lab7_Lauren

Lauren PID A53280444

Table of contents

Class 7: Unsupervised Learning and Dimensional Reduction 1
- K means Clustering 4
- for loop in R 8
- Hierarchical Clustering 9
- Principal Component Analysis 12
  - UK food data 12
  - Running the PCA 13
  - Variable Loadings 17

Class 7: Unsupervised Learning and Dimensional Reduction

Setup

```r
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.1.1
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
v ggplot2 3.3.5     v purrr   0.3.4
v tibble  3.1.3     v dplyr   1.0.7
v tidyr   1.1.4     v stringr 1.4.0
v readr   2.1.1     v forcats 0.5.1
```

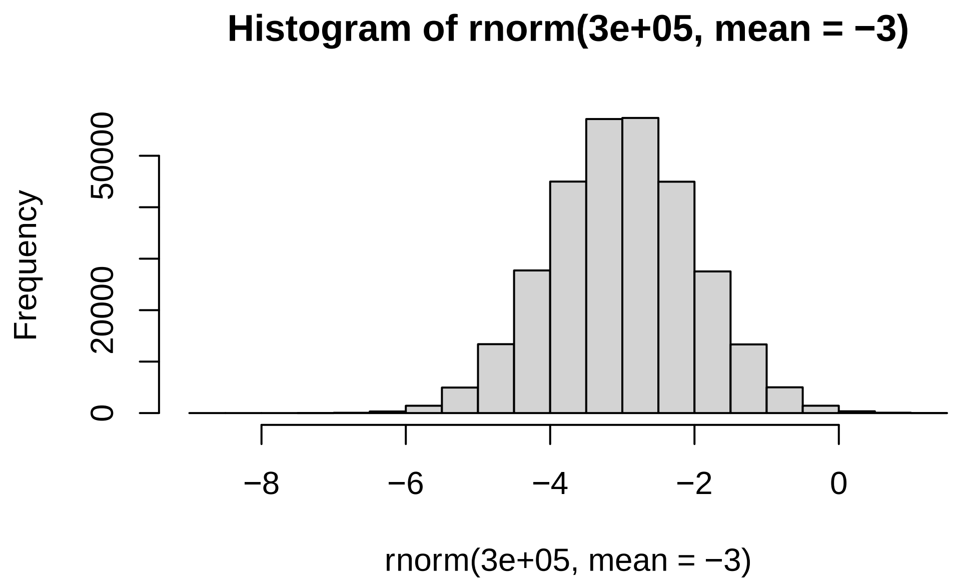
```
Warning: package 'tidyr' was built under R version 4.1.1
```

```
Warning: package 'readr' was built under R version 4.1.2

Warning: package 'purrr' was built under R version 4.1.1

Warning: package 'dplyr' was built under R version 4.1.1

Warning: package 'stringr' was built under R version 4.1.1

Warning: package 'forcats' was built under R version 4.1.1

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(ggplot2)
theme_set(theme_bw())
```

Setup random data: normally distributed

```
hist(rnorm(300000, mean = -3))
```



I want a small vector of 30 points, 2 groupings inside.

```
rnorm(30, mean = -3)
```

```
 [1] -2.914925 -2.879418 -3.999115 -3.073378 -3.670336 -4.858009 -5.523001
 [8] -2.474859 -3.844522 -1.344382 -2.744577 -2.435825 -2.247521 -1.820053
[15] -3.707977 -2.471757 -1.389225 -4.611010 -3.967947 -3.558287 -3.736233
[22] -2.271582 -2.633540 -2.579032 -1.570465 -1.811529 -4.743105 -4.151943
[29] -4.192348 -2.428269
```

More points centered at +3

```
rnorm(30, 3)
```

```
 [1] 2.866347 3.579156 2.929570 3.628346 2.936481 2.428164 1.743122 3.283682
 [9] 2.770548 2.624753 2.804646 3.522914 3.051543 2.993221 3.175758 2.005447
[17] 2.882475 3.484633 1.625376 2.953316 2.461160 4.566247 1.683684 2.139578
[25] 2.365410 3.749058 2.102497 1.766101 3.220234 2.843437
```

