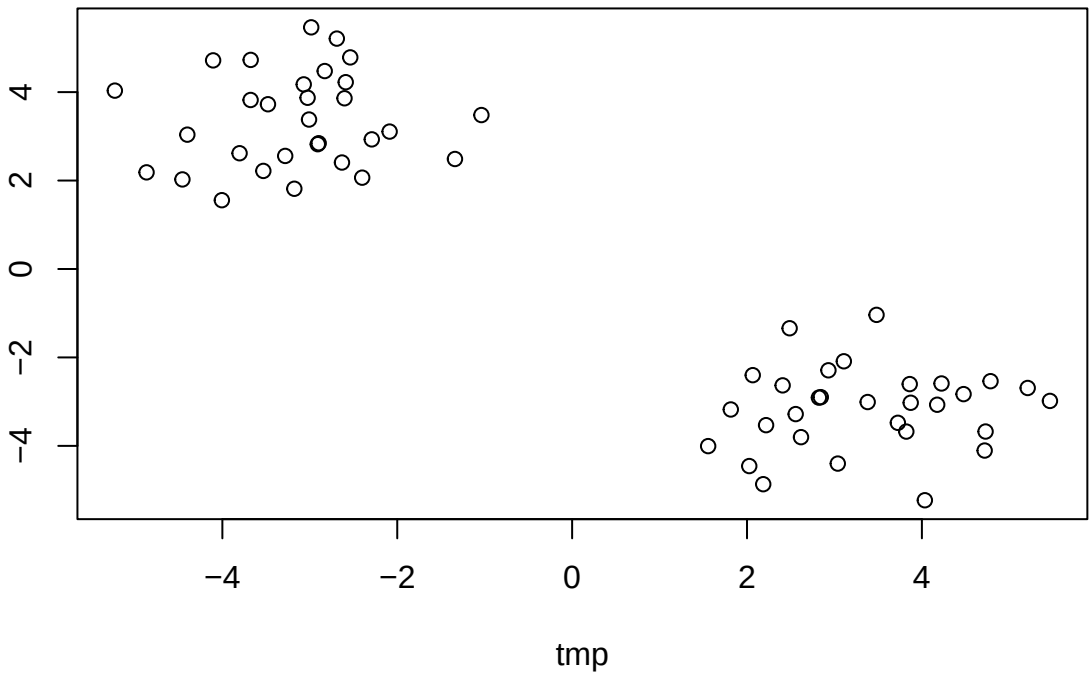
Find groups (clusters) in my data.

K-means clustering

Make up some data to test with.

```
# Make up some data with two clear groups
tmp <- c(rnorm(30, mean = 3), rnorm(30, mean = -3))
x <- cbind(tmp, rev(tmp))

plot(x)
```



The `kmeans()` function does K-means clustering

```
k <- kmeans(x, centers = 4, nstart = 20)
k
```

```
## K-means clustering with 4 clusters of sizes 17, 13, 13, 17
##
## Cluster means:
##         tmp
## 1 -3.066367  2.561070
## 2 -3.268819  4.391981
## 3  4.391981 -3.268819
## 4  2.561070 -3.066367
##
## Clustering vector:
##  [1] 4 3 4 3 3 4 4 4 4 4 3 3 3 3 4 4 4 3 4 3 4 4 3 4 3 4 3 4 4 3 2 1 1 2 1 2 1 2
## [39] 1 1 2 1 2 1 1 1 2 2 2 2 1 1 1 1 1 2 2 1 2 1
##
## Within cluster sum of squares by cluster:
## [1] 22.71961 10.79466 10.79466 22.71961
##  (between_SS / total_SS =  95.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

How many points are in each cluster?

We can use the dollar syntax to get at the results (components of the list)

```
k$size
```

```
## [1] 17 13 13 17
```

There are 30 points in each cluster.

Q2. What 'component' of your result object details - cluster size? - cluster assignment/membership? - cluster center?

```
k$size
```

```
## [1] 17 13 13 17
```

```
k$cluster
```

```
##  [1] 4 3 4 3 3 4 4 4 4 4 3 3 3 3 4 4 4 3 4 3 4 4 3 4 3 4 3 4 4 3 2 1 1 2 1 2 1 2
## [39] 1 1 2 1 2 1 1 1 2 2 2 2 1 1 1 1 1 2 2 1 2 1
```

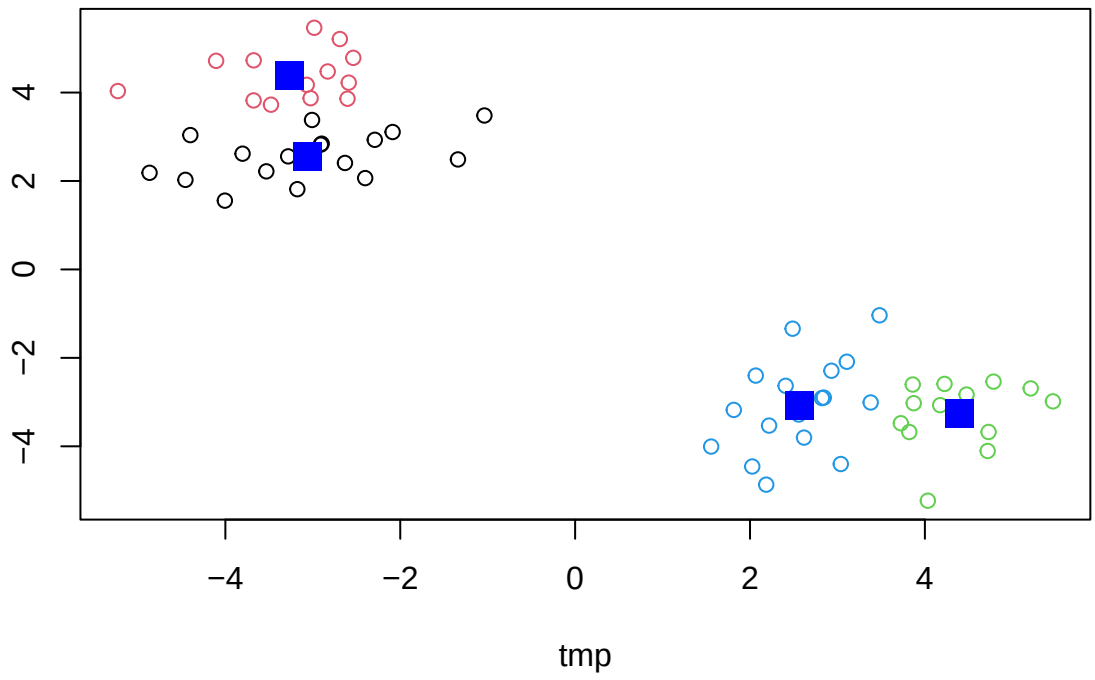
```
k$centers
```

