# Unsupervised learning - clustering and dimension reduction

*Brad Kim*

*Fall 2018*

## Lab Section

Download auto data from the github page https://github.com/ayeaton/BMSC-GA-4439-Fall2018/blob/master/Auto.data.txt or the *Statistical Learning* book website here: http://www-bcf.usc.edu/~gareth/ISL/data.html

Today, we are going over Hierarchical clustering, K-Means Clustering, PCA, ICA, and NMF.

*HW- PCA, ICA, K-means clustering, NNFM on iris data*

```r
setwd("/Users/MOOSE/Desktop/ML/Assignments/Lab2")
Auto_data <- read.table("Auto.data.txt", header=T, stringsAsFactors = F)
#remove cars with unknown horsepower and set horsepower to numeric
Auto_data <- Auto_data[-which(Auto_data$horsepower == "?"),]
Auto_data$horsepower <- as.numeric(Auto_data$horsepower)
#save car names
Auto_data_names <- Auto_data$name
#use the numeric values
Auto_data_clust <- Auto_data[,1:8]
dim(Auto_data_clust)
```

```
## [1] 392   8
```

```r
#392 is too much for a demo, so lets take the first 25
Auto_data_clust <- Auto_data_clust[1:25,]
rownames(Auto_data_clust) <- Auto_data_names[1:25]
```

*I'm just gonna take all 150 rows cuz I feel like it. And I'll just use HPC if my labtop is too slow. Remove the comment for the next 2 lines after dim() if you would like to just use the first 25. Reminder that the 150 flowers are just named 1-150. The cluster will look messy.*

```r
data(iris)
iris_subs <- iris[iris$Species,c(1, 2, 3, 4)]
#Why is species saved as factors in R? Need to clean the data.
#iris = data.frame(lapply(iris$Species, as.character))

iris <- factor2character(iris)
species <- iris$Species
#save species as character, not factors
iris_clust <- iris[,1:4]
dim(iris_clust)
```

```
## [1] 150   4
```

```
#make.names handles the unique row ERROR for rownames
rownames(iris_clust) <- make.names(species, unique=TRUE)


# I'm just gonna take all 150 rows cuz I feel like it. And I'll just use HPC if my labtop is t

#iris_clust <- iris_clust[1:25,]
#rownames(iris_clust) <- species[1:25]

#392 is too much for a demo, so lets take the first 25
#Auto_data_clust <- Auto_data_clust[1:25,]
#rownames(Auto_data_clust) <- Auto_data_names[1:25]
```

### Hierarchical agglomerative clustering

Step 1. Assign each item to it's own cluster. We start with 25 clusters, one for each car.

Step 2. Calculate a similarity matrix between each cluster.

Step 3. Find the pair of clusters closest in similarity.

Step 4. Merge these clusters/recalculate similarity between clusters. Options are: single linkage (nearest neighbor), complete linkage (furthest neighbor), average linkage (mean distance between all pairs of data from the two different clusters), centroid linkage (distance between the means of all points in the clusters). Now we have 24 clusters.

Step 5. Repeat Step 3 and 4 until there is only one cluster.

### In practice

Step 1. Each car is a cluster.

Step 2. Create a distance matrix from Auto_data_clust.

```
help("dist")
iris_hierarchical_dist <- as.matrix(dist(iris_clust, method = "euclidean"))
View(iris_hierarchical_dist)
```

Step 3. Find the two cars that are the most similar to each other and print the names of those two cars

```
diag(iris_hierarchical_dist) <- NA
arrayInd(which.min(iris_hierarchical_dist), dim(iris_hierarchical_dist))
```

```
##      [,1] [,2]
## [1,]  143  102
```

```
#positions 143 and 102 are the most similar. Lets go back to the names of the cars
species[143]
```

```
## [1] "virginica"
```

```
species[102]
```

```
## [1] "virginica"
```

Step 4. Merge the two clusters together using average linkage.

```
#replace pos 102 with the average of pos 143 and 102
iris_hierarchical_dist[,102] <- apply((iris_hierarchical_dist[,c(143,102)]),1,mean)
iris_hierarchical_dist[102,] <- apply((iris_hierarchical_dist[c(143,102),]),2,mean)

#remove pos 143
iris_hierarchical_dist <- iris_hierarchical_dist[-143,-143]

#now position 102 represents the cluster containing the vrginicas
```

Step 5. To complete the algorithm, go back to step 3 and iterate through all of the previous steps until there are no more rows left

```
diag(iris_hierarchical_dist) <- NA
arrayInd(which.min(iris_hierarchical_dist), dim(iris_hierarchical_dist))
```

```
##      [,1] [,2]
## [1,]   40    8
```

```
#postitions 40 and 8 are the most similar
species[40]
```

```
## [1] "setosa"
```

```
species[8]
```

```
## [1] "setosa"
```

```
#Step 3. Find the two cars that are the most similar to each other and print the names of thos
diag(iris_hierarchical_dist) <- NA
arrayInd(which.min(iris_hierarchical_dist), dim(iris_hierarchical_dist))
```

```
##      [,1] [,2]
## [1,]   40    8
```

```
#replace pos 8 with the average of pos 40 and 8
iris_hierarchical_dist[,8] <- apply((iris_hierarchical_dist[,c(40,8)]),1,mean)
iris_hierarchical_dist[8,] <- apply((iris_hierarchical_dist[c(40,8),]),2,mean)

#remove pos 40
iris_hierarchical_dist <- iris_hierarchical_dist[-40,-40]

#now position 8 represents the cluster containing setosa
```

REPEAT. Don't really need this.

```r
diag(iris_hierarchical_dist) <- NA
arrayInd(which.min(iris_hierarchical_dist), dim(iris_hierarchical_dist))
```

```
##      [,1] [,2]
## [1,]   18    1
```

```r
##postitions 6 and 4 are the most similar. Lets go back to the names of the cars
#Auto_data_names[6]
#Auto_data_names[4]
#
##postitions 6 and 4 are the most similar
#
##replace pos 4 with the average of pos 6 and 4
#hierarchical_dist[,4] <- apply((hierarchical_dist[,c(6,4)]),1,mean)
#hierarchical_dist[4,] <- apply((hierarchical_dist[c(6,4),]),2,mean)
#
##remove pos 4
#hierarchical_dist <- hierarchical_dist[-6,-6]
#
##now position 6 represents the cluster containing the ford galaxie 500 and amc rebel sst
```

