Week 6: Machine Learning with PCA

Divya Shetty (A15390408)

2/13/2022

Principal Component Analysis [PCA]

PCA of UK Food Data

The UK food data is imported from a given input csv file.

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

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
## 7      Fresh_potatoes     720   874      566      1033
## 8           Fresh_Veg     253   265      171       143
## 9           Other_Veg     488   570      418       355
## 10 Processed_potatoes     198   203      220       187
## 11      Processed_Veg     360   365      337       334
## 12        Fresh_fruit    1102  1137      957       674
## 13            Cereals    1472  1582     1462      1494
## 14          Beverages      57    73       53        47
## 15        Soft_drinks    1374  1256     1572      1506
## 16   Alcoholic_drinks     375   475      458       135
## 17      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  5
```

There are 17 rows and 5 columns in data frame 'x'. You can use dim(x) or nrow(x) and ncol(x) to determine this.

There should be 4 columns: one for each country. Data frame is adjusted accordingly.

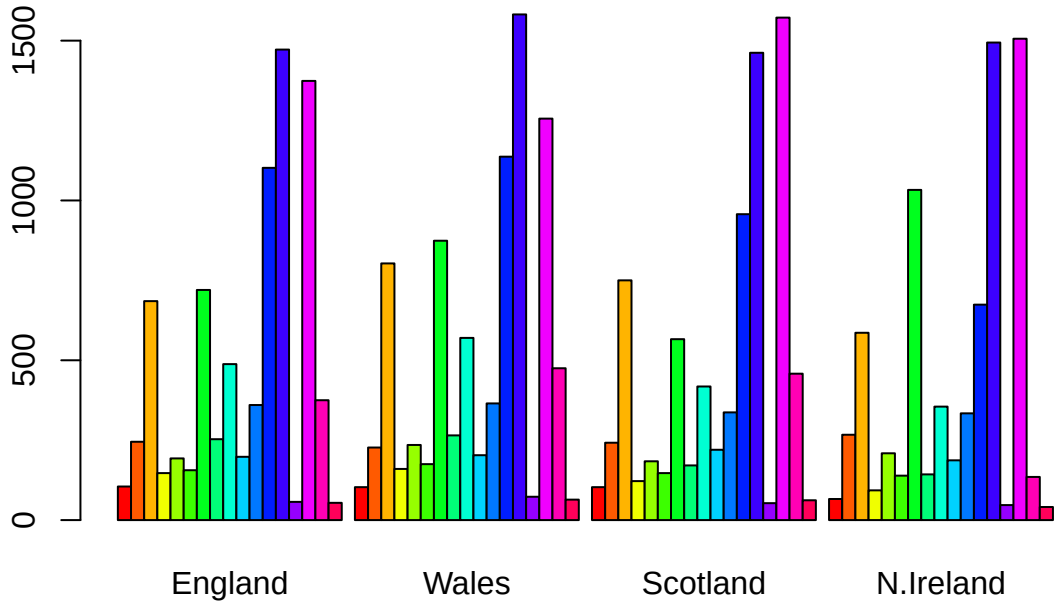
```
x <- read.csv(url, row.names = 1)