```
##         tmp
## 1 -3.066367  2.561070
## 2 -3.268819  4.391981
## 3  4.391981 -3.268819
## 4  2.561070 -3.066367
```

Cluster size is 'size', cluster membership is 'cluster', and cluster center is 'center'.

Q3. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points.

```
plot(x, col = k$cluster)
points(k$centers, col = "blue", pch = 15, cex = 2)
```



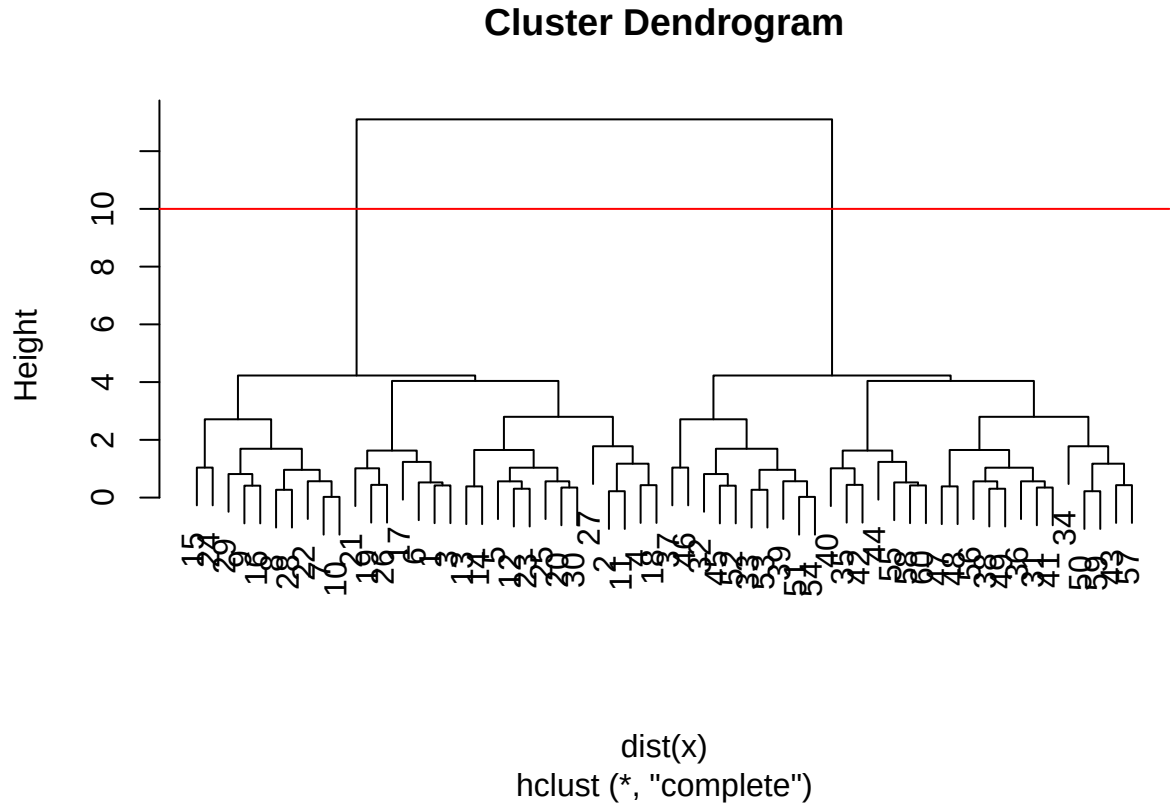
Hierarchal Clustering

The `hclust()` function needs a distance matrix as input but not our original data. For this we use the `dist()` function.

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

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

```
# Visualize plot
plot(hc)
abline(h = 10, col = "red")
```



To get our cluster membership vector, we need to cut our tree and for this we use the `cutree()`

```
# Cut tree at height = 10
cutree(hc, h = 10)
```

```
##  [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
```

You can cut by a given height h = or into a given number of groups with k =

```
# Cut tree into 2 groups
cutree(hc, 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 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
```

Principal Component Analysis (PCA)

PCA of UK food data

Let's read our data about the stuff people from the UK eat and drink.

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
```

Look at the first bit of the file.

```
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. What are the dimensions in this dataset?