Put both into 1 vector

```
temp <- c(rnorm(30, -3), rnorm(30, 3))
```

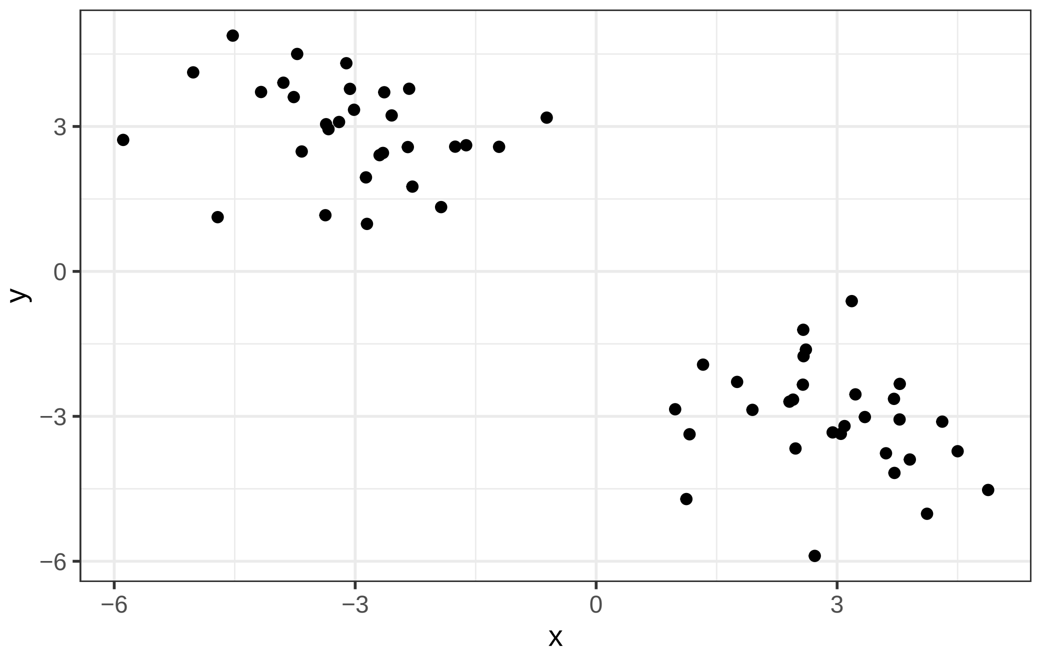
Make dataframe where x = -3 numbers then + 3 numbers, y is reverse order

```
x <- data.frame(x = temp, y = rev(temp))
head(x)
```

```
          x        y
1 -3.064841 3.777987
2 -5.888721 2.721689
3 -4.712100 1.123238
4 -3.362196 3.046463
5 -3.201217 3.092280
6 -1.618549 2.611367
```

Check that we get what we expect

```
ggplot(x, aes(x, y)) + geom_point()
```



K means Clustering

```
km <- kmeans(x, 2)
# centers equals the number of clusters
# clustering vector (below) is assigning points to each cluster (i.e. cluster 1 and 2)
# Within cluster sum of squares by cluster: how good algorithm is at assignment (read more
```

It’s important to not just run the analysis but be able to get the important results back!

Find the size of the clusters

```
km$size
```

```
[1] 30 30
```

```
# size = number of points in each cluster
```

Center of clusters

```
km$centers
```

```
         x        y
1 -3.07198  2.92839
2  2.92839 -3.07198
```

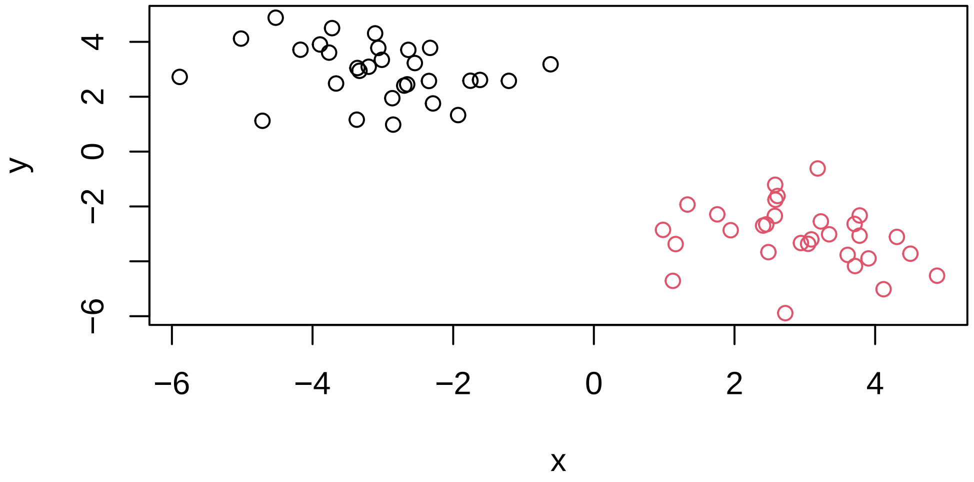
Where do I find the main result - the cluster assignment vector?

```
km$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 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
```

Can we make a summary figure showing our result? With the points colored by cluster assignment and add cluster centers?

```
plot(x, col = km$cluster)
```



The BIG PROBLEM with kmeans: you set the cluster #, so the output is somewhat self-determining