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
```

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 adjusting the row names in the read.csv() function since it requires less lines of code. It's also more robust because running "x <- x[,-1])" multiple times will continuously adjust the row names until there are no more columns left whereas editing read.csv() doesn't have this problem.

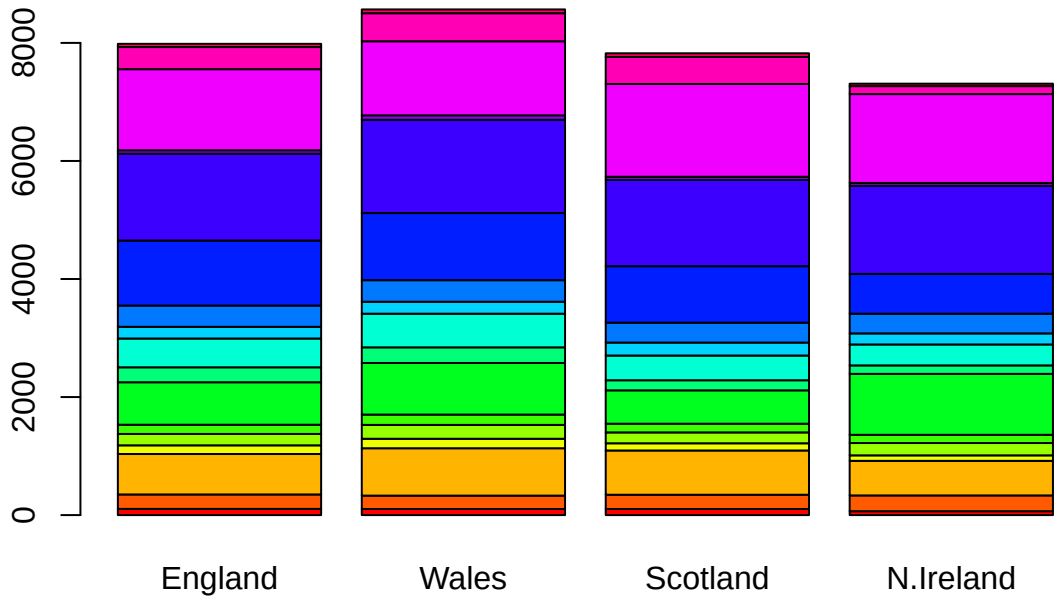
A few types of data visualizations are attempted, though they are uninformative.

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



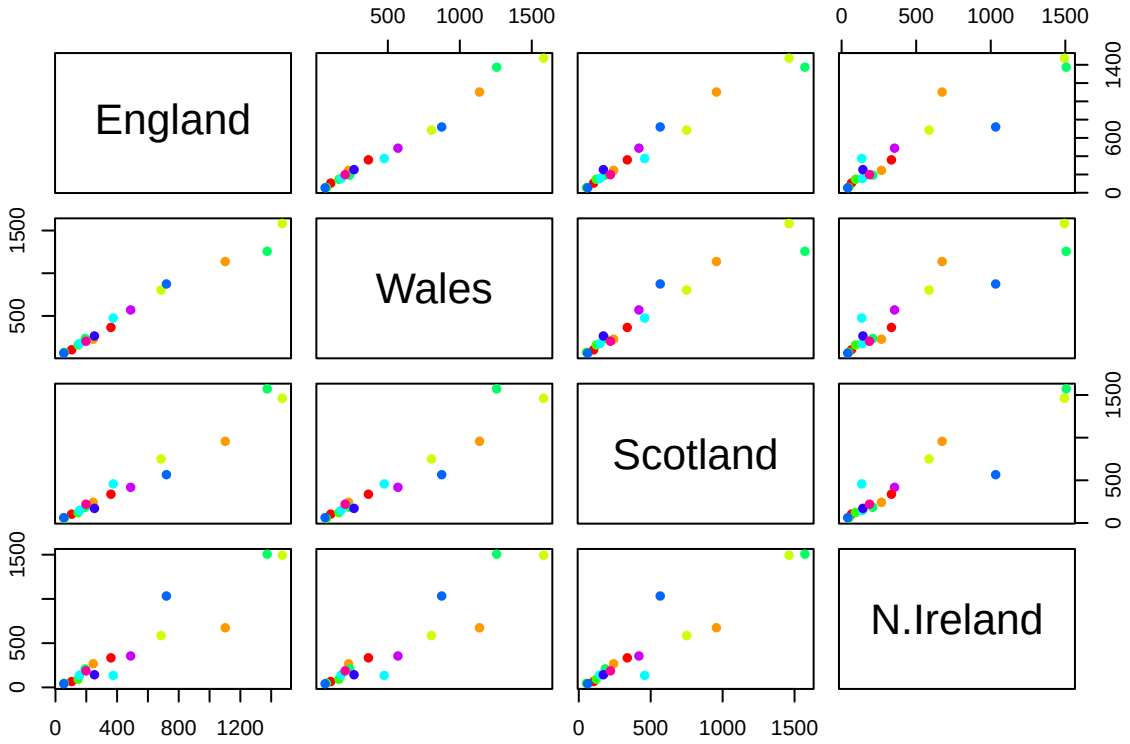
Q3: Changing what optional argument in the above barplot() function results in the following plot?
Change the value of beside to FALSE. This argument indicates whether the bar plot should have the columns side-by-side (TRUE) or have the stacked (FALSE)

```
barplot(as.matrix(x), beside = FALSE, 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?

```
#pairwise plot visualization
pairs(x, col = rainbow(10), pch = 16)
```



Each plot illustrates the data comparison between two given countries. A given point on the diagonal of a given plot means the same amount of some food is consumed between the two countries.

Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?
N. Ireland appears to have more differences in food consumptions than the other countries compared to one another as shown by the N. Ireland plots having more points spread out.

PCA to the rescue! PCA helps with analysis of many variables or large datasets. The main base R function is "prcomp()". This functions expects rows are observations and columns are variables, so input must be transposed.

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

#extra info about pca
attributes(pca)
```

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

