Class 07: Machine Learning 1

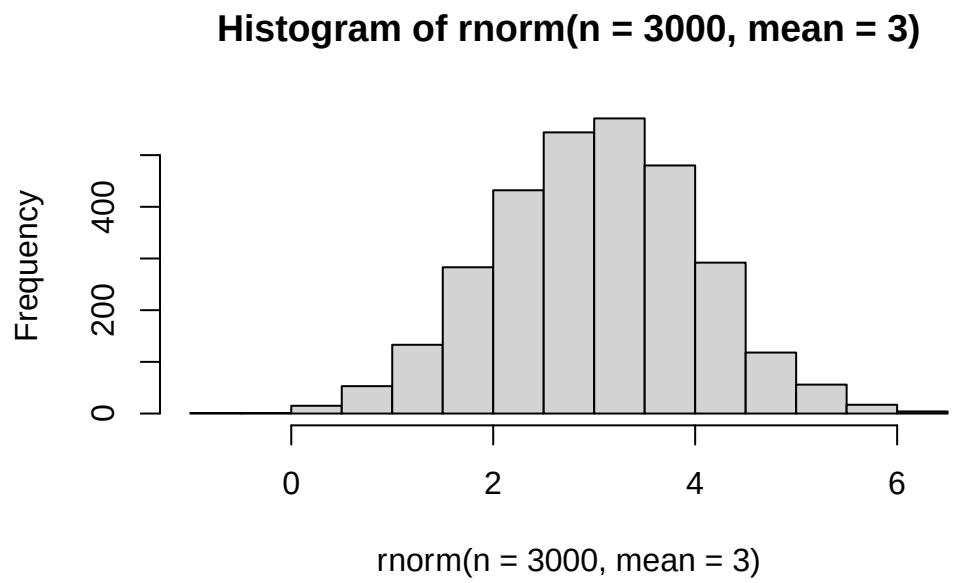
Emily Ignatoff (A16732102)

Today we will be exploring unsupervised machine learning methods such as clustering and dimensionality reduction.

Let us first make up some data where we already know the answer (where we know there are clear groups that we can use to test clustering methods):

We can use the `rnorm()` function to help us here

```
hist(rnorm(n=3000, mean=3))
```

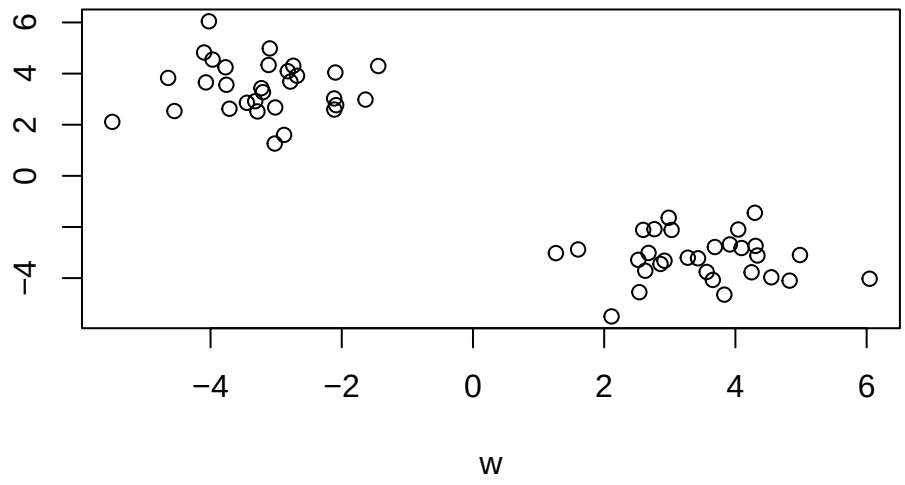


Make data `z` with two "clusters":

```r
w <- c(rnorm(30, mean= -3), rnorm(30, mean= 3))

z <- cbind(w, rev(w)) #cbind stands for column bind
head(z)
```

```
             w
[1,] -2.097526 4.043037
[2,] -3.021835 1.263768
[3,] -1.638498 2.983237
[4,] -4.098514 4.826437
[5,] -3.097274 4.984106
[6,] -2.115785 3.027435
```

```r
plot(z)
```



How big is z?

```r
nrow(z)
```

```
[1] 60
```

```
ncol(z)
```

```
[1] 2
```

K-means clustering

THe first method we will try is K-means clustering. The main function in base R to do this is `kmeans()`

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

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

Cluster means:
          w
1 -3.207466  3.452064
2  3.452064 -3.207466

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 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] 55.85929 55.85929
 (between_SS / total_SS =  92.3 %)

Available components:

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

```
attributes(k)
```

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

$class
[1] "kmeans"
```

Q. How many points lie in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What component of our results tells us about the cluster membership? (i.e. which points lie in which cluster?)

```
k$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 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
```

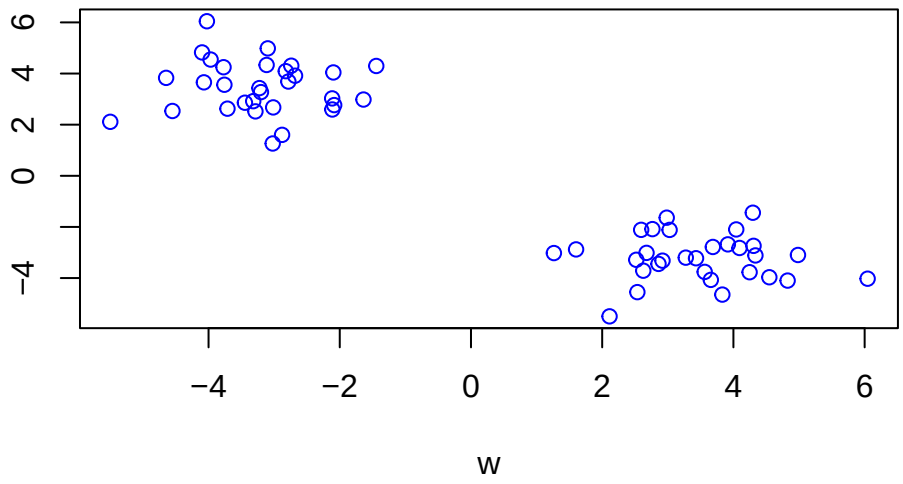
Q. Center of each cluster?

```
k$centers
```

```
          w
1 -3.207466  3.452064
2  3.452064 -3.207466
```

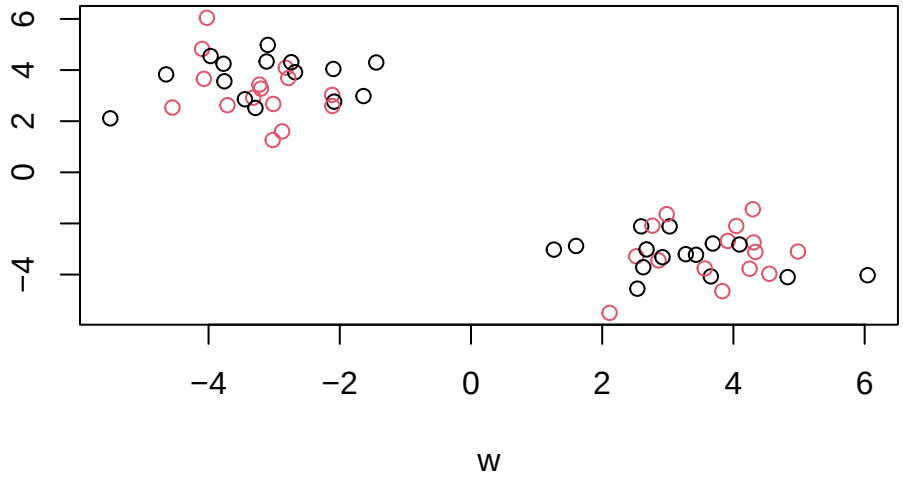
Q. Put this result infor together to make a little 'base R' plot of our clustering results. Also add the center cluster points to this plot.

