class08

Sanghun Lee

June 10, 2019

### Visualizing data with UK food data. Principal Component Analysis (PCA) is used here.

uk.foods <- read.csv("UK\_foods.csv")

Viewing and trimming our UK foods data using R

dim(uk.foods)

## [1] 17 5

summary(uk.foods) # Presence of "x" in the first column.

## X England Wales Scotland   
## Alcoholic\_drinks : 1 Min. : 54.0 Min. : 64.0 Min. : 53.0   
## Beverages : 1 1st Qu.: 156.0 1st Qu.: 175.0 1st Qu.: 147.0   
## Carcass\_meat : 1 Median : 253.0 Median : 265.0 Median : 242.0   
## Cereals : 1 Mean : 469.6 Mean : 503.9 Mean : 460.2   
## Cheese : 1 3rd Qu.: 685.0 3rd Qu.: 803.0 3rd Qu.: 566.0   
## Confectionery : 1 Max. :1472.0 Max. :1582.0 Max. :1572.0   
## (Other) :11   
## N.Ireland   
## Min. : 41.0   
## 1st Qu.: 135.0   
## Median : 209.0   
## Mean : 429.9   
## 3rd Qu.: 586.0   
## Max. :1506.0   
##

rownames(uk.foods) <- uk.foods[,1]  
uk.foods <- uk.foods[,-1]  
  
head(uk.foods) # x is gone.

## 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

The rownames function allows us to set the row names of our data frame as the first column values of this data frame. In this case, that would be the various kinds of food (Cheese, Carcass\_meat, etc.).

After that, we remove the troublesome first row, with the header X and the numbers for rows.

The following code accomplishes the exact same goal, albeit more efficient.

uk.foods <- read.csv("UK\_foods.csv", row.names = 1)  
  
uk.foods

## 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

#### Class Question: 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?

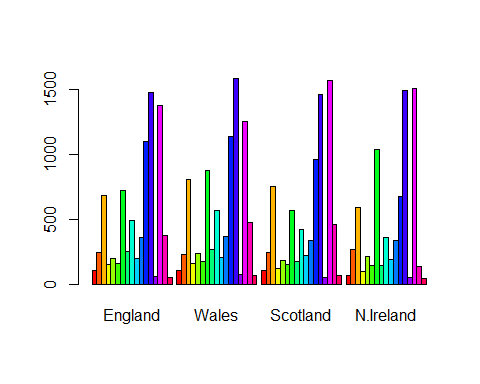
The first code chunk (uk.foods <- uk.foods[,-1]) can delete columns that we may need.

The second code chunk (read.csv(file.name, row.names = n)) can be difficult to use if the data layout is more complicated than the example.

I personally prefer the 2nd method.

### Spotting major difference and trends

barplot(as.matrix(uk.foods), beside = T, col = rainbow(nrow(uk.foods)))



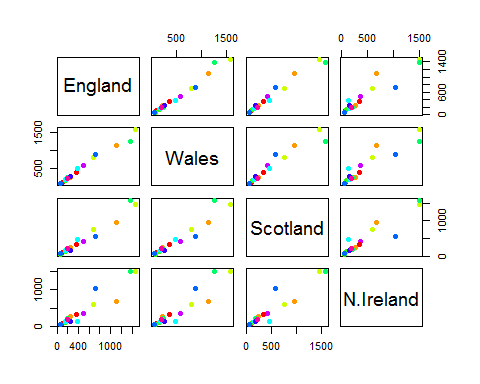
#### Class question: Changing what optional argument in the above barplot() function results in the following plot?

The “beside” parameter. This parameter allows us to visualize each data ‘besides’ each other. Setting this to FALSE will produce a stacked column chart.

#barplot(as.matrix(uk.foods), beside = F, col = rainbow(nrow(uk.foods)))

#### Class question: 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(uk.foods, col = rainbow(10), pch = 16)



### PCA to the rescue

Usage of prcomp() function and constructing a PCA dataset.

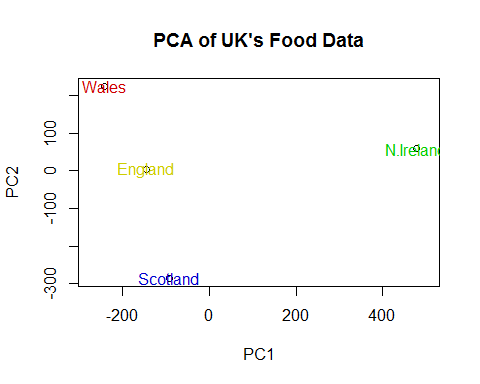
pca <- prcomp(t(uk.foods))  
summary(pca)

## 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

t() function is the “transpose” fucntion. This switches the row and the columns in a data frame.