```
summary(pca)
```

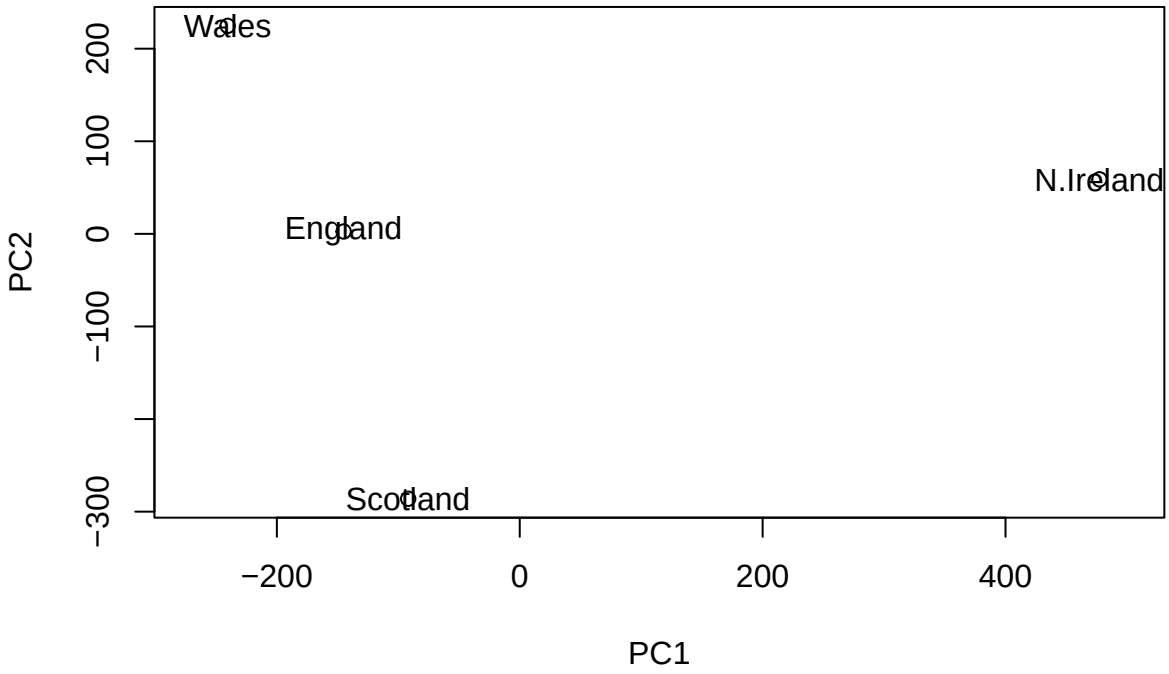
```
## Importance of components:
```

```
##                             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
```

To create our PCA plot, we must access the "pca$x" component.

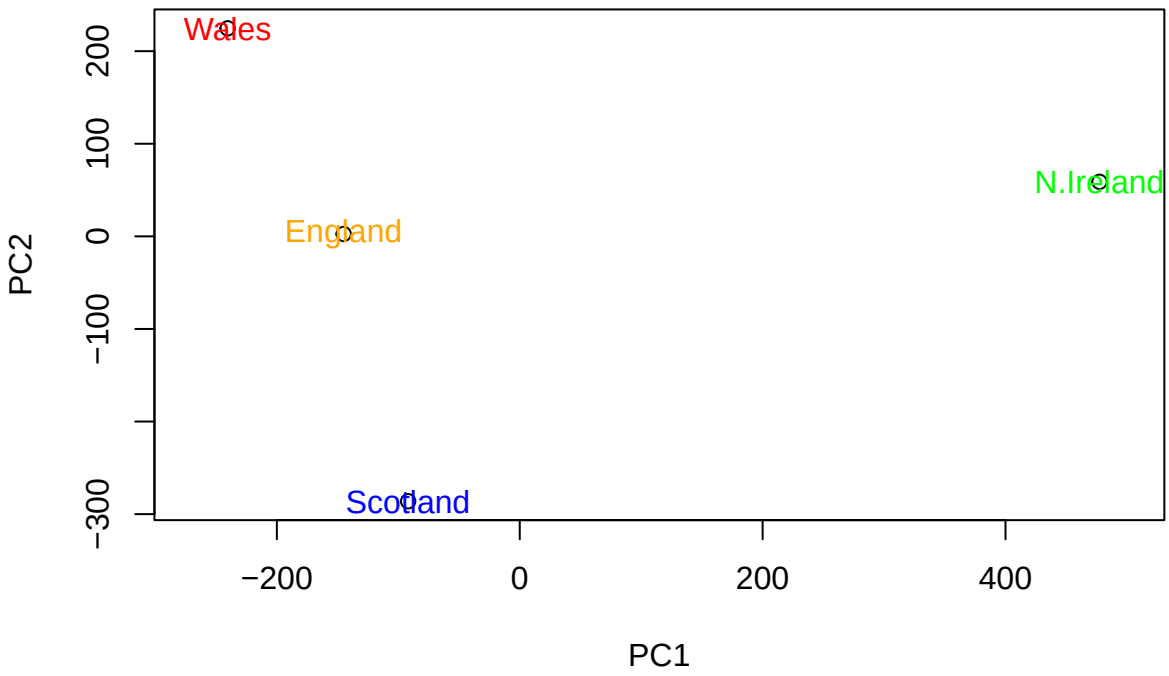
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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



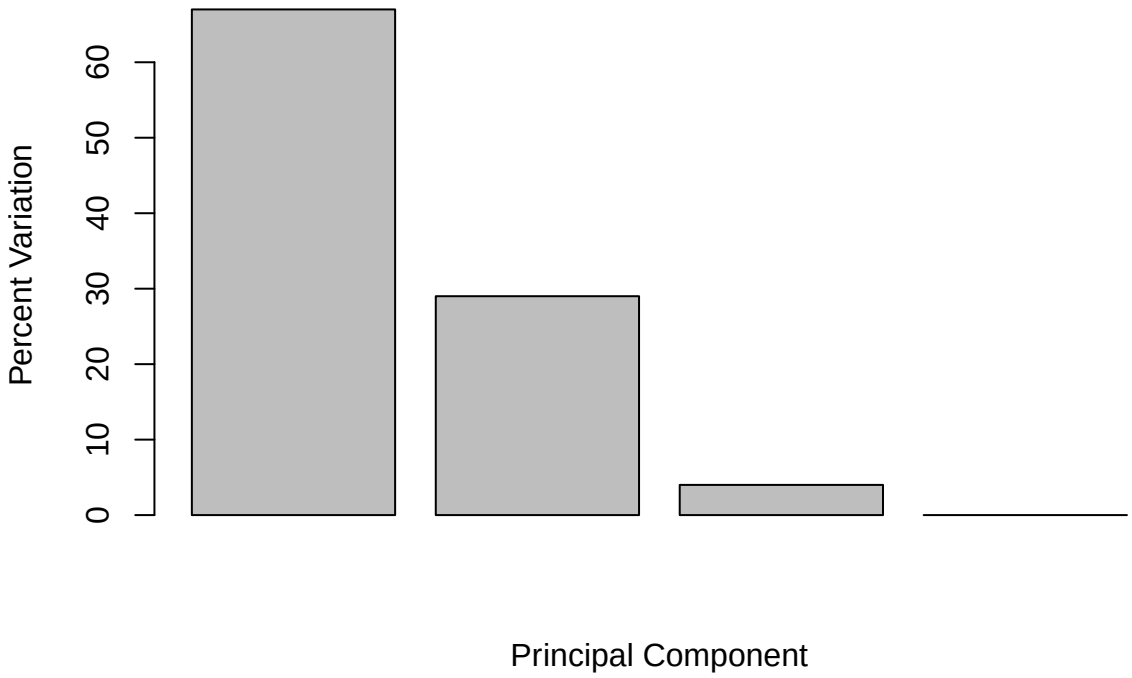
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
country_color <- c("orange", "red", "blue", "green")

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 = country_color)
```