```
plot(z, col="blue")
```

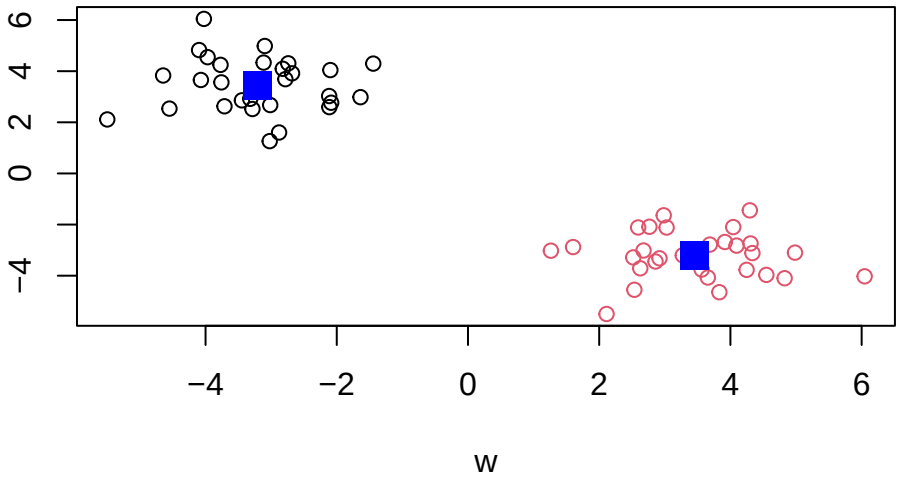


You can color by number.

```
plot(z, col= c(1, 2))
```

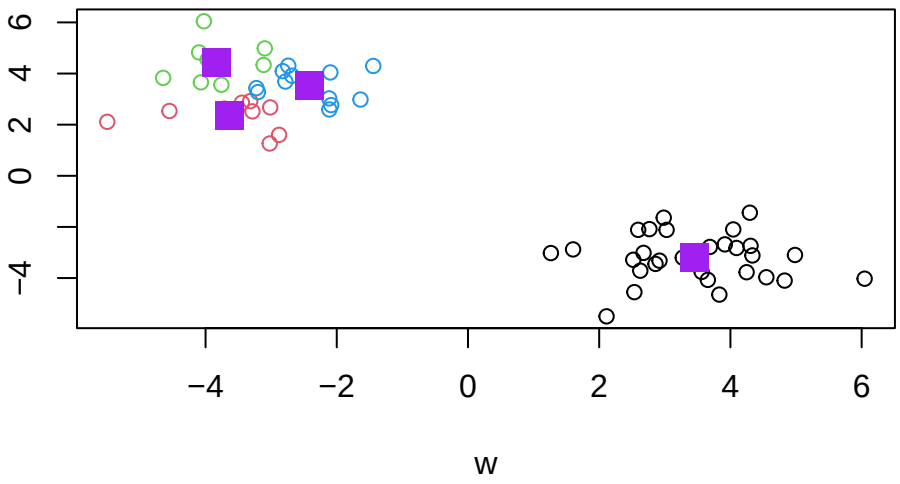


```r
plot(z, col=k$cluster)
points(k$centers, col="blue", pch=15, cex=2)
```



Q. Run kmeans on our input `z` and define 4 clusters, making the same result visualization plot.

```r
y <- kmeans(z, centers=4)

plot(z, col=y$cluster)
points(y$centers, col="purple", pch= 15, cex=2)
```




```
y$tot.withinss
```

```
[1] 78.8558
```

```
k$tot.withinss
```

```
[1] 111.7186
```

Hierarchical clustering:

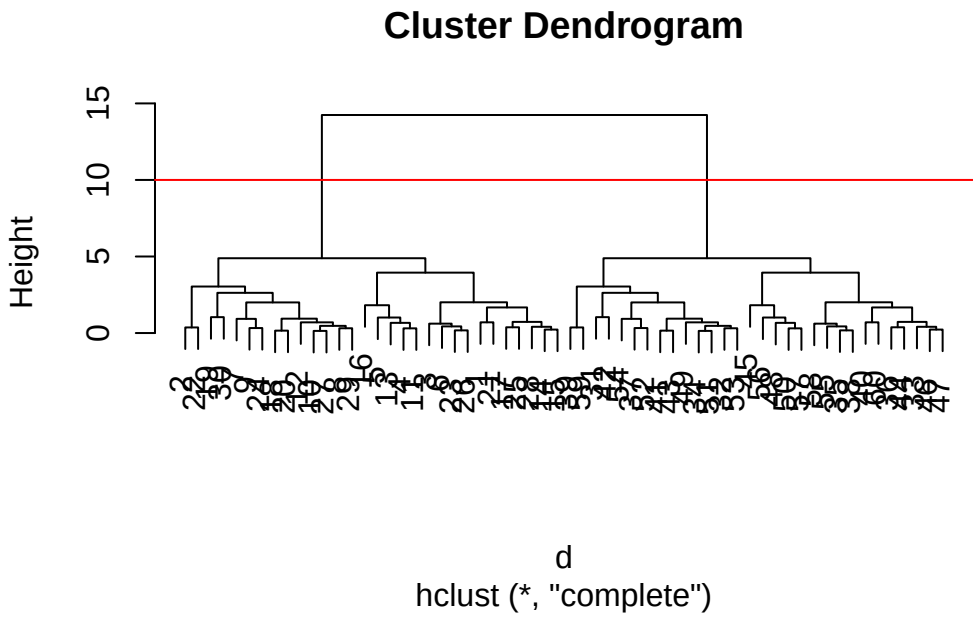
We will next utilize the `hclust()` function to accomplish hierarchical clustering. This function does not calculate distance for you, adding an extra step but increasing the flexibility of the clustering methods. Our input to the function will be a distance matrix.

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

```
Call:
```

```
hclust(d = d)

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

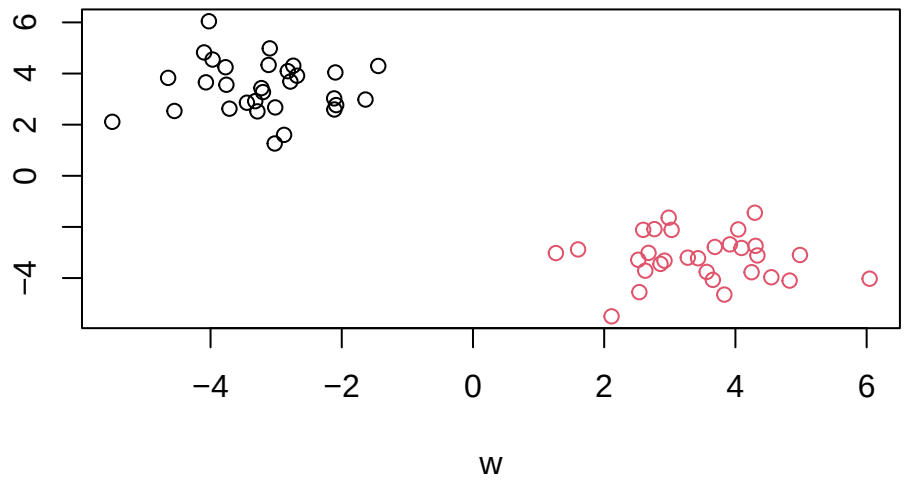
```
plot(hc)
abline(h=10, col="red")
```



Once I inspect the tree (dendrogram), I can "cut" the tree to yield my groupings or clusters. The function to do this is called `cutree()`

```
grps <- cutree(hc, h=10)
```

```
plot(z, col=grps)
```



Principal Component Analysis (PCA)

In an PCA, each axis is a principal component. PC1 follows a best fit line through the data, PC2 are 'surfaces' closest to to the observations. Data can be plotted in these new axes to better represent the relationships within the data. Maximum variance along PC1, then PC2, etc.