```
dim(x)
```

```
## [1] 17  4
```

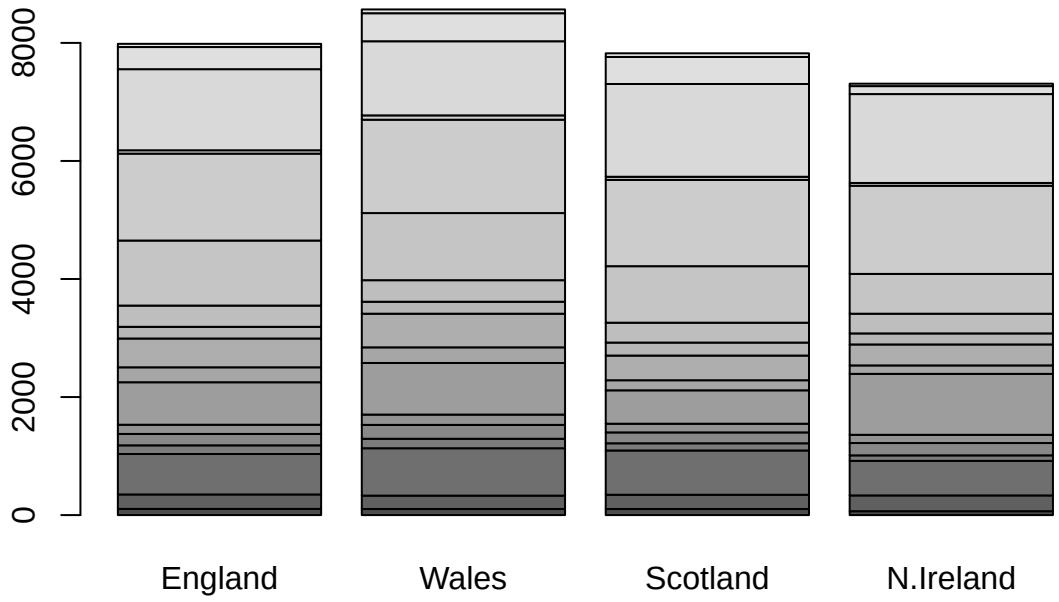
Q2. How do you solve the "row-names problem?"

I solved the row-names problem when assigning the csv file to "x," changing the row names to the first column.

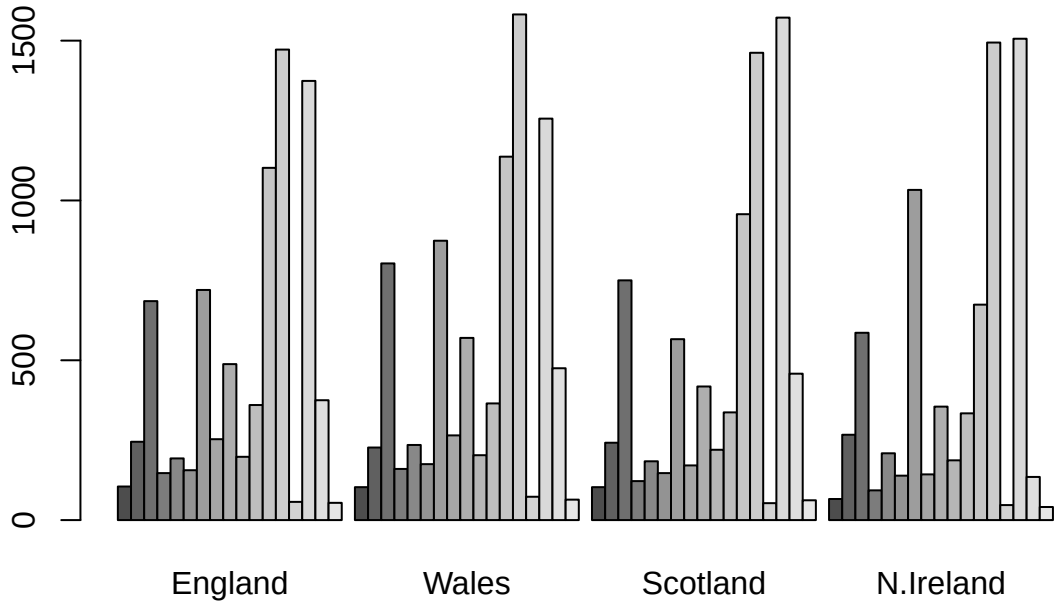
We can try to make some plots to understand the data a bit more.

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

```
# Make barplot
barplot(as.matrix(x))
```




```
# Edit barplot
barplot(as.matrix(x), beside = TRUE)
```



Using the argument within barplot() of `beside = FALSE` results in a stacked plot.

Q6. What is the main difference between N. Ireland and other countries in the UK?

Although the differences are hard to visualize using the more basic plots, N. Ireland has a more varied spread of food eaten, whereas the other countries in the UK have very similar food data.

PCA to the rescue

The main base R function for PCA is called `prcomp()`.