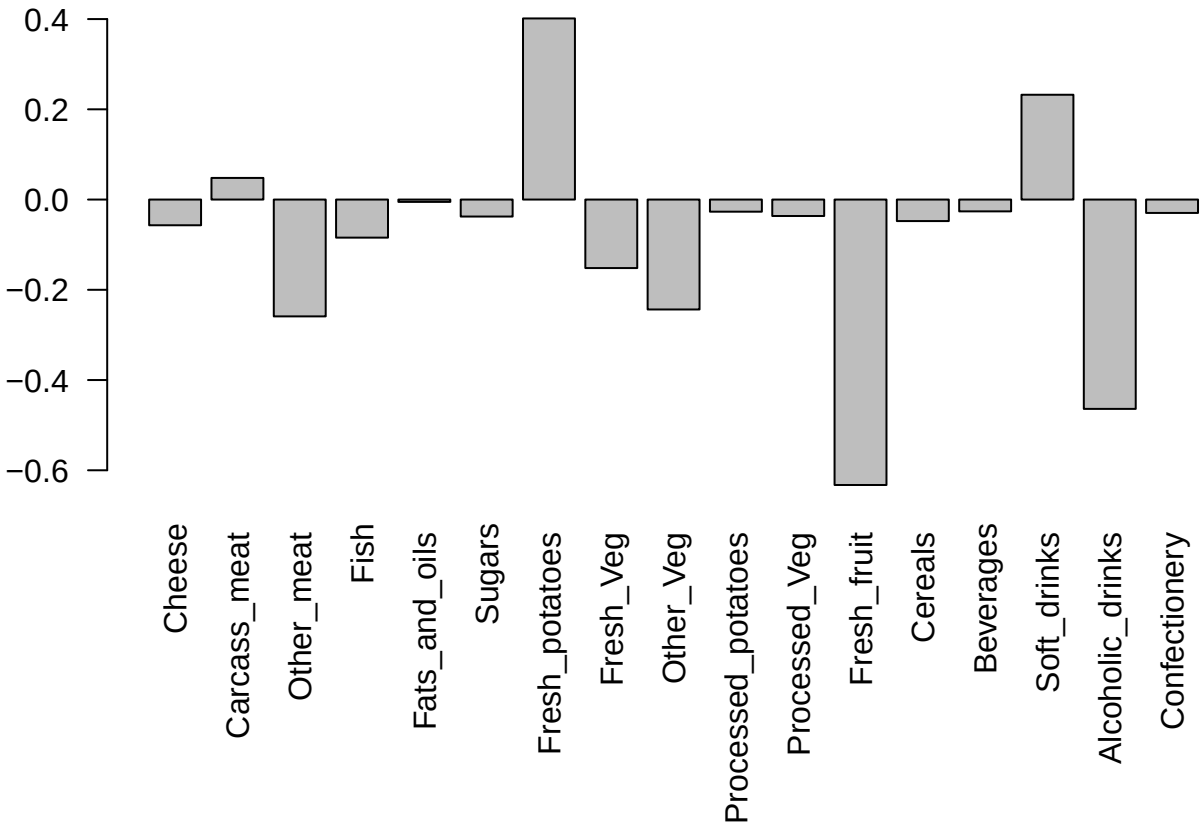
The square of "pca$sdev" can be used to calculate how much variation in the original data each PC accounts for.