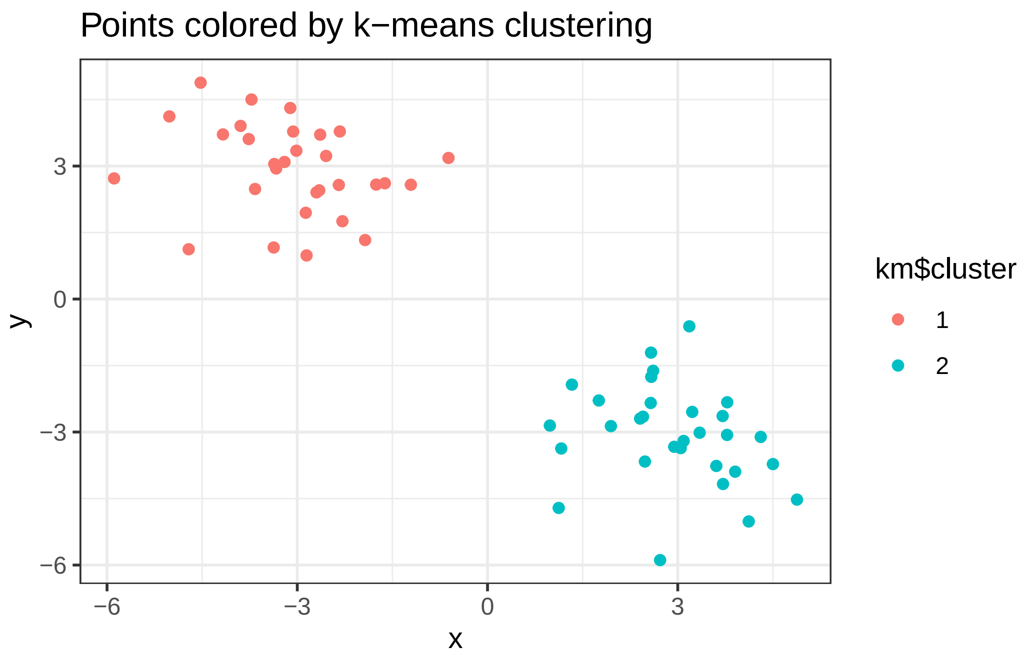
One way to check the right number of clusters: try multiple cluster #, then for each k check the value of tot.withinss, plot as a function of k, this is called a **Scree Plot**

Output of the scree plot: the "cliff face" or **inflection point** is the point that caused the most change (after which the tot.withinss doesn't drop much more it levels off)

If the Scree plot is linear or doesnt drop off much then your data can't be classified well

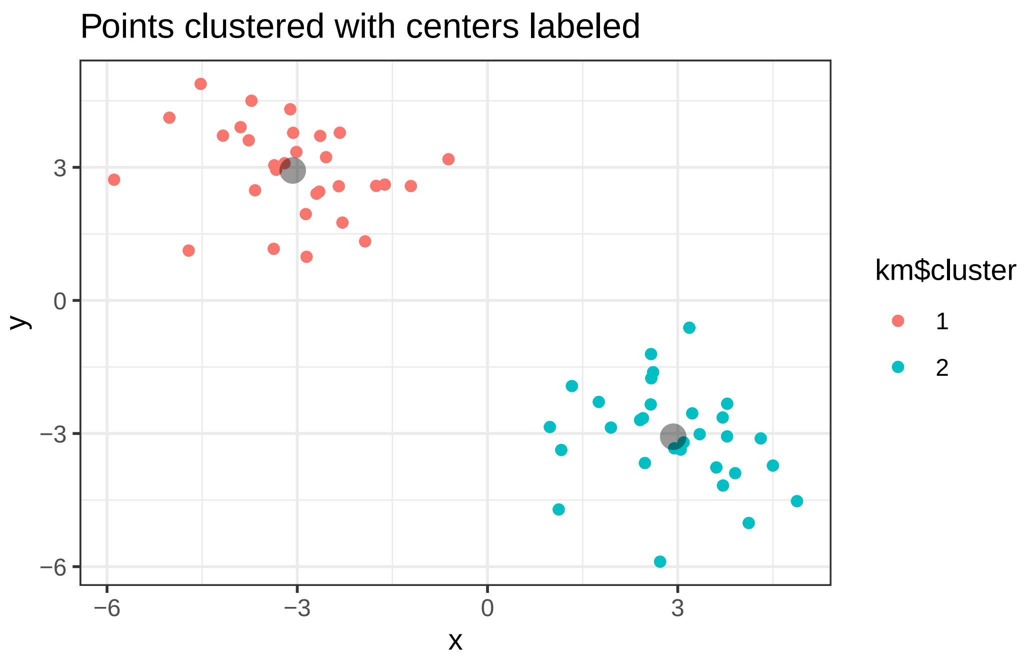
Ggplot version

```r
# set cluster as a factor variable
km$cluster <- as.factor(km$cluster)
# then plot
ggplot(data = x) +
    aes(x = x, y =y, color = km$cluster) +
    geom_point() +
    # geom_point(data = km$centers, aes(x = km$centers[,1], y = km$centers[,2])) +
    labs(title = "Points colored by k-means clustering")
```



How to plot the center points? - When plotting from two separate dataframes, need the first data argument in ggplot to be empty, put the data inside the aesthetics - Also, make sure to make km$centers into its own dataframe