```
# Make countries as rows, foods as variables
t(x)
```

```
##           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
```

```
# prcomp
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
```

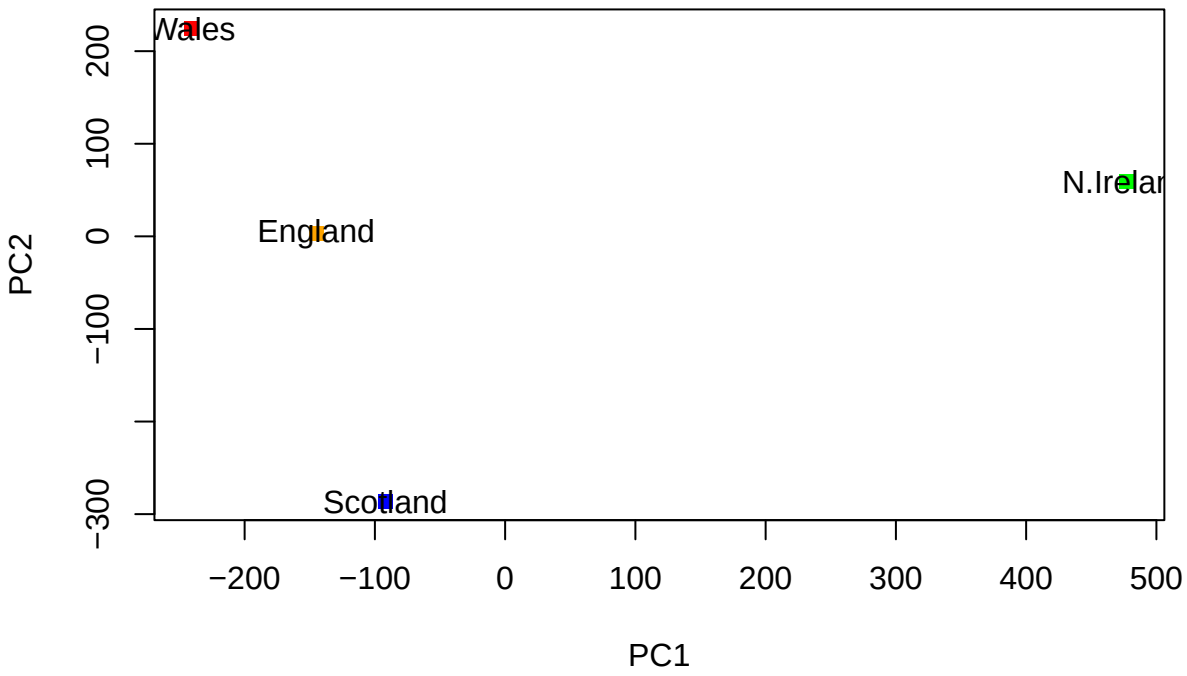
The more variance in a PC, the better. What is in this returned PCA object?

```
# Look at attributes of pca
attributes(pca)
```

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

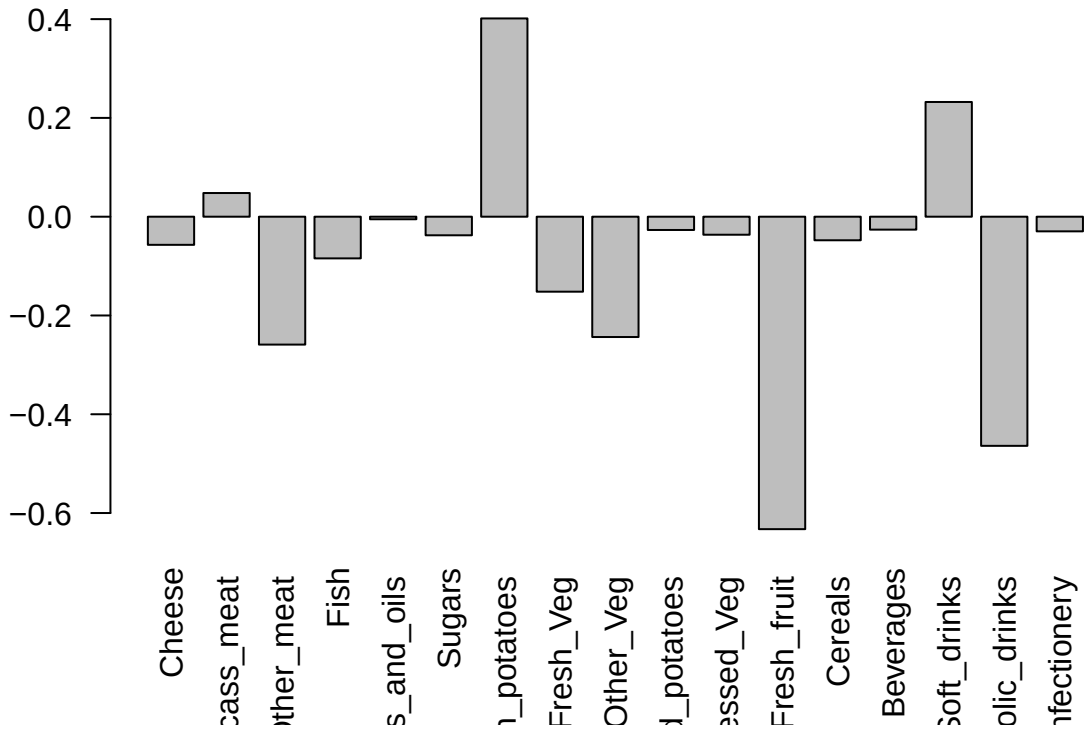
Q7. Complete the code to plot PC1 versus PC2. Q8. Customize the plot so the colors of the country match the colors in the UK and N. Ireland map.