We will examine a 17 dimensional data set detailing food consumption in the UK. Are the countries different? How so?

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1)
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
Fresh_potatoes      720   874      566      1033
```

```
Fresh_Veg                253   265       171       143
Other_Veg                488   570       418       355
Processed_potatoes       198   203       220       187
Processed_Veg            360   365       337       334
Fresh_fruit             1102  1137       957       674
Cereals                 1472  1582      1462      1494
Beverages                 57    73        53        47
Soft_drinks             1374  1256      1572      1506
Alcoholic_drinks         375   475       458       135
Confectionery             54    64        62        41
```

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

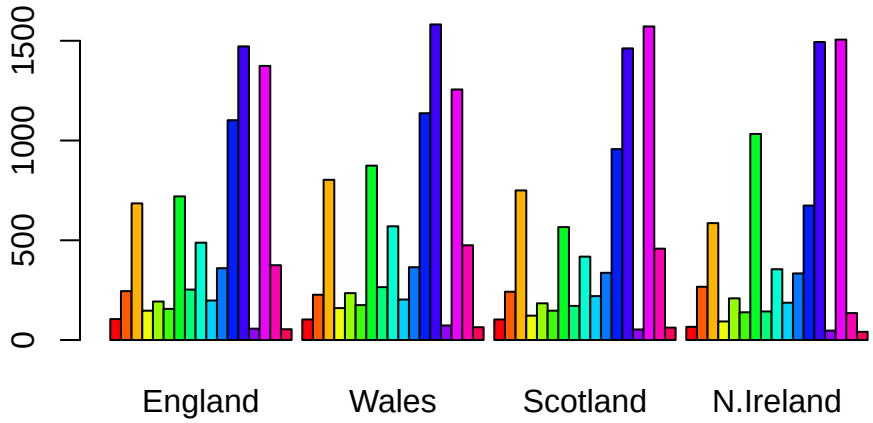
```
dim(x)
```

```
[1] 17  4
```

> 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 `row.names =1` approach as it allows us to quickly assign the row names prior to uploading the dataset.

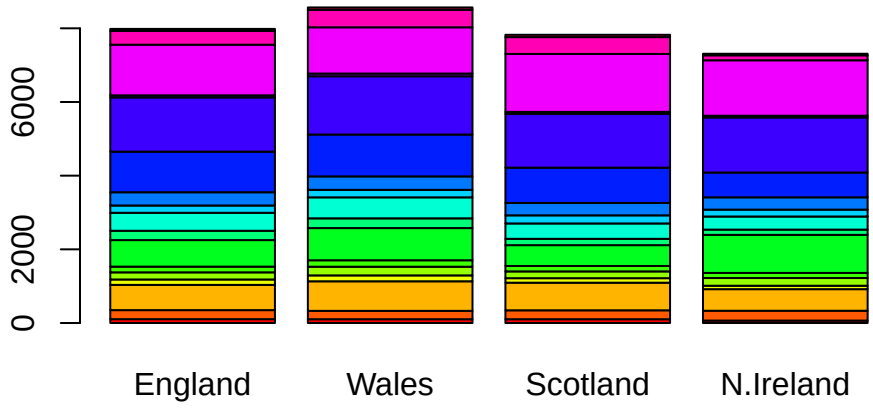
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

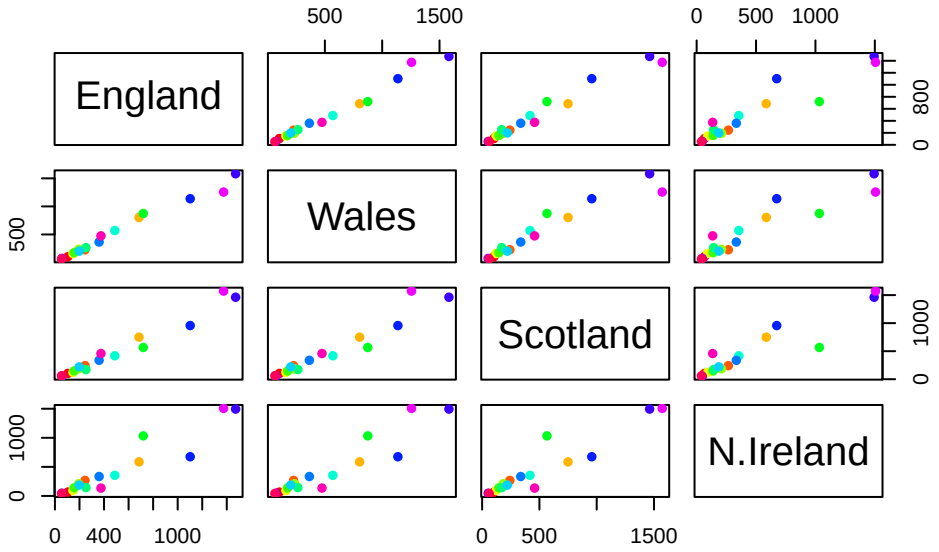
Change `beside = T` to `beside = F`!

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



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=rainbow(nrow(x)), pch=16)
```



This plot compares consumption of food items between countries in a pairwise manner. Linear fit of a point relates to similarity of consumption of the corresponding food item between the two countries being compared.

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

N. Ireland has much lower linear fit compared to other countries

Looking at these plots can be helpful but it does not scale well and kind of sucks!

PCA to the rescue!

The main function for PCA in 'base R' is called `prcomp()`. This function wants the transpose of out input data (i.e. the food categories as columns and countries as rows)

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

```
Importance of components:
                            PC1      PC2      PC3          PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
```

```
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

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

Let's see what is in our PCA result object `pca`

```
attributes(pca)
```

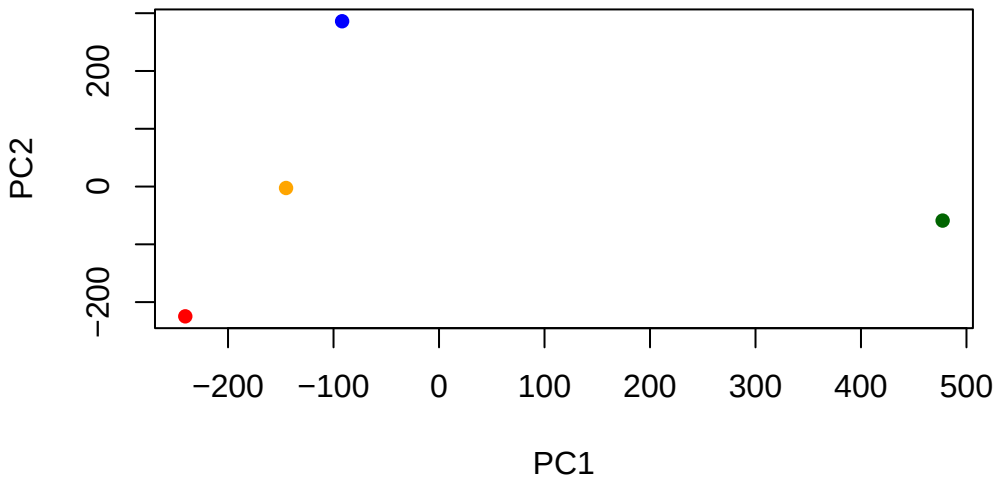
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

The `pca$x` result object is where we will focus first as this details how the countries are related to each other in terms of the new "axis" (aka PCs) generated in the prior steps.