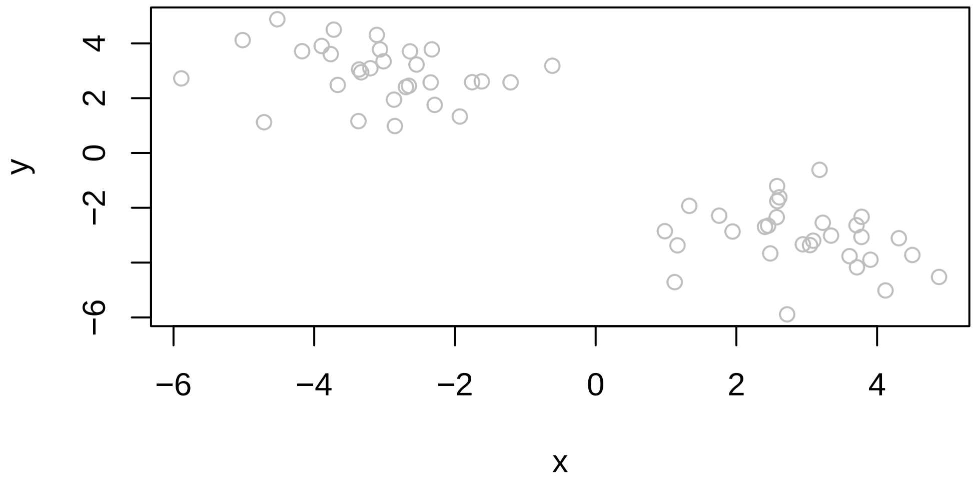
```r
centers <- data.frame(km$centers)
ggplot(data = NULL) +
    geom_point(data = x, aes(x = x, y = y, color = km$cluster)) +
    geom_point(data = centers, aes(x = x, y = y), size = 4, alpha = 0.4) +
    labs(title = "Points clustered with centers labeled")
```



Making repeating vectors

```
mycols <- rep("grey", 60)
```

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



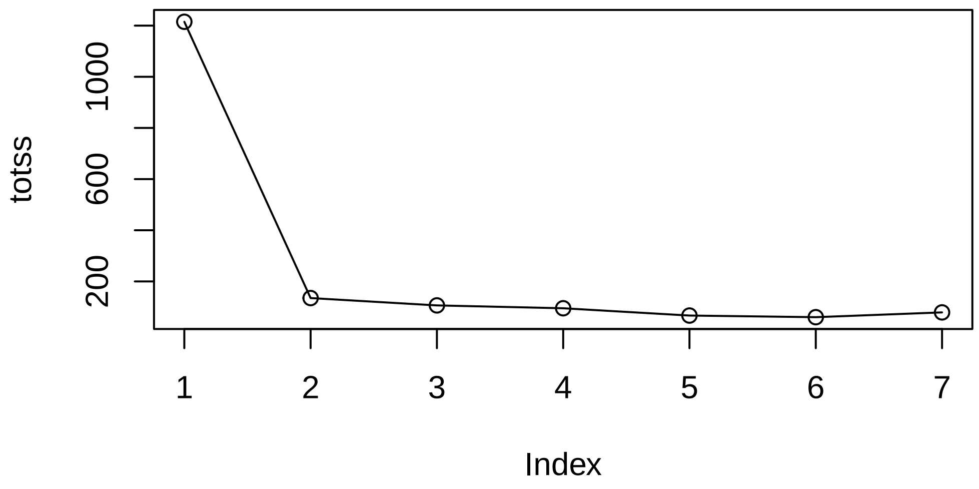
for loop in R

Try out various numbers for k, 1-7. We will write a for-loop to do this for us and store relevant output.

```
totss <- NULL      # define empty output first
k <- 1:7           # set k values to test
for (i in k) {
  # cat(i, "\n")     # print the current k value
  totss <- c(totss, kmeans(x, centers = i)$tot.withinss)
        # above runs kmeans, only saves tot.withinss values to the totss empty
}
```

Scree Plot

```
plot(totss, typ = "o")
```



From Scree plot can see proper number of clusters is 2.

Hierarchical Clustering

Starts with every point being in its own cluster. However we can't just give the function `hclust()` our input data `x` like we did for `kmeans()`. We need to first calculate a **distance matrix**. Can calculate this with the `dist()` function, by default will calculate Euclidean distance.

Calculate distance matrix:

```
d <- dist(x)
head(d)
```

```
[1] 3.0149733 3.1242843 0.7896495 0.6991370 1.8581613 2.6972521
```

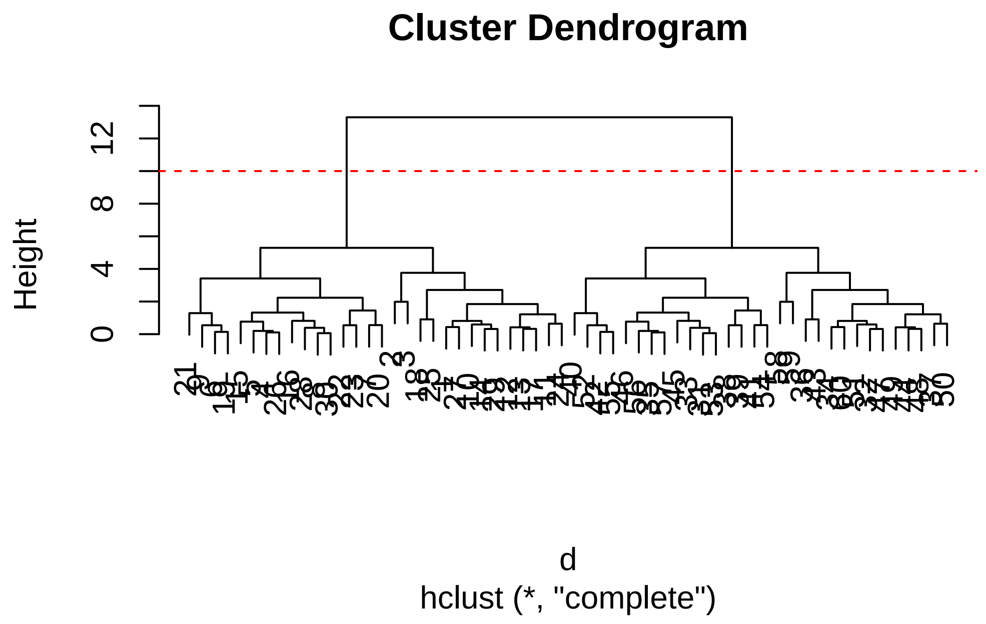
Use distance matrix for hierarchical clustering:

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

```
Call:
hclust(d = d)

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

The print out isn't helpful but the plot method is useful. It makes a **dendrogram**.