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
barplot(v, xlab = "Principal Component", ylab = "Percent Variation")
```



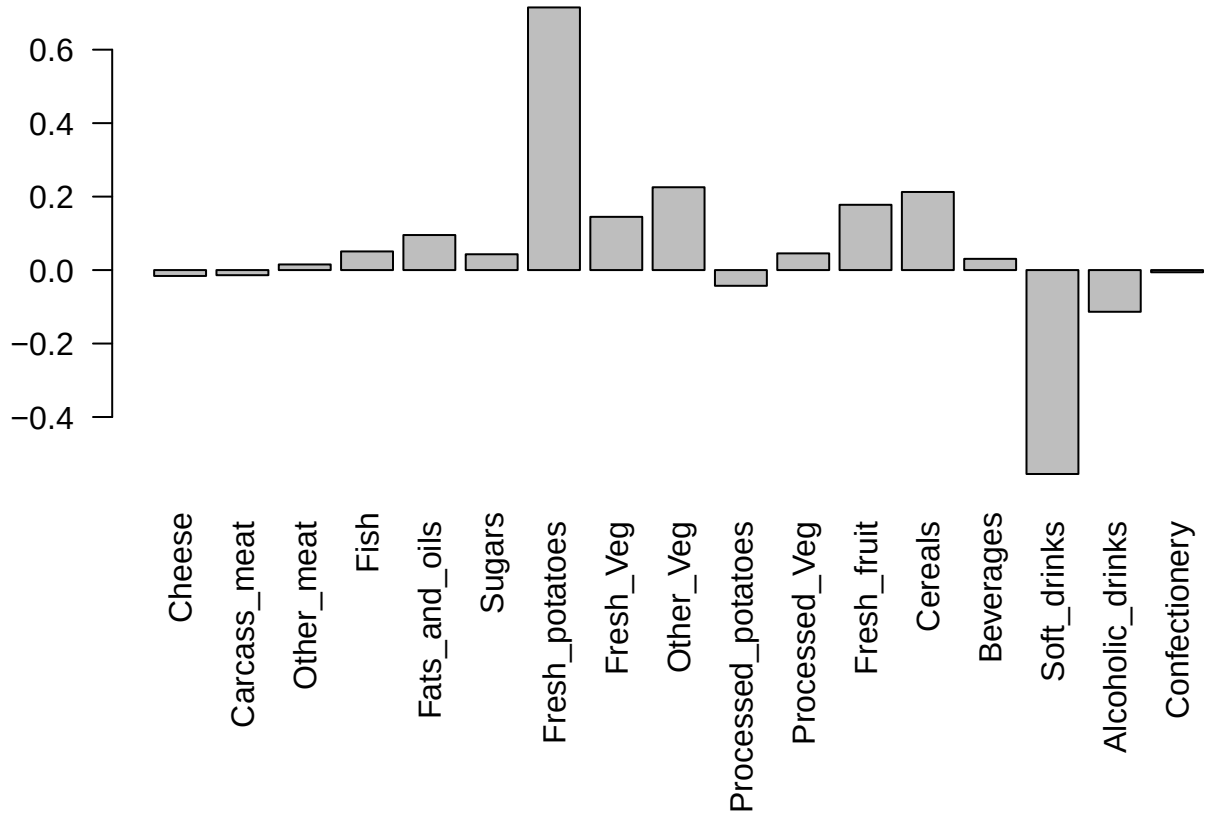
Digging deeper! The "pca$rotation" component can determine the influence of each of the original variables on the principal components ie. the loading scores.

```
#loadings plot for PC1
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 maninly tell us about?

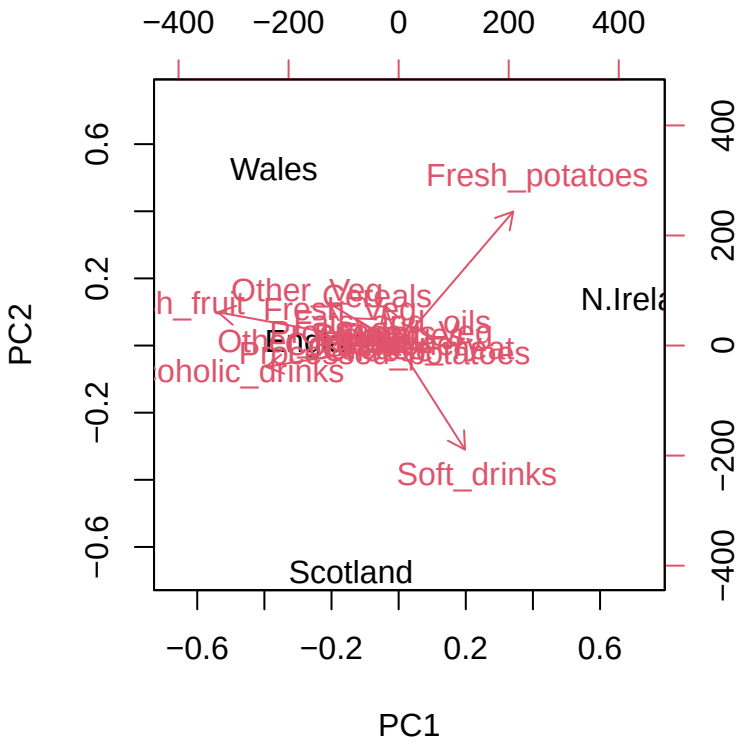
```
#loadings plot for PC2
par(mar = c(10, 3, 0.35, 0))
barplot(pca$rotation[,2], las = 2 )
```