Generating a plot of PC1 vs. PC2.

plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2", xlim = c(-270, 500),  
 main = "PCA of UK's Food Data")  
  
colvec = c("yellow3", "red3", 'blue3', 'green3')  
  
text(pca$x[,1], pca$x[,2], colnames(uk.foods), col = colvec)



With this chunk, I am understanding how much variation in each PC accounts for.

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

## [1] 67 29 4 0

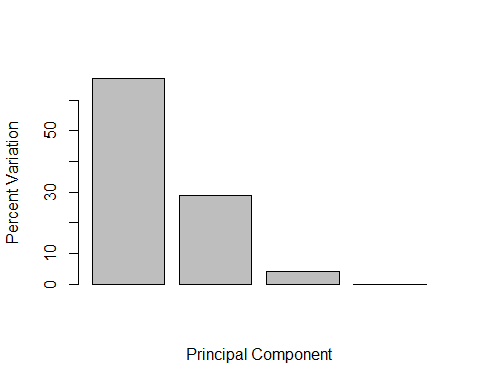
z <- summary(pca)  
z$importance

## 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

From the R code chunk, we can see that PC1 accounts for 67% of the variation in the data, and PC2 accounts for 29%. Generally speaking, almost all the variation in the data will be captured by PC1 and PC2.

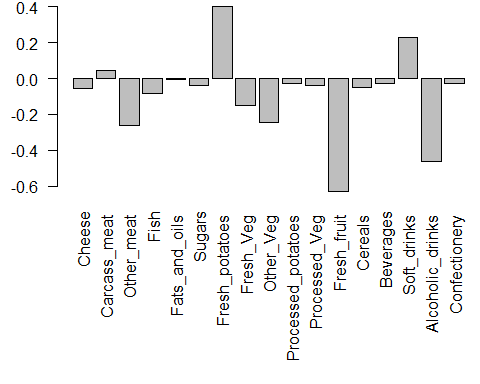
Graphing the variation in our PCA.

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



### Digging deeper into understanding PCA.

par(mar = c(10, 3, 0.35, 0))  
barplot(pca$rotation[,1], las = 2)

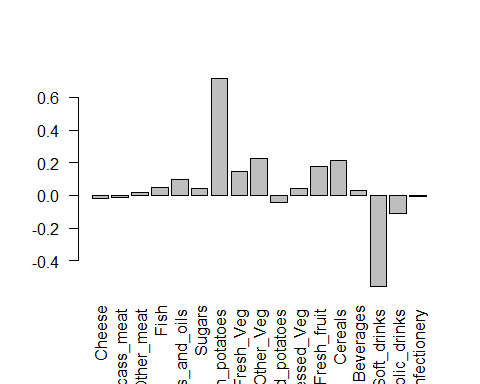


par(mar = c()) sets the plot’s dimensions. This is to clean up our barplot or plot function.

pca$rotation, or rotation, is called “loading scores”. This loading scores help us to determine which components have the largest effect on where they are plotted in the PCA plot.

So here, with pcarotation[,2], however, we see that soft drinks actually have a largely negative effect on PC2.

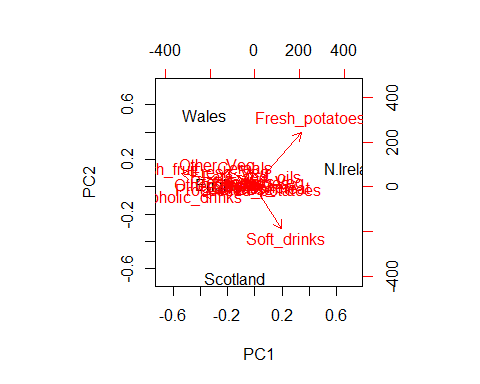
(barplot(pca$rotation[,2], las = 2))



## [,1]  
## [1,] 0.7  
## [2,] 1.9  
## [3,] 3.1  
## [4,] 4.3  
## [5,] 5.5  
## [6,] 6.7  
## [7,] 7.9  
## [8,] 9.1  
## [9,] 10.3  
## [10,] 11.5  
## [11,] 12.7  
## [12,] 13.9  
## [13,] 15.1  
## [14,] 16.3  
## [15,] 17.5  
## [16,] 18.7  
## [17,] 19.9

Constructing Biplots

biplot(pca)



A biplot represents a sample as a point on the graph (Eng, Scot, Wales, N.Ire), and variables as a vector (cheese, carcass\_meat, confectionary, etc.).