```
plot(hc) +
abline(h = 10, col = "red", lty = 2)
```



```
integer(0)
```

```
# annotated the place where clustering will be done, resulting in 2 clusters
```

The numbers in the dendrogram are the rownames (helpful if your input data is genes in the future).

The height coordinate on the graph corresponds to the Euclidean distance between the set of points. The biggest "goalposts" or vertical lines is the place where there is the biggest distance between 2 groups, likely indicates where a cluster should be.

Don't look at how close horizontally two numbers/rows are, that doesn't necessarily mean anything! check the height of the bars that is the indicator of distance. If you want to double check look at the distance matrix.

To actually cluster the data by the red line in the graph above out of a hclust object, I can use the `cutree()`. Cutree returns a vector with the points annotated into clusters (the branches of the tree) resulting from a cut of that height.

```
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 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
```

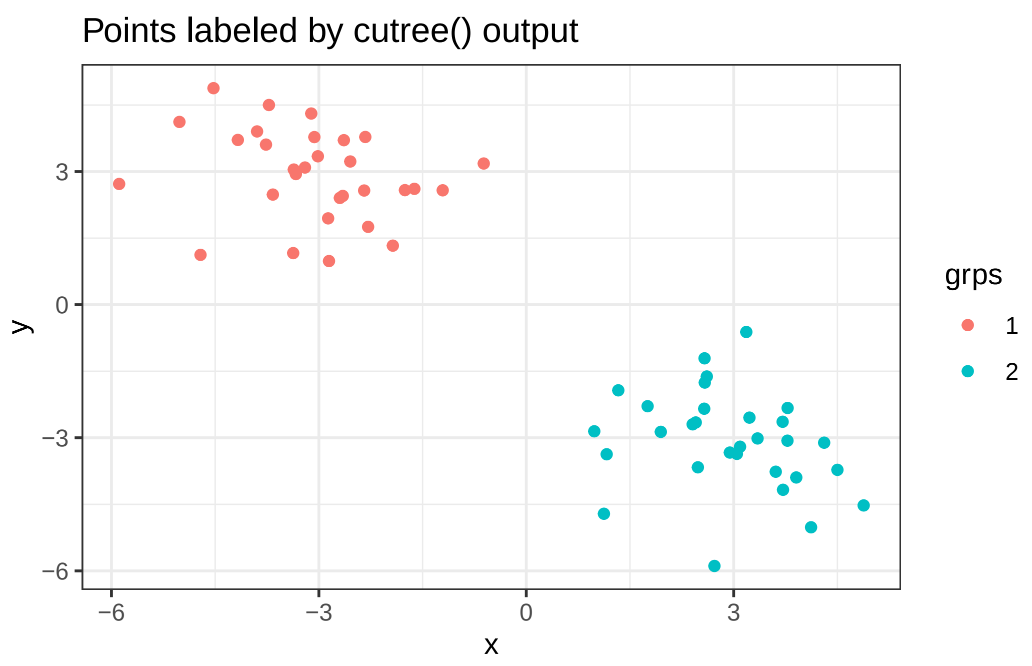
```
# can cut by h (height of red line) or k
```

You can also cutree by k (easier if there are tons of points)

```
grps <- cutree(hc, k = 2)
```

Figure for cutree output

```
grps <- as.factor(grps)
ggplot(data = x) +
  aes(x = x, y = y, color = grps) +
  geom_point() +
  labs(title = "Points labeled by cutree() output")
```



Principal Component Analysis

Dimensional Reduction:

On a PCA plot, the first PC (PC1) follows the "best fit" through the data. Principal components are new low dimensional axes closest to the observations. Once we find the line of best fit and the next line describing variation, we plot along these lines (not original dimensions).

The function is `prcomp()`.

UK food data

Importing data

```
ukfood <- read.csv("https://bioboot.github.io/bggn213_f17/class-material/UK_foods.csv", he
# made sure to set row names to be the food categories not numbers
```

Get the number of dimensions:

```
dim(ukfood)
```

```
[1] 17  4
```

```
View(ukfood)
```

Preview first 6 lines

```
head(ukfood)
```

```
               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
```

```
ukfood_tidy <- ukfood %>% pivot_longer(c(England, Wales, Scotland, N.Ireland), names_to =

#ggplot(data = ukfood_tidy) +
#  aes(x = rownames(ukfood_tidy), y = counts) +
#  geom_bar(aes(group = region, color = region))
```

Running the PCA

Don't for get to transpose the matrix! PCA needs this.