```
head(pca$x)
```

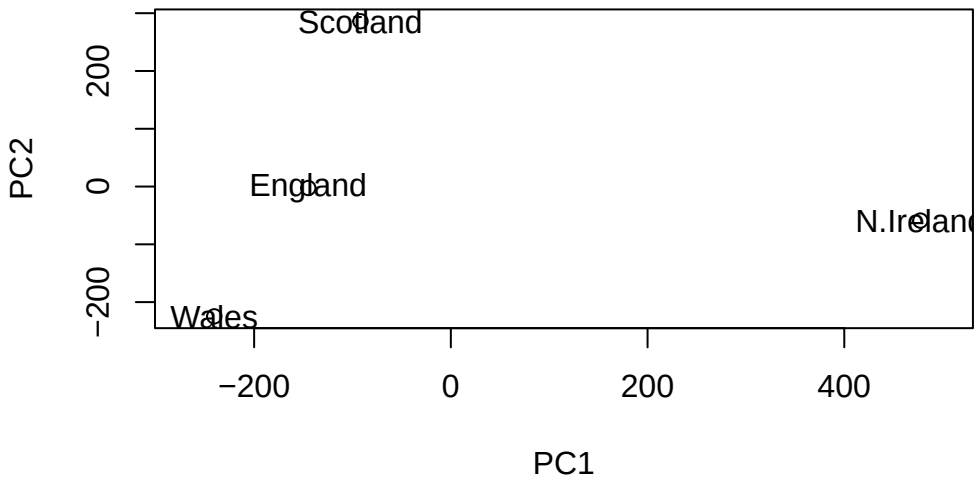
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
plot(pca$x[,1], pca$x[,2], col=c("orange","red","blue","darkgreen"), pch=16, xlab= "PC1", yla
```



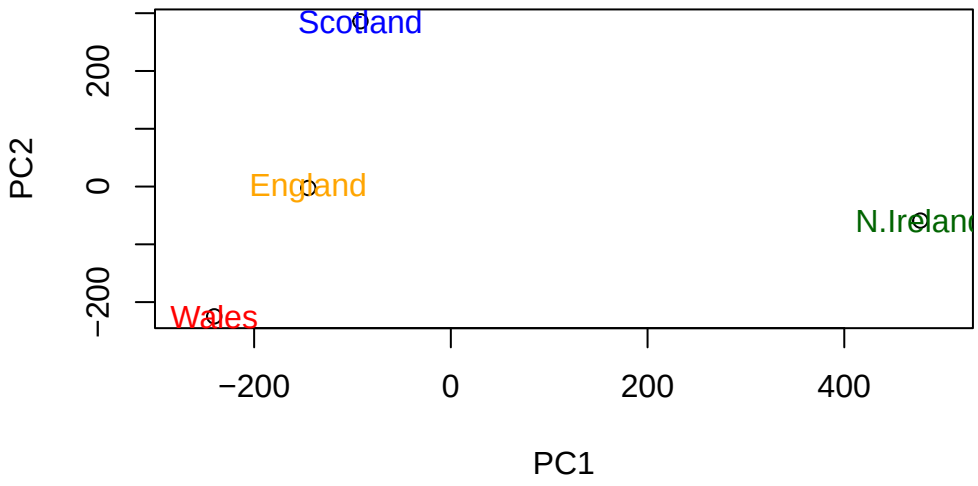
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
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.

```
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","darkgreen"))
```



We can look at the so-called PC "loadings" result object to see how the original foods compare to our new PCs (how old variable contribute to new, better variables)

```
pca$rotation[,1]
```

```
          Cheese       Carcass_meat         Other_meat                Fish
    -0.056955380         0.047927628       -0.258916658       -0.084414983
Fats_and_oils              Sugars     Fresh_potatoes          Fresh_Veg
    -0.005193623       -0.037620983         0.401402060       -0.151849942
   Other_Veg  Processed_potatoes       Processed_Veg         Fresh_fruit
    -0.243593729       -0.026886233       -0.036488269       -0.632640898
      Cereals            Beverages       Soft_drinks    Alcoholic_drinks
    -0.047702858       -0.026187756         0.232244140       -0.463968168
Confectionery
    -0.029650201
```

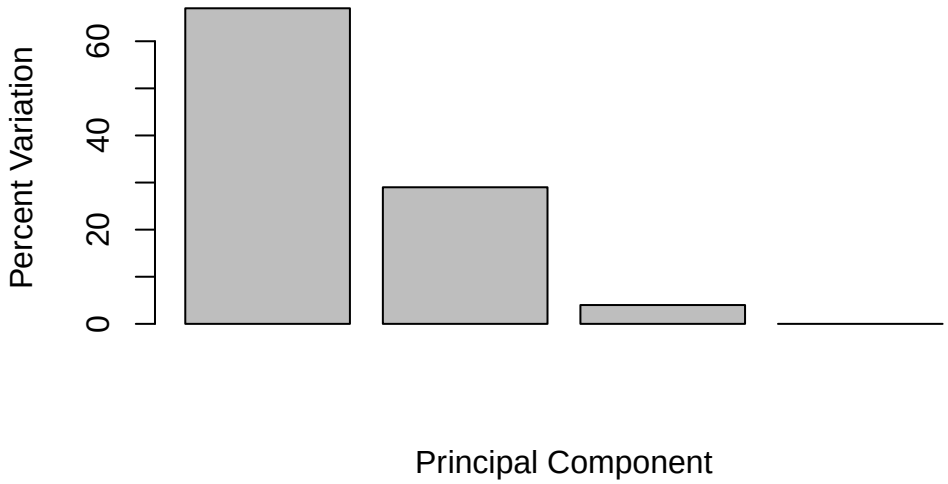
We can also use standard deviation squared to determine the variance contributed by each PC:

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

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

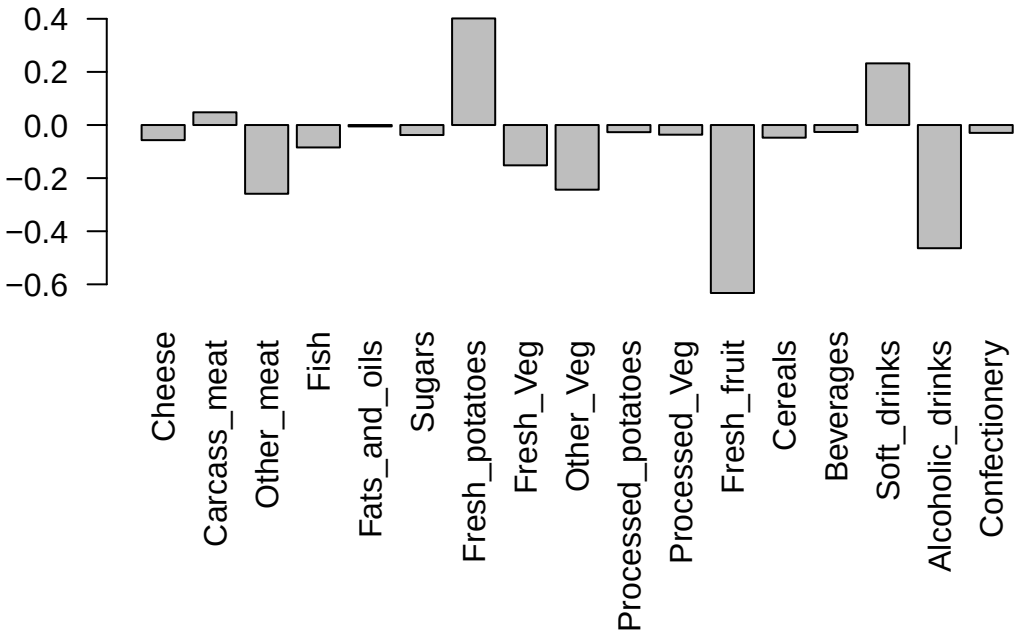
We can use a barplot to visualize how much each PC is contributing to the percent of variance:

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



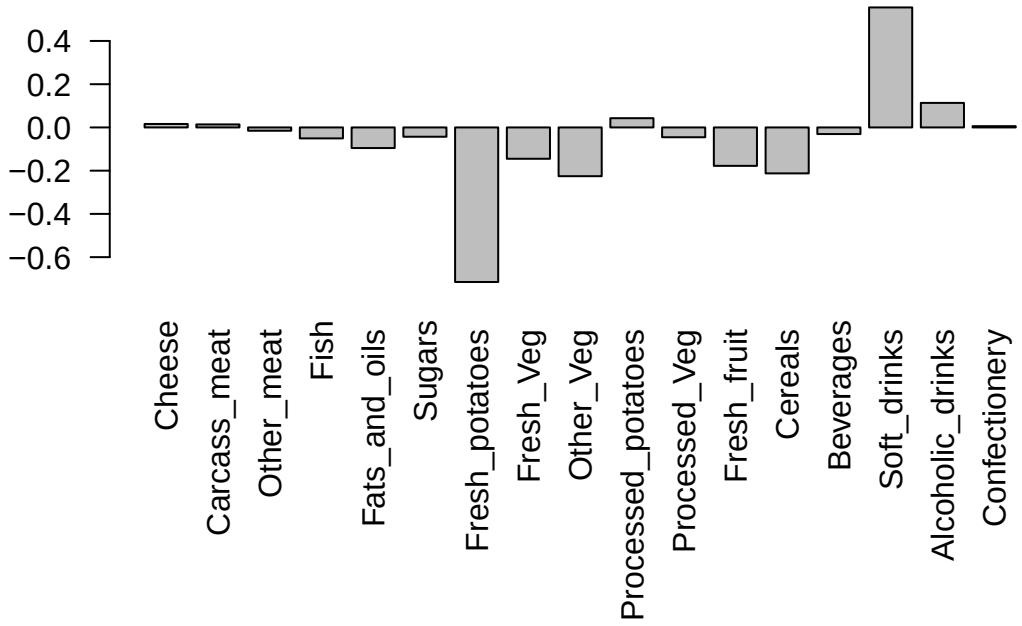
Let us now use loadings from `pca$rotation` to make a plot:

```
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 )
```



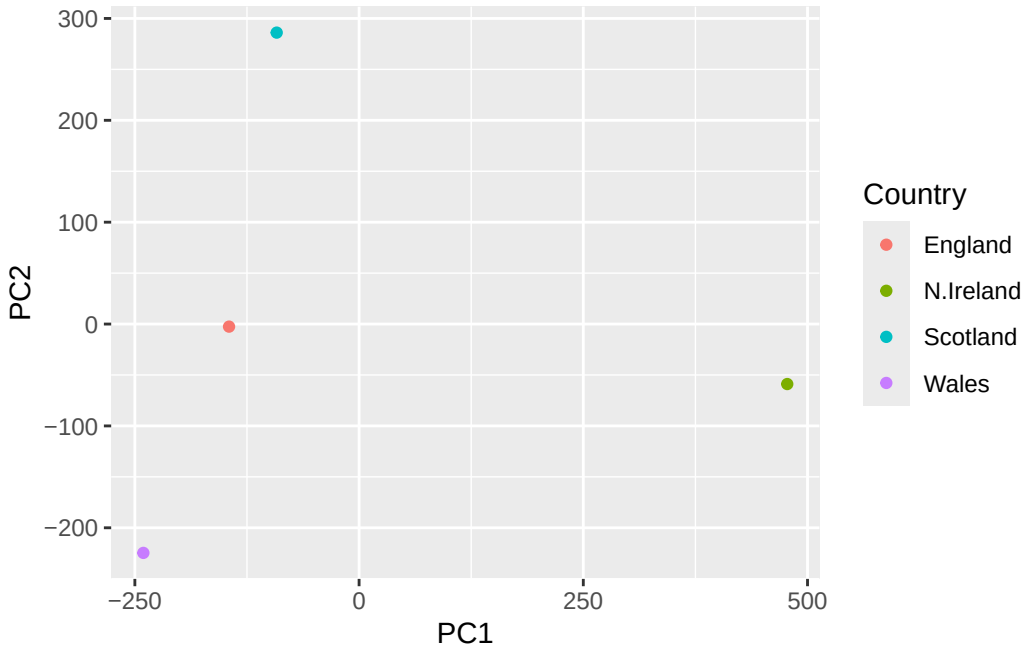
This tells us that in PC2, fresh potatoes have a strong negative (leftwards) push, while soft drinks have a strong positive (rightwards) push on the countries.

#ggplot of these data:

```
#install.packages("tibble")
library(ggplot2)
```

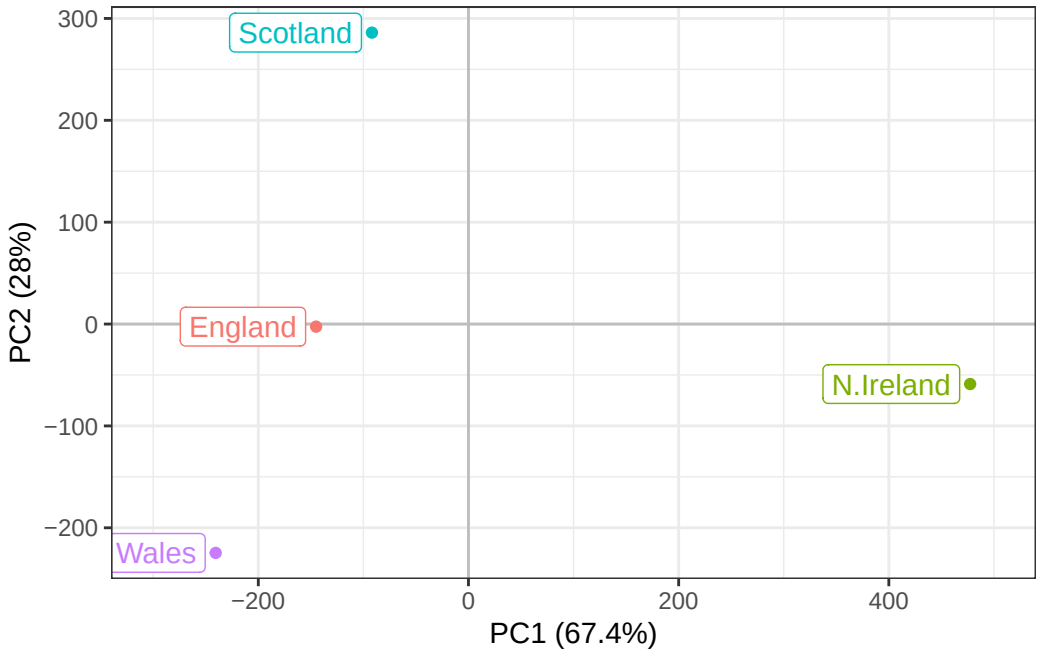
```
Warning: package 'ggplot2' was built under R version 4.3.3
```

```
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()
```