Fresh_potatoes and Soft_drinks are both featured prominently in the plot for PC2. Since Fresh_potatoes has a positive score this indicates that this food "pushes countries upwards, and since Soft_drinks has a negative score, it"pushes" countries downwards in the PCA plot.

Biplots can also be used to visualize this information, usually for small datasets.

```
biplot(pca)
```



PCA of RNA-Seq Data

Read the RNA-seq count dataset into the "rna.data" data frame.

```
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?

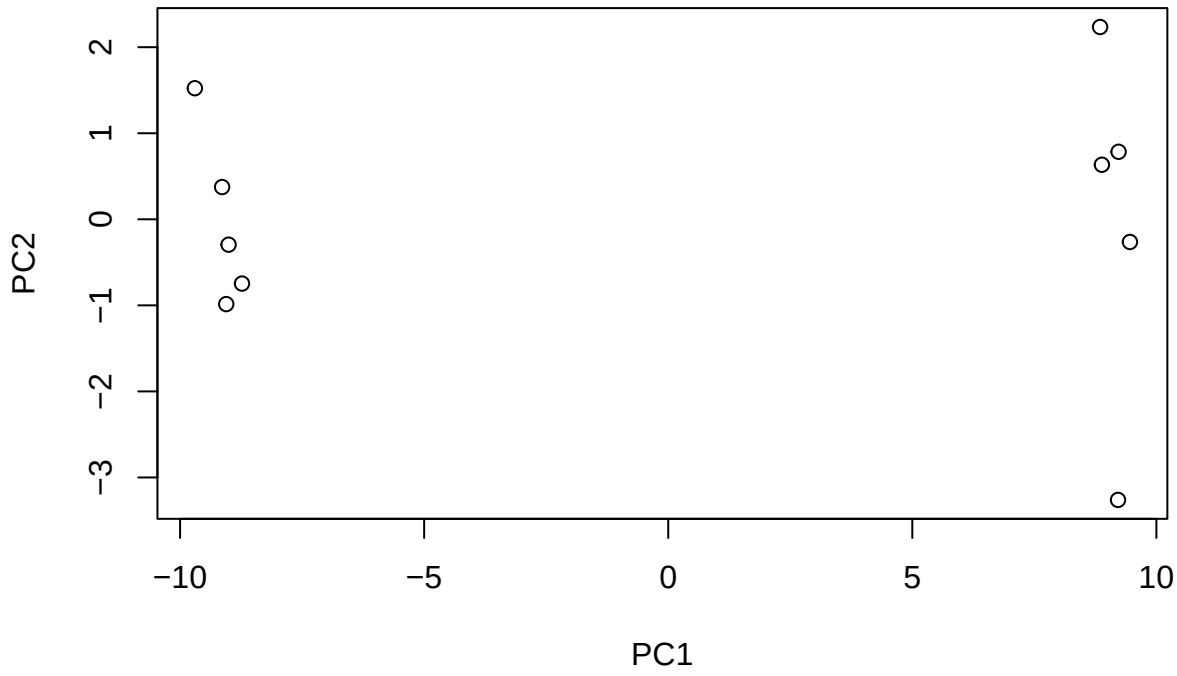
```
dim(rna.data)
```

```
## [1] 100  10
```

There are 10 genes and 100 samples.

A PCA plot is generated for the above data. Remember, the data must be transposed! A summary is generated to show how much variation in the original data each PC accounts for.