```
# Plot PC1 against PC2
plot(pca$x[,1:2], col = c("orange", "red", "blue", "green"), pch = 15)
text(pca$x[,1], pca$x[,2], labels = colnames(x))
```



We can look at how the variables contribute to our new PCs by examining the `pca$rotation` component of our new PCs.

```
barplot(pca$rotation[,1], las = 2)
```



PCA of RNA-seq data

Let's read the data of gene expression.

```
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 (rows)?

```
nrow(rna.data)
```

```
## [1] 100
```

How many experiments (columns)?

```
ncol(rna.data)
```

```
## [1] 10
```

There are 100 genes, with 5 wildtype samples and 5 knockout samples for each gene (10 total).

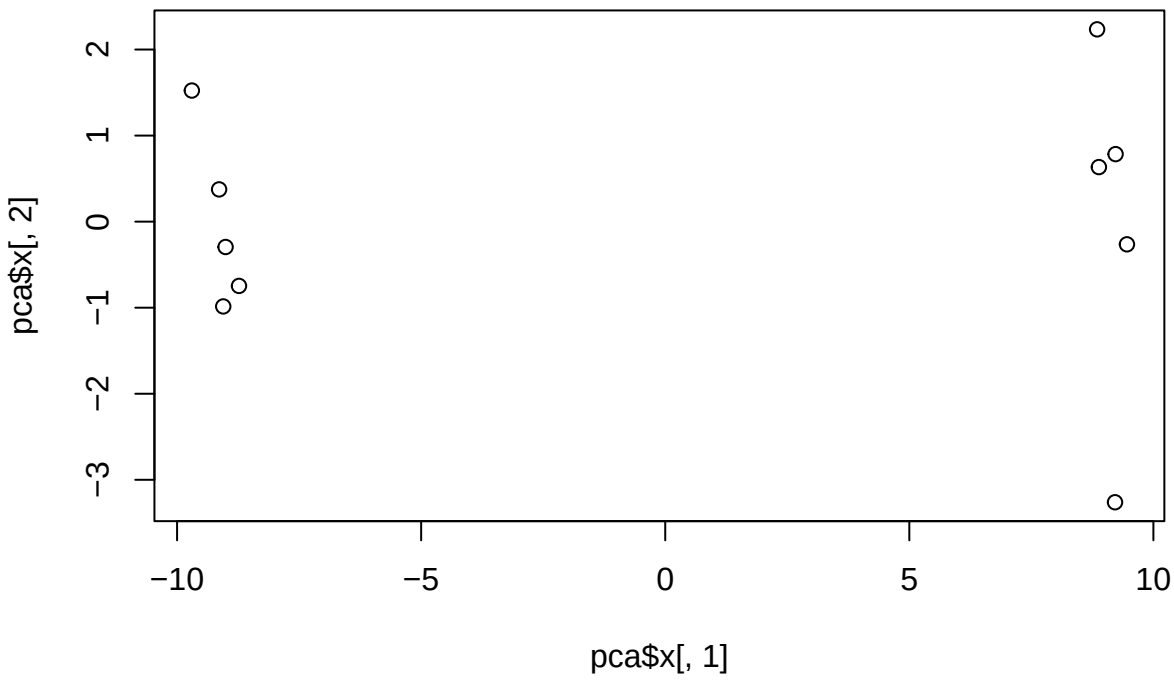
Let's do PCA of this dataset. First we take the transpose as that is what the prcomp() function wants.

```
pca <- prcomp(t(rna.data), scale = TRUE)
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 3.348e-15
## Proportion of Variance 0.00385 0.00364 0.000e+00
## Cumulative Proportion  0.99636 1.00000 1.000e+00
```

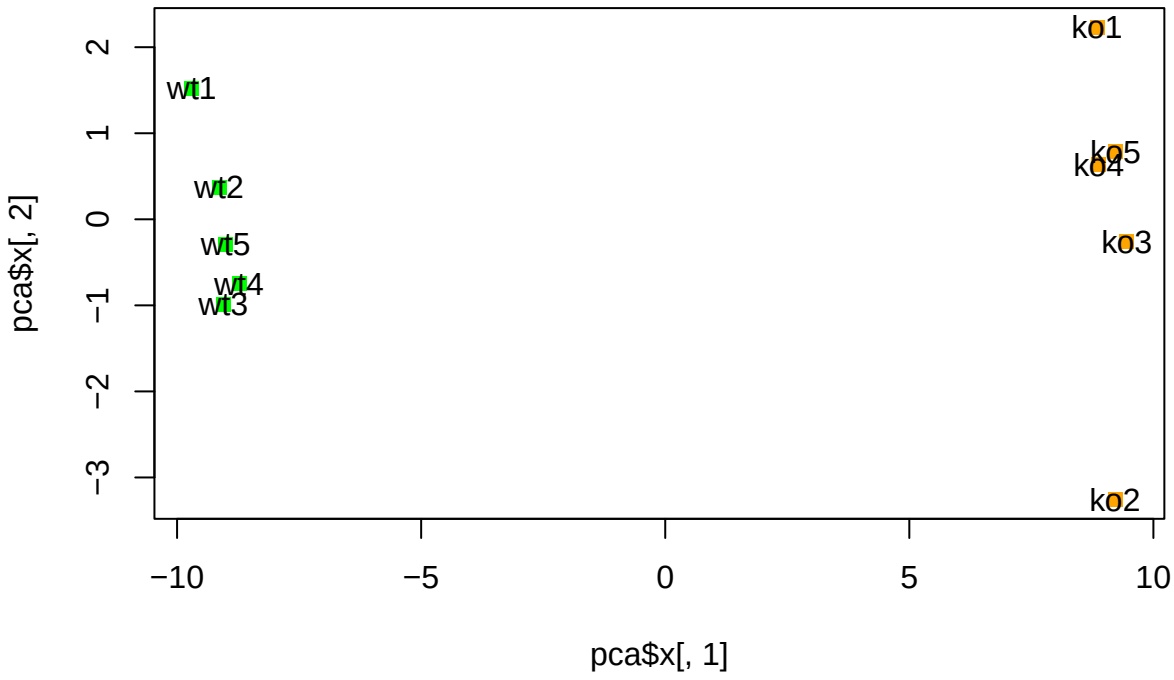
PC1 is the best measure of variance, with 92.62% of the variance being accounted for in PC1. We can make our score (PCA) plot from the `pca$x`.