```
pca <- prcomp(t(ukfood))
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 second row **Proportion of Variance** tells you what proportion (%) of the variation is captured by the PC#.

See all attributes of the dataset (PCA)

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

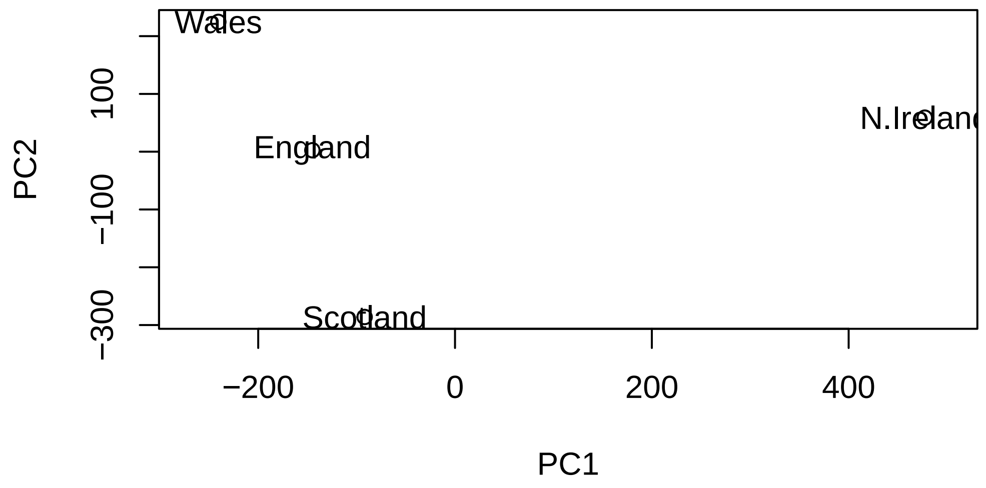
"x" is what we want for plotting

```
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
```

Plotting PCA results: also called "score plot", "PCA plot"

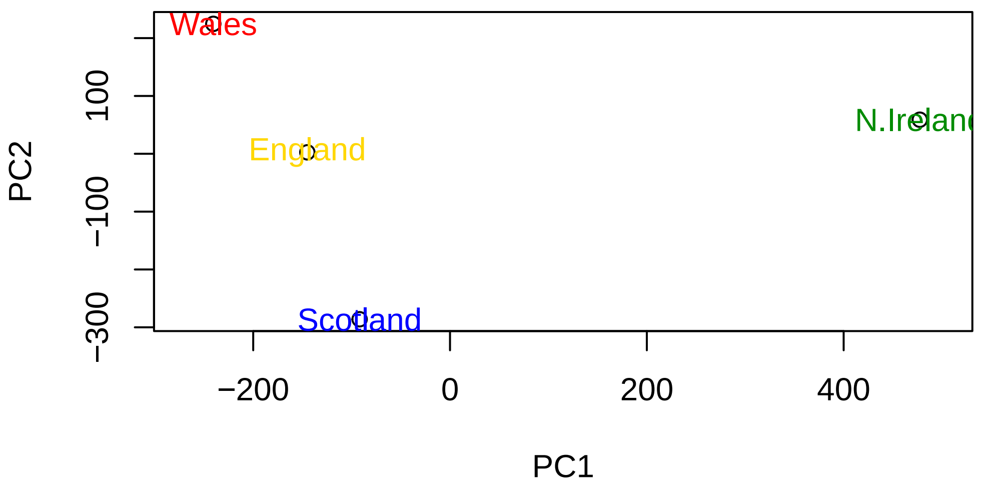
```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))+
text(pca$x[,1], pca$x[,2], colnames(ukfood))
```



```
integer(0)
```

Colored plot

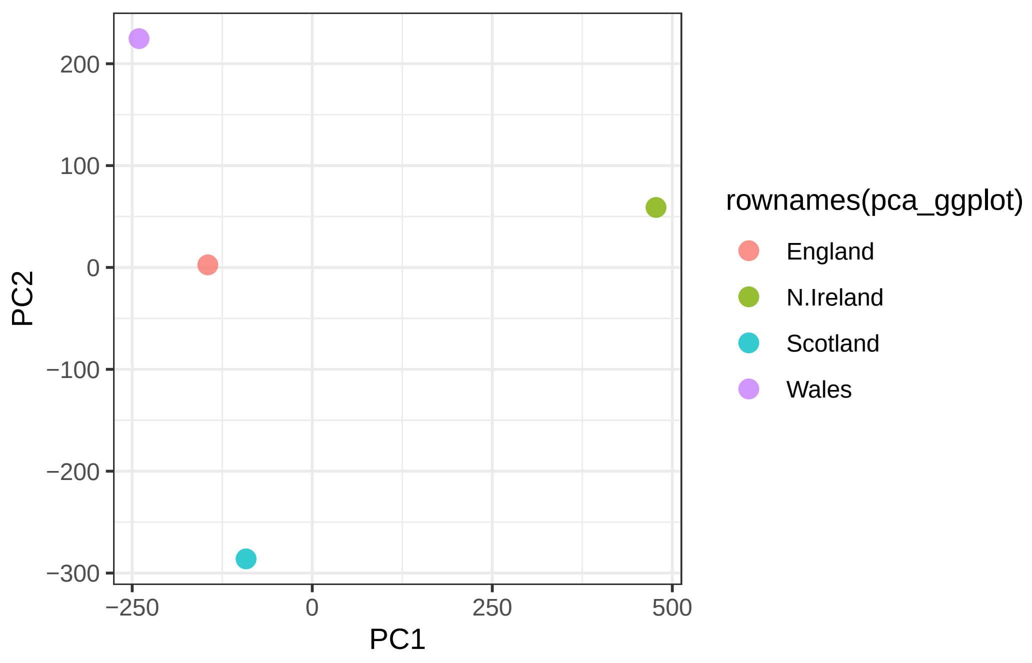
```r
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500)) +
  text(pca$x[,1], pca$x[,2], colnames(ukfood), col = c("gold", "red", "blue", "green4"))
```



```
integer(0)
```