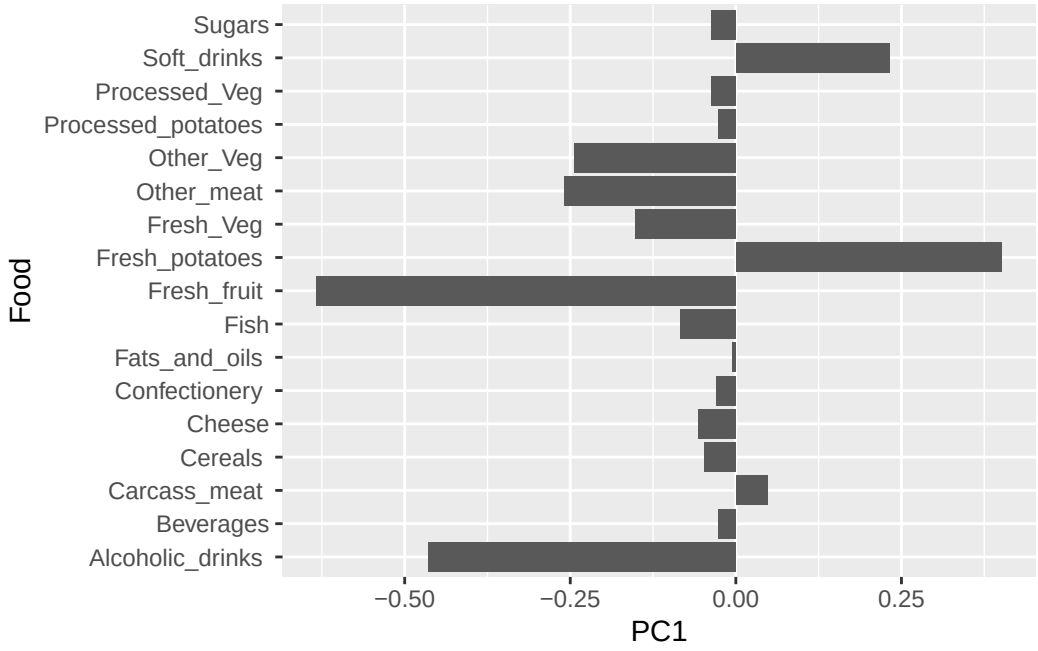
Making this plot look nicer:

```
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()
```

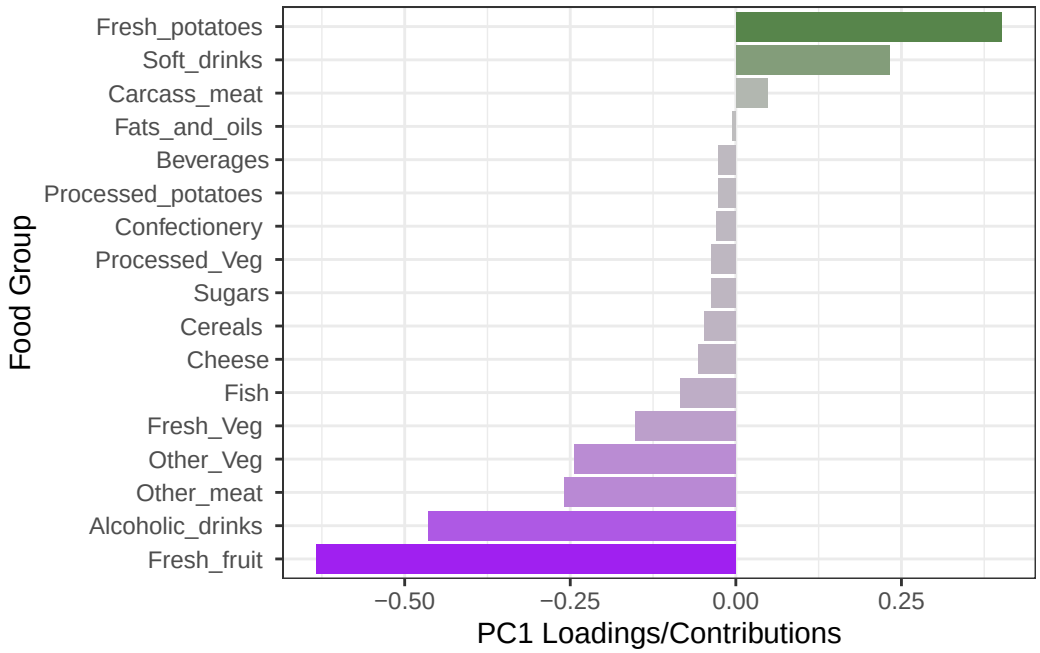


We can also make an updated version of our loadings plot:

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

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

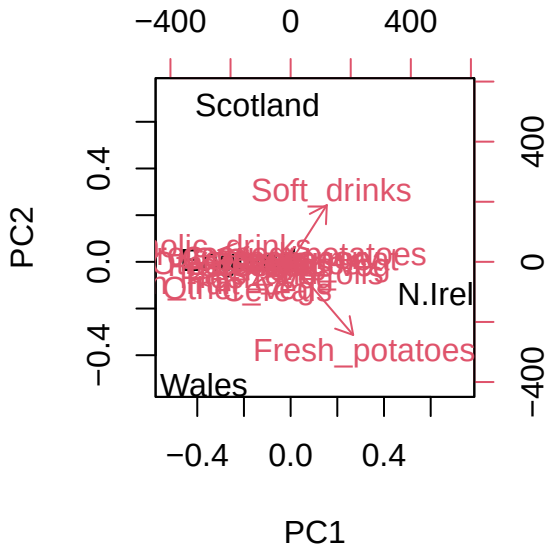



```
ggplot(ld_lab) +
  aes(PC1, reorder(Food, PC1), bg=PC1) +
  geom_col() +
  xlab("PC1 Loadings/Contributions") +
  ylab("Food Group") +
  scale_fill_gradient2(low="purple", mid="gray", high="darkgreen", guide=NULL) +
  theme_bw()
```



The biplot function shows the magnitude and directionality of each dimension and can be more useful for data with less categories:

```
biplot(pca)
```



PCA of RNA-seq data:

I will now complete a PCA analysis on data from RNA-sequencing:

```
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
```