```
pca2 <- prcomp(t(rna.data), scale = TRUE)
plot(pca2$x[,1], pca2$x[,2], xlab = "PC1", ylab = "PC2")
```



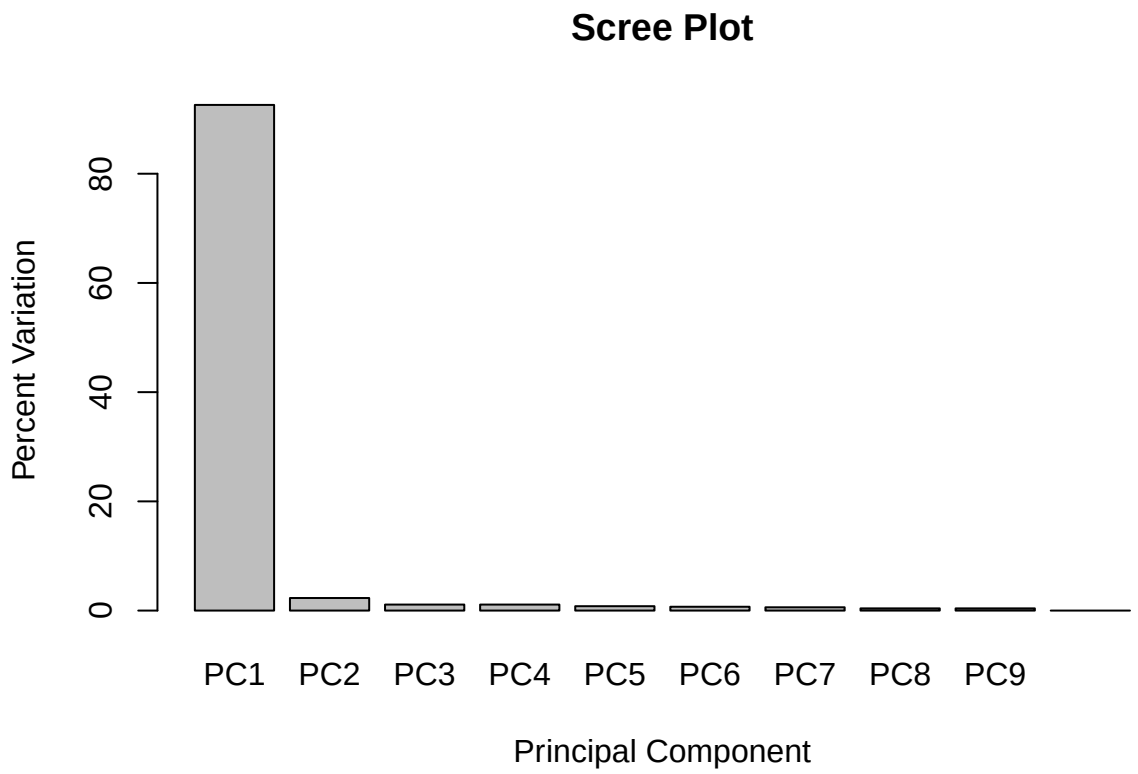

```
#variation info
summary(pca2)
```

```
## 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
```

Using the information provided from the summary, a scree plot can be generated to visualize the how much variation in the original data each PC accounts for.

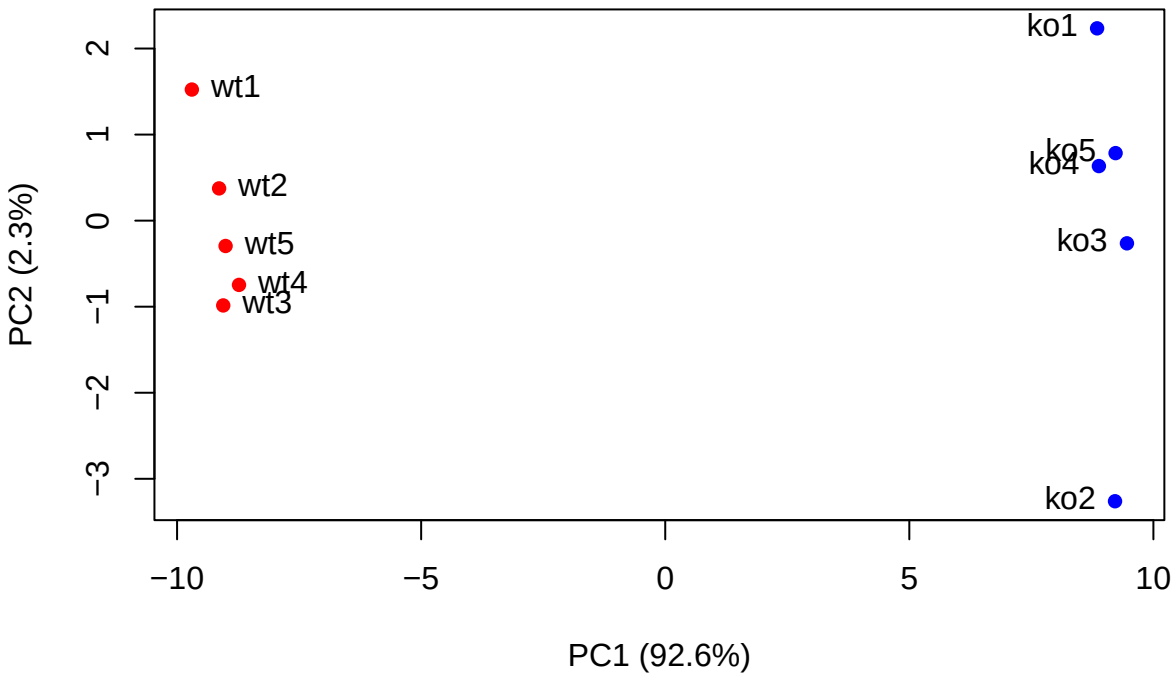
```
#variance per PC
pca2.var <- pca2$sdev^2
```

```r
#percent variance bc more informative to look at
pca2.var.per <- round(pca2.var/sum(pca2.var)*100, 1)