### PCA of RNA-Seq data

rna.data <- read.csv("expression.csv", 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

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

nrow(rna.data)

## [1] 100

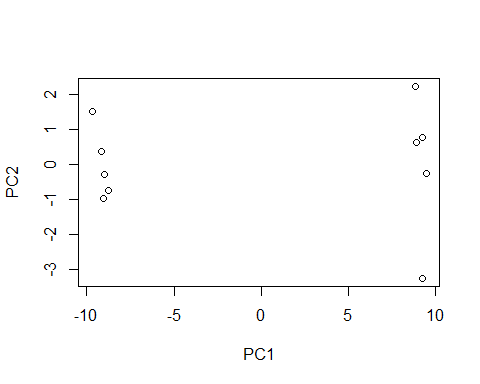
ncol(rna.data)

## [1] 10

We have 100 genes each with 10 samples.

#### Generating a plot for this new RNA-Seq data.

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



Interesting how there is a clear divide between two groups of data frame.

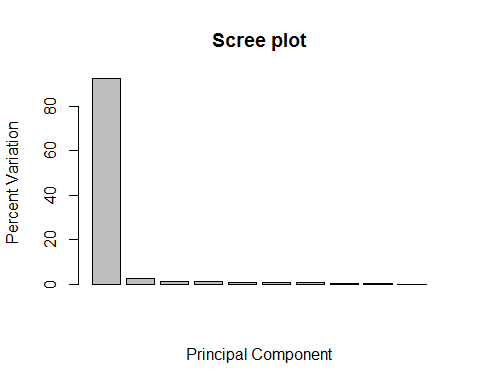
PCA variance percentage calculation

pca.var <- pca.rna$sdev^2  
pca.var.per <- round(pca.var/sum(pca.var)\*100, 1)  
  
pca.var.per

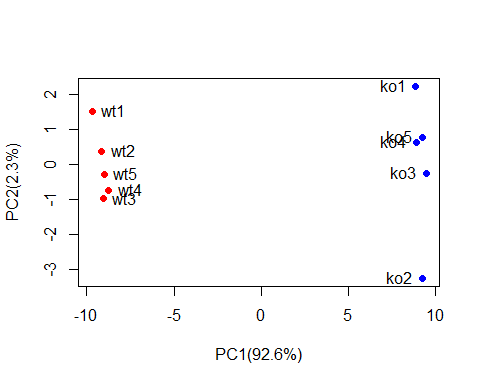
## [1] 92.6 2.3 1.1 1.1 0.8 0.7 0.6 0.4 0.4 0.0

Barplot of PCA variance percentage

barplot(pca.var.per, main = 'Scree plot',  
 xlab = "Principal Component",  
 ylab = "Percent Variation")

 We can see from this that all the action (variance) is in PC1. PC1 accounts for almost 93% of the variance in the data.

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)))

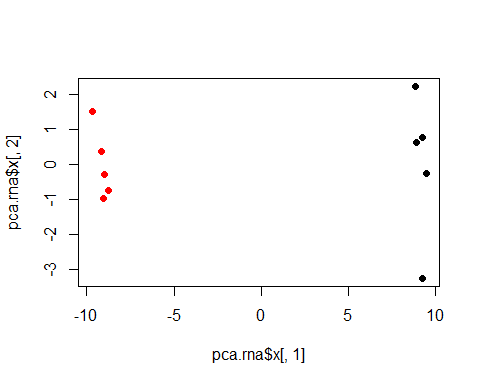


Easier method to color in our dot plot.

sample.type <- substr(colnames(rna.data), 1,2)  
sample.type

## [1] "wt" "wt" "wt" "wt" "wt" "ko" "ko" "ko" "ko" "ko"

plot(pca.rna$x[,1], pca.rna$x[,2], col = as.factor(sample.type), pch = 16)



Finding genes that contribute most to our PCA plot.

gene.scores <- abs(pca.rna$rotation[,1])  
gene.scores.ranked <- sort(gene.scores, decreasing = T)  
  
top.10.genes <- names(gene.scores.ranked[1:10])  
top.10.genes

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

The above genes are affected by an abs() function, which applies an absolute value to all the values in a vector.

This is because a gene with a rotation score of 2.6 has a lesser effect on PCA when compared to a gene with a rotation score of -10.6.

#### Conclusion

Lecture 08 was a great lesson in data visualization and analysis with R. Principal Component Analysis can be done with R, and I learned the key components of PCA.

PCA clusters like data points together. Since Principal Components 1 and 2 explain almost all of the variation in a data, an analyst can use R to see the clustering of datapoints on a plot. PC1 captures the direction where the most variation is, and PC2 the 2nd most variation, and so on.

Additionally, biplots can be used to see not only the sample (represented by a point), but the variables in a dataset (represented by a vector).