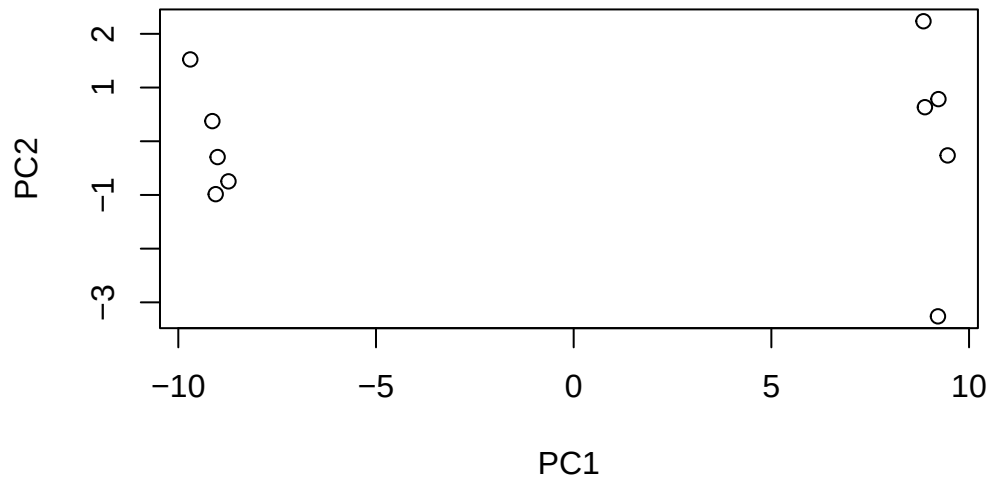
Q10: How many genes and samples are in this data set?

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

```
[1] 100  10
```

There are 10 samples and 100 genes in this data set. Let's plot a PCA to analyze!

```
pca1 <- prcomp(t(rna.data), scale=TRUE)

plot(pca1$x[,1], pca1$x[,2], xlab="PC1", ylab="PC2")
```

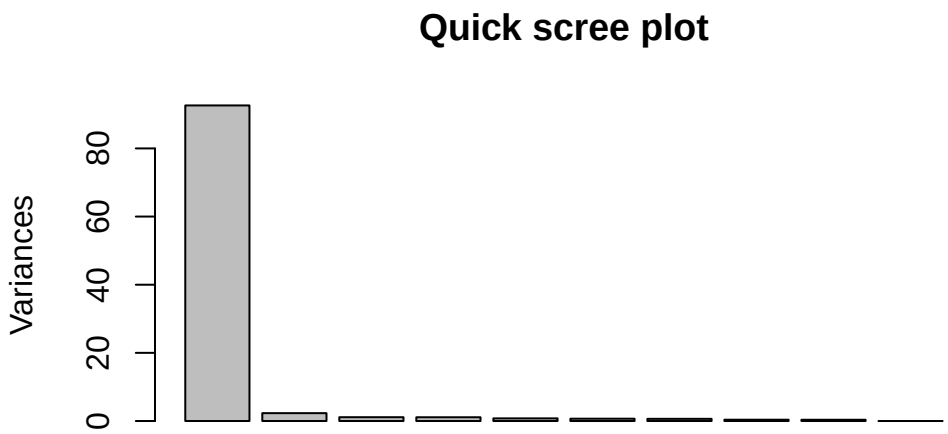



```
summary(pca1)
```

```
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 3.457e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

PC1 contains 92% of our variation! This can be seen very well through a scree plot:

```
plot(pca1, main="Quick scree plot")
```

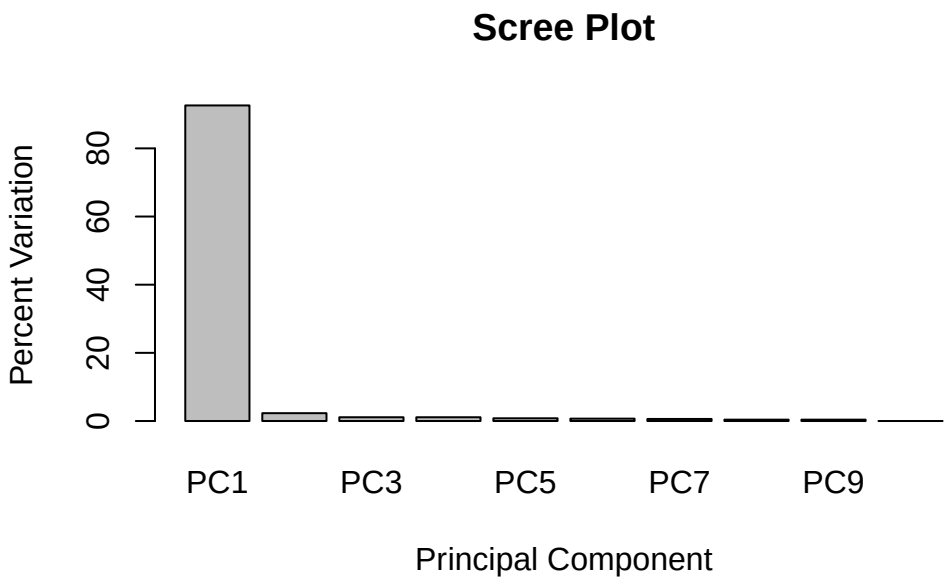


Let's improve this scree plot:

```
pca1.var <- pca1$sdev^2

pca1.var.per <- round(pca1.var/sum(pca1.var)*100, 1)
pca1.var.per
```

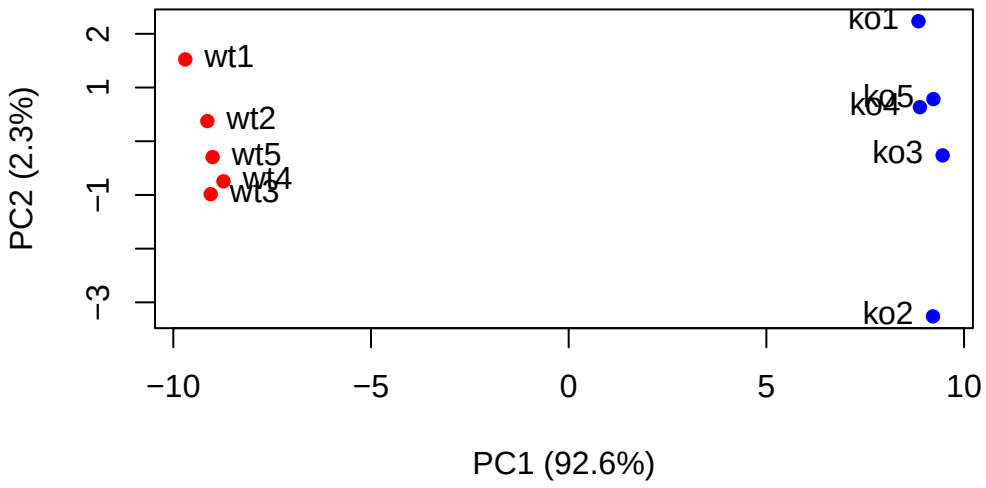
```
 [1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

```
barplot(pca1.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



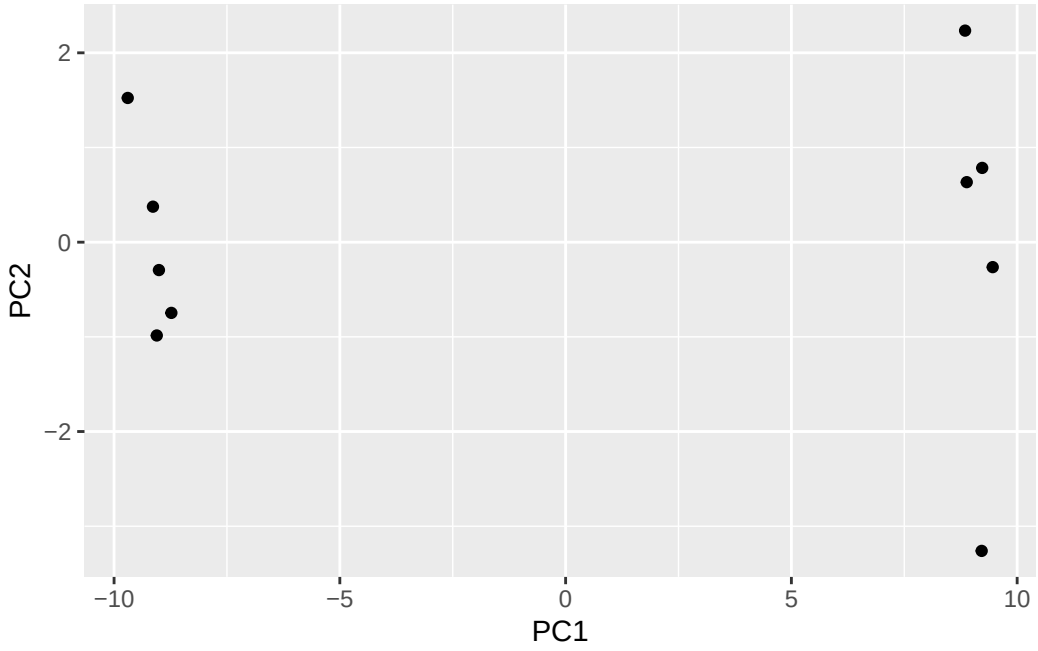
Using base R we can improve our PCA plot:

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

plot(pca1$x[,1], pca1$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca1.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca1.var.per[2], "%)"))

text(pca1$x[,1], pca1$x[,2], labels = colnames(rna.data), pos=c(rep(4,5), rep(2,5)))
```