Ggplot

```r
pca_ggplot <- data.frame(pca$x)
ggplot(data = pca_ggplot) +
  aes(x = PC1, y = PC2) +
  geom_point(aes(color = rownames(pca_ggplot)), size = 3, alpha = 0.8)
```



```
  # scale_color_manual(names = c("England", "N.Ireland", "Scotland", "Wales"), values = ro
```

Finding the variation: Below we can use the square of **pca$sdev**, which stands for "standard deviation", to calculate how much variation in the original data each PC accounts for

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

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

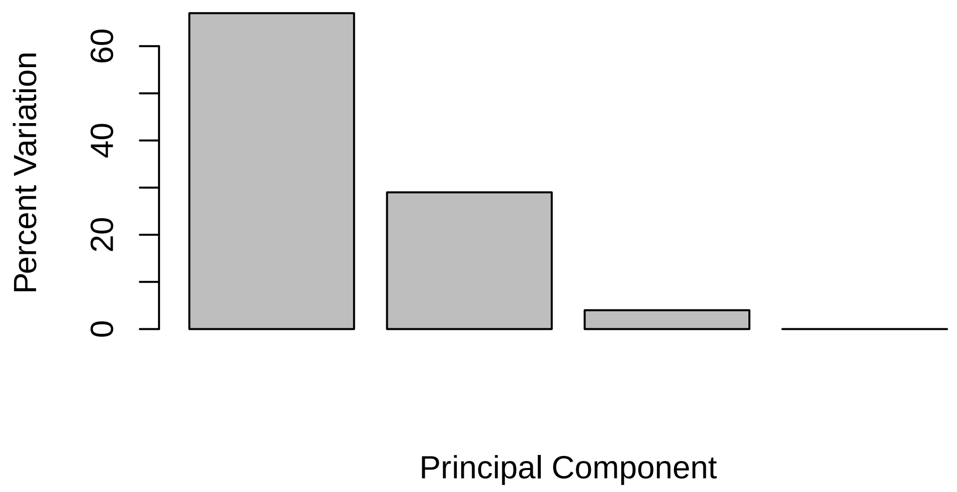
Can also see here:

```
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
```

Plot the variances (eigenvalues) with respect to the number of pc's (eigenvector number):