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



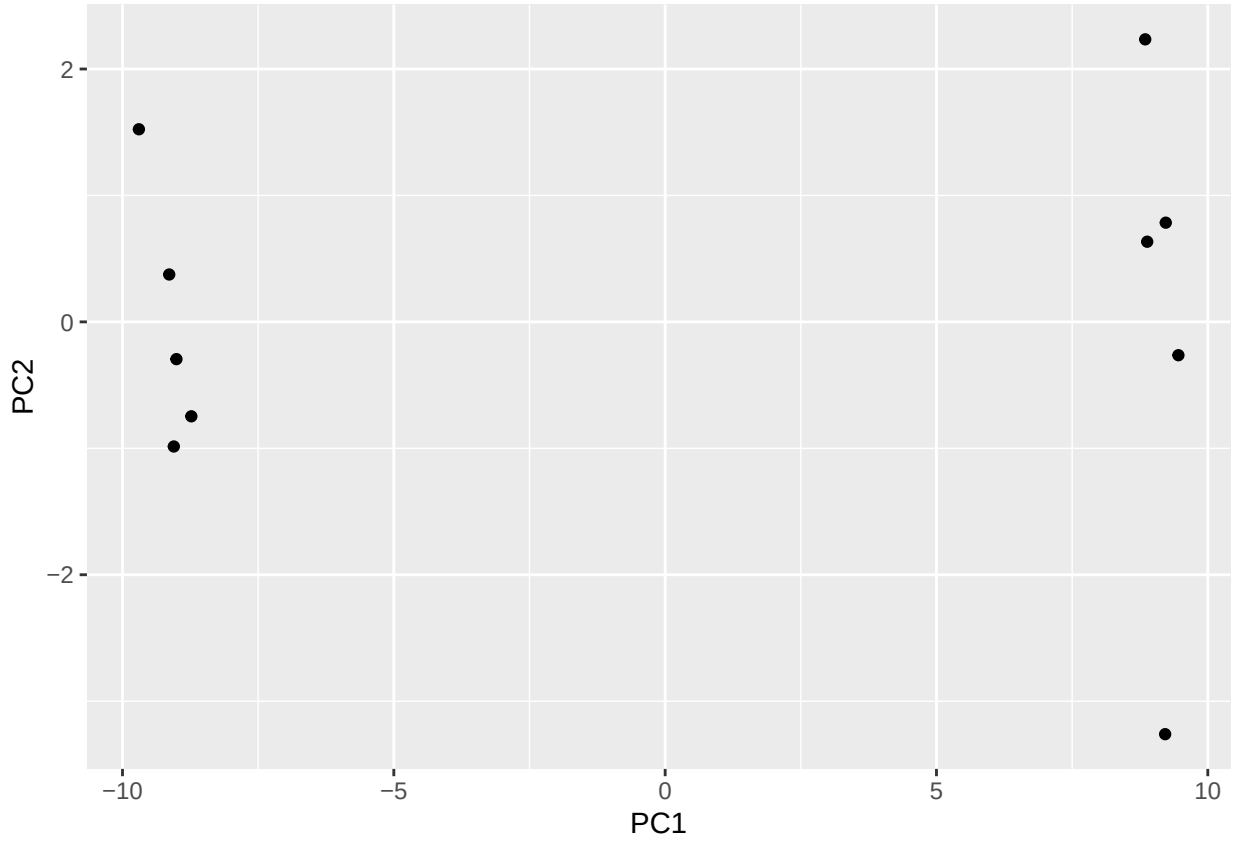
Make a color vector to color in the plot by wt and ko

```
colvec <- c(rep("green", 5), rep("orange", 5))
plot(pca$x[,1], pca$x[,2], col = colvec, pch = 15)
text(pca$x[,1], pca$x[,2], labels = colnames(rna.data))
```



Use ggplot to make new plots.