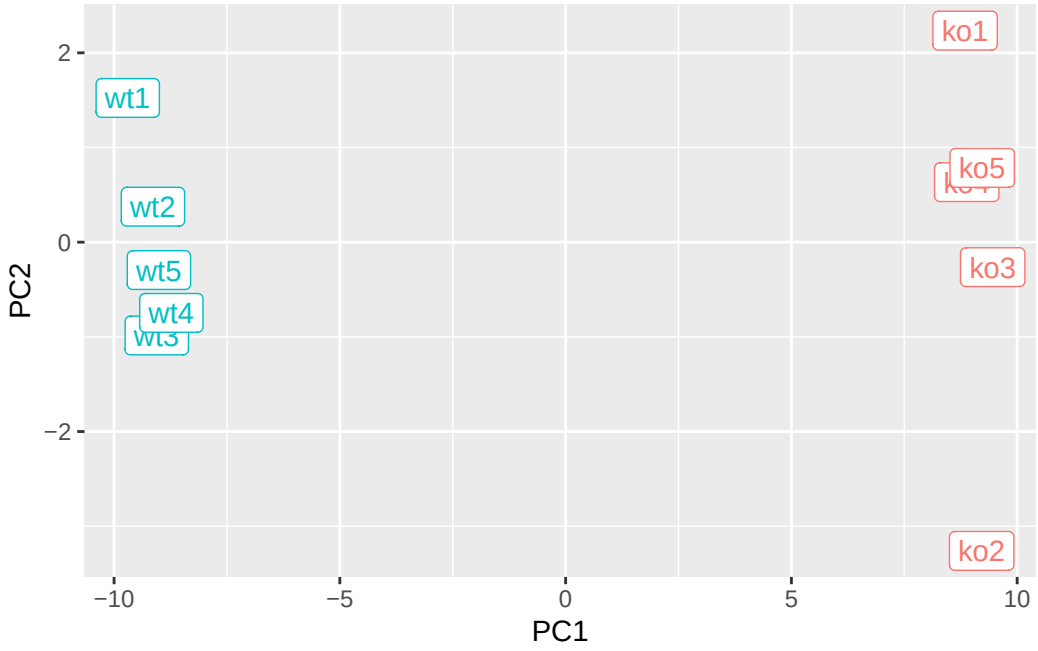


We can make a much more professional and attractive version of this using ggplot:

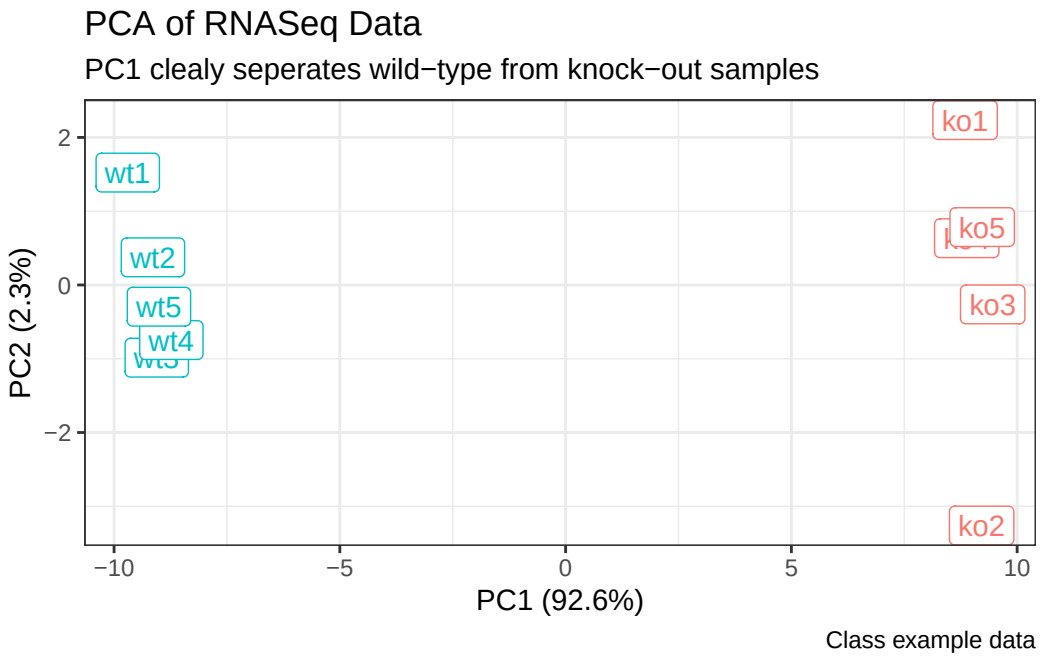
```
df1 <- as.data.frame(pca1$x)

ggplot(df1) +
  aes(PC1, PC2) +
  geom_point()
```




```
df1$samples <- colnames(rna.data)
df1$condition <- substr(colnames(rna.data),1,2)

p <- ggplot(df1) +
        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 (", pca1.var.per[1], "%)"),
       y=paste0("PC2 (", pca1.var.per[2], "%)"),
       caption="Class example data") +
     theme_bw()
```



We can also isolate the top ten genes contributing to PC1, both positively and negatively:

```
loading_scores <- pca1$rotation[,1]


gene_scores <- abs(loading_scores)
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

top_10_genes <- names(gene_score_ranked[1:10])
top_10_genes
```

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