#make scree plot
barplot(pca2.var.per, main = "Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab = "Principal Component", ylab = "Percent Variation")
```



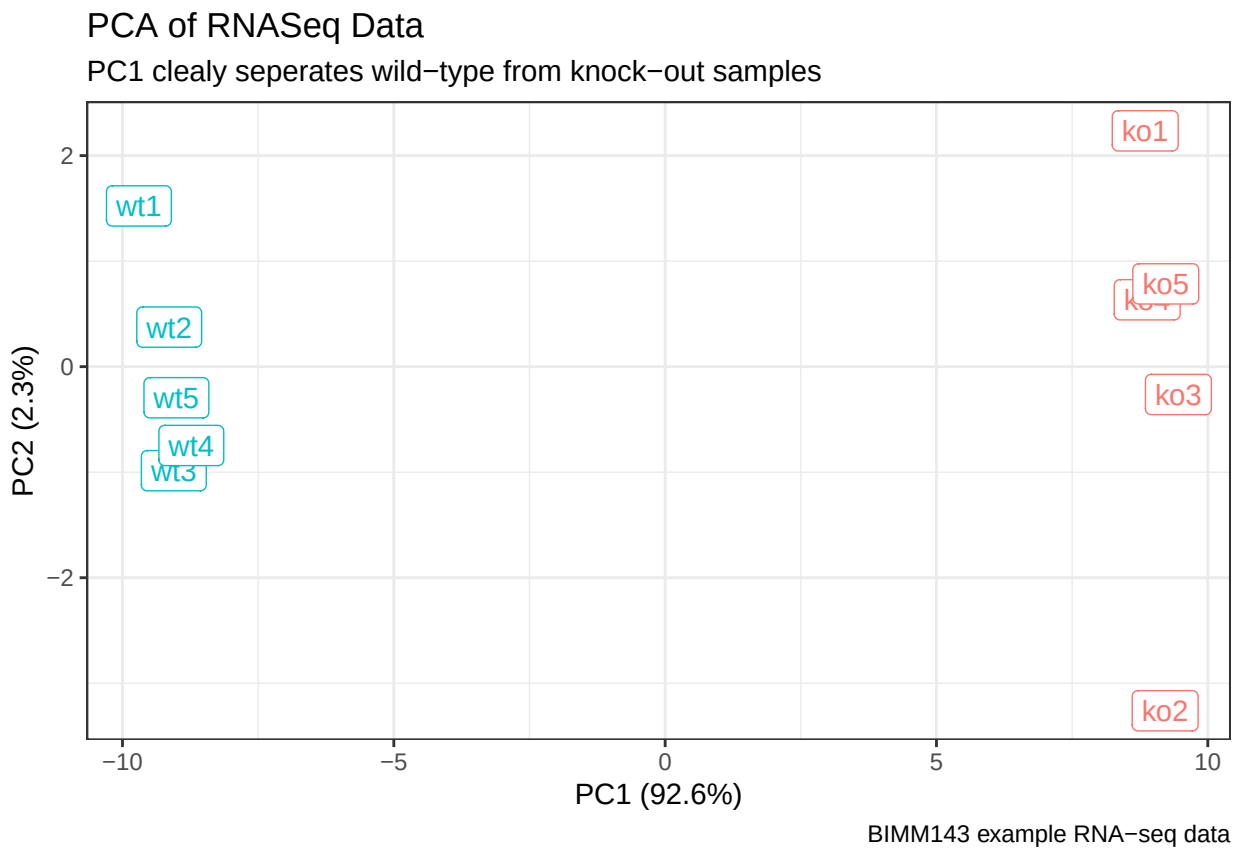
Now! Make the PCA plot look more presentable.

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

#make PCA plot
plot(pca2$x[,1], pca2$x[,2], col = colvec, pch = 16,
     xlab = paste0("PC1 (", pca2.var.per[1], "%)"),
     ylab = paste0("PC2 (", pca2.var.per[2], "%)"))

#labels
text(pca2$x[,1], pca2$x[,2], labels = colnames(rna.data), pos = c(rep(4,5), rep(2,5)))
```



We can also use ggplot2 to visualize the data in a different format.

```
library(ggplot2)

#convert PCA data into data frame with 'wt' and 'ko'
df <- as.data.frame(pca2$x)
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)

#ggplot visualization
ggplot(df) +
    aes(PC1, PC2, label = samples, col = condition) +
    geom_label(show.legend = FALSE) +
    labs(title="PCA of RNASeq Data",
       subtitle = "PC1 clealy seperates wild-type from knock-out samples",
       x = paste0("PC1 (", pca2.var.per[1], "%)"),
       y = paste0("PC2 (", pca2.var.per[2], "%)"),
       caption="BIMM143 example RNA-seq data") +
    theme_bw()
```



OPTIONAL: Finding the top ten genes that contribute the most to PC1

```
loading_scores <- pca2$rotation[,1]

#greatest to least abs val to find top 10 contributors
#that's positive (+) OR negative (-)
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] "gene100" "gene66"  "gene45"  "gene68"  "gene98"  "gene60"  "gene21"
##  [8] "gene56"  "gene10"  "gene90"
```