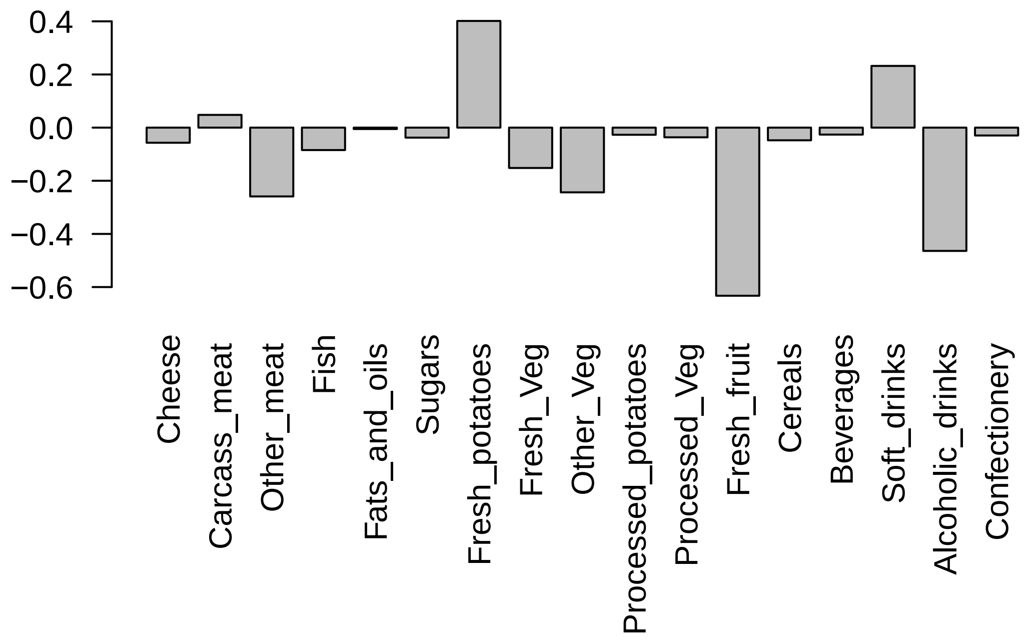
```r
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Variable Loadings

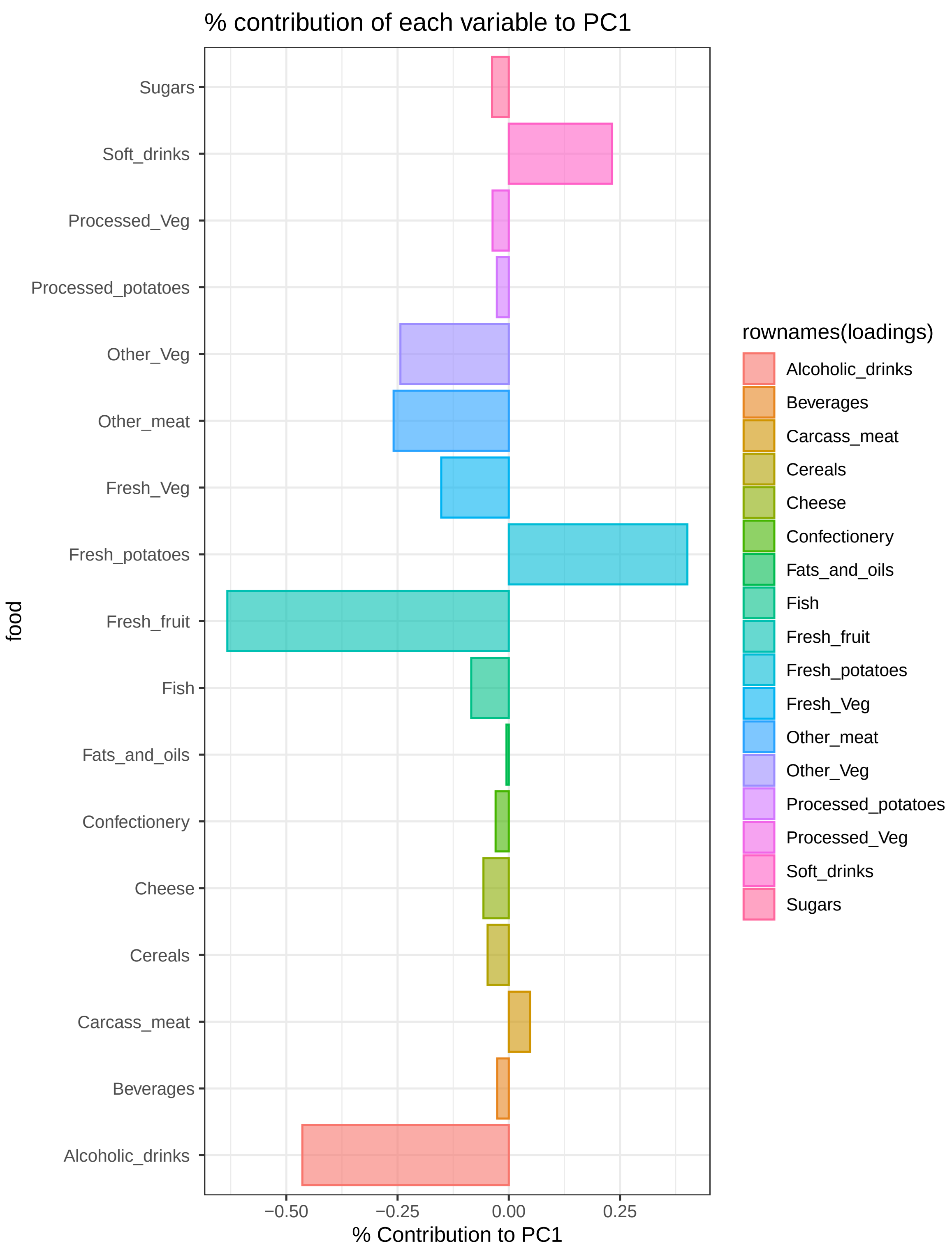
See what % each original variable contributes to the new PCs. These are stored as `pca$rotation`.

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



```
# plots the first column of pca$rotation
```

```
loadings <- as.data.frame(pca$rotation)
ggplot(data = loadings) +
  aes(x = PC1, rownames(loadings)) +
  geom_col(aes(color = rownames(loadings), fill = rownames(loadings)), alpha = 0.6) +
  labs(title = "% contribution of each variable to PC1") + ylab("food") + xlab("% Contribu
```


% contribution of each variable to PC1
Sugars
Soft_drinks
Processed_Veg
Processed_potatoes
Other_Veg
Other_meat
Fresh_Veg
Fresh_potatoes
Fresh_fruit
Fish
Fats_and_oils
Confectionery
Cheese
Cereals
Carcass_meat
Beverages
Alcoholic_drinks
food
−0.50
−0.25
0.00
0.25
% Contribution to PC1
rownames(loadings)
Alcoholic_drinks
Beverages
Carcass_meat
Cereals
Cheese
Confectionery
Fats_and_oils
Fish
Fresh_fruit
Fresh_potatoes
Fresh_Veg
Other_meat
Other_Veg
Processed_potatoes
Processed_Veg
Soft_drinks
Sugars

Bars with positive value on PC1 axis, mean N.ireland has **more** of that variable. Negative values mean N.ireland has **less**.