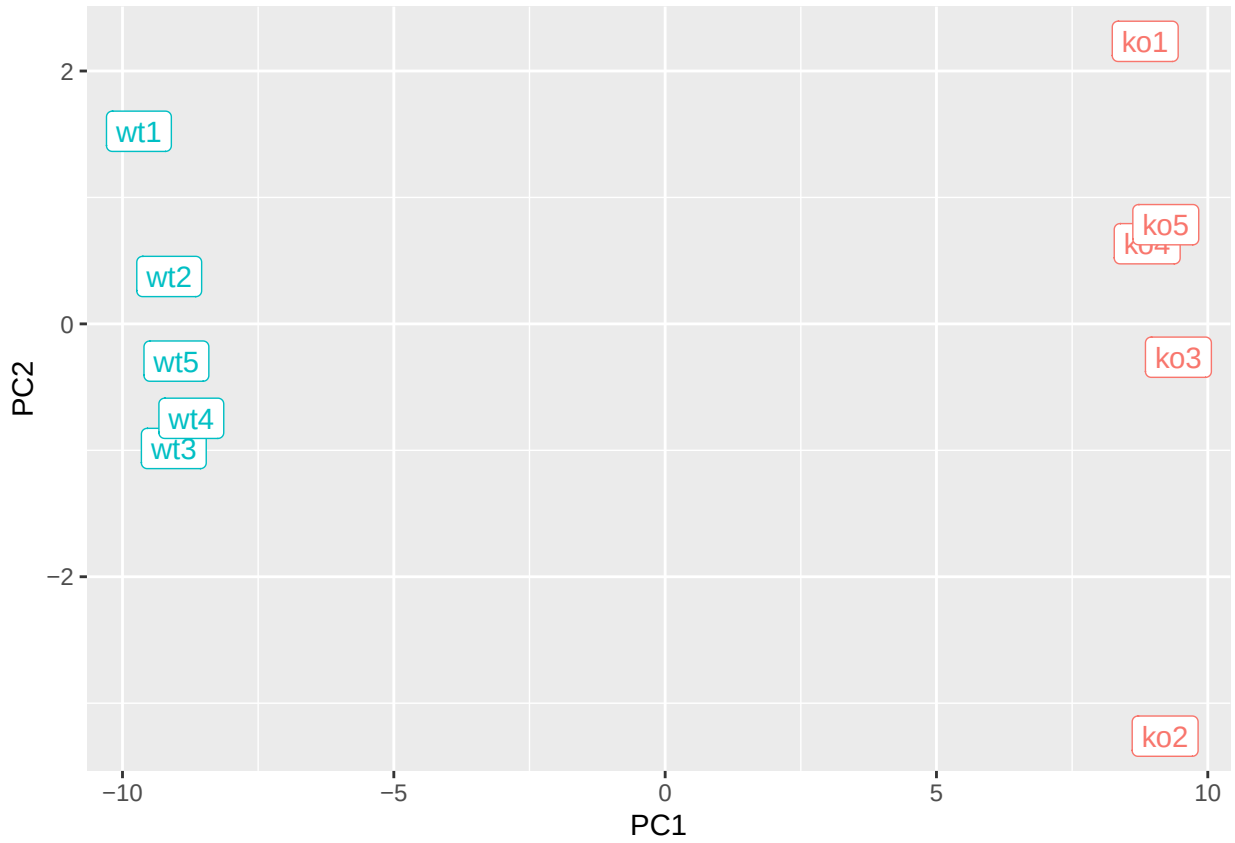
```
# Load ggplot
library(ggplot2)
df <- as.data.frame(pca$x)
# Make basic plot
ggplot(df) + aes(PC1, PC2) + geom_point()
```



Edit and add to the ggplot.

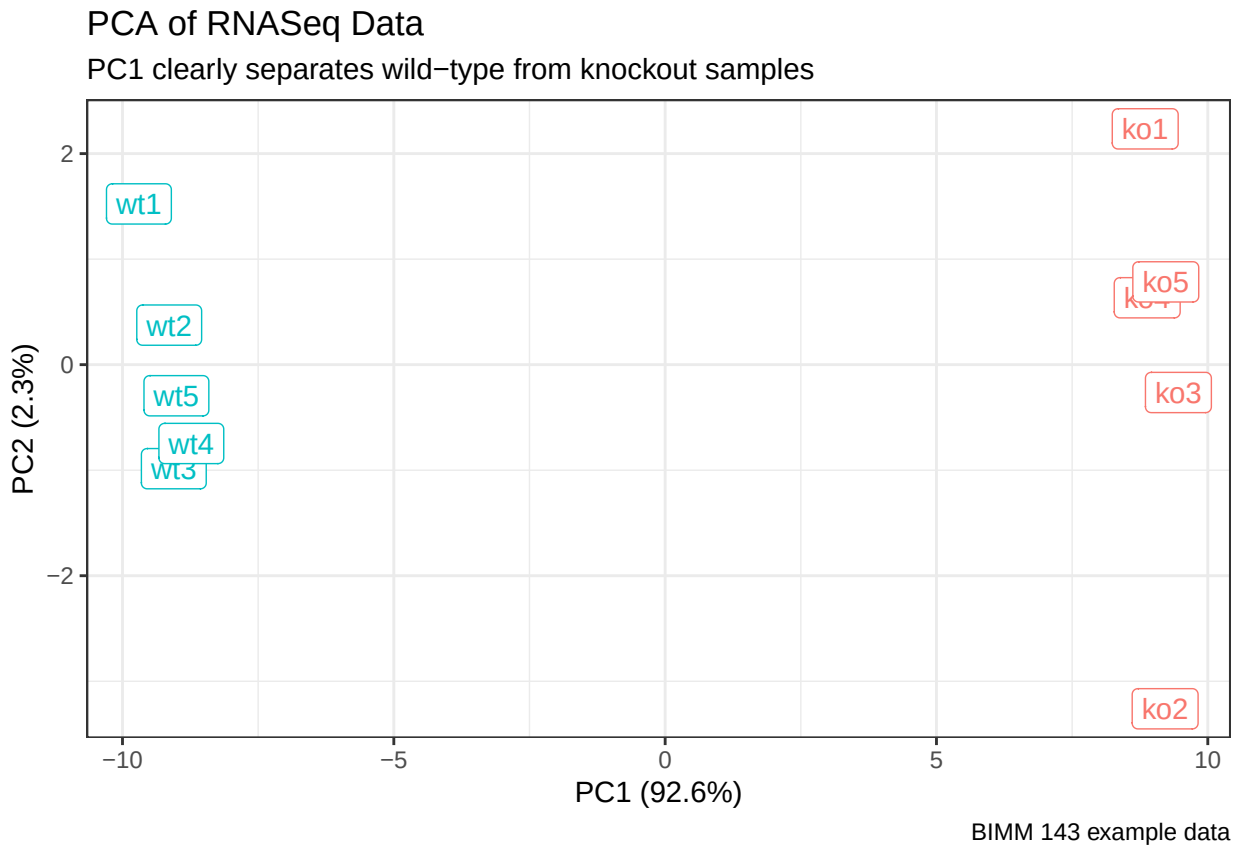
```
# Add `wt` and `ko` condition column
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data), 1, 2)

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

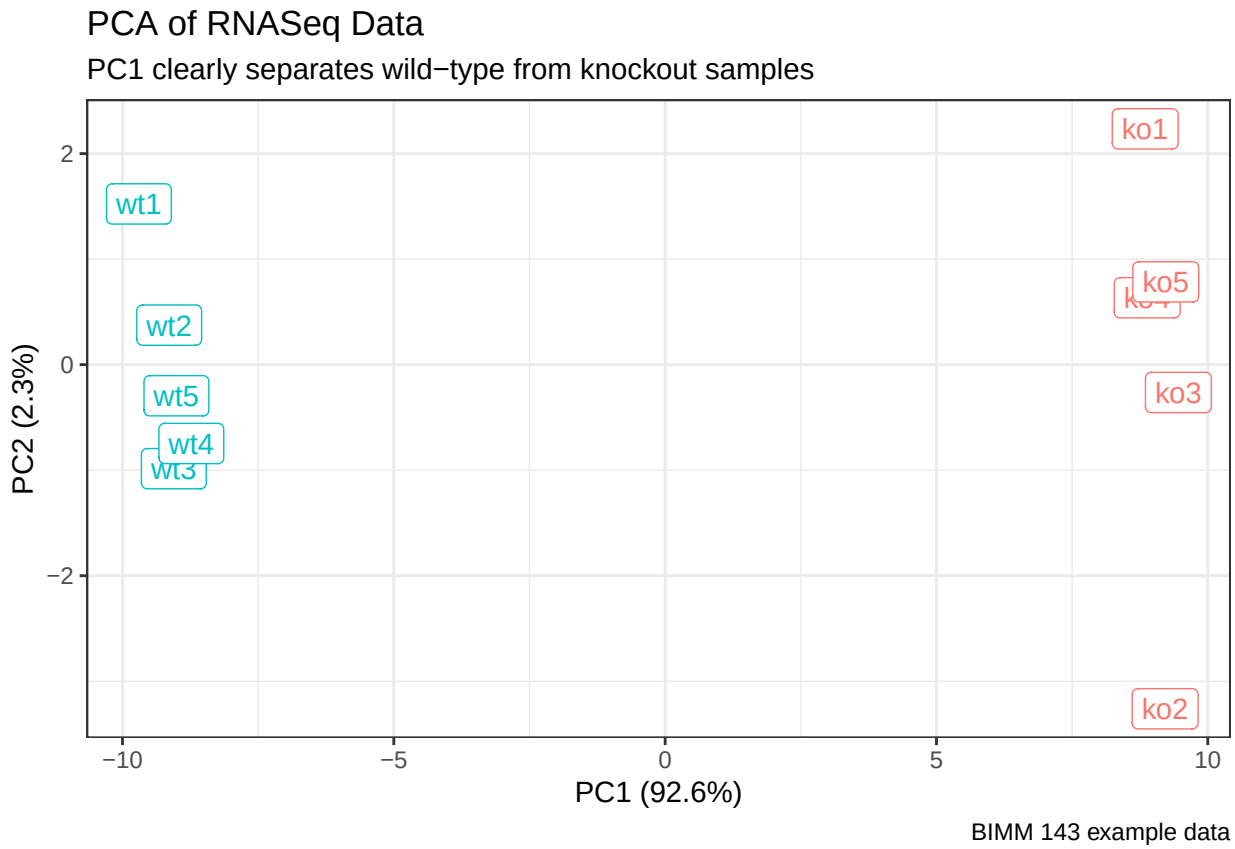


Make the ggplot more cohesive by adding more elements.

```
p + labs(title = "PCA of RNASeq Data", subtitle = "PC1 clearly separates wild-type from knockout samples
```




```
# Make pca.var.per
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
# Test ggplot with pca.var.per
p + labs(title = "PCA of RNASeq Data", subtitle = "PC1 clearly separates wild-type from knockout samples
```



Gene loadings

```
loading_scores <- pca$rotation[,1]
# Find top 10 genes that contribute most to PC1 in either direction
gene_scores <- abs(loading_scores)
gene_score_ranked <- sort(gene_scores, decreasing = TRUE)
# Show names of top 10 genes
top_10_genes <- names(gene_score_ranked[1:10])
top_10_genes
```

```
##  [1] "gene100" "gene66"  "gene45"  "gene68"  "gene98"  "gene60"  "gene21"
##  [8] "gene56"  "gene10"  "gene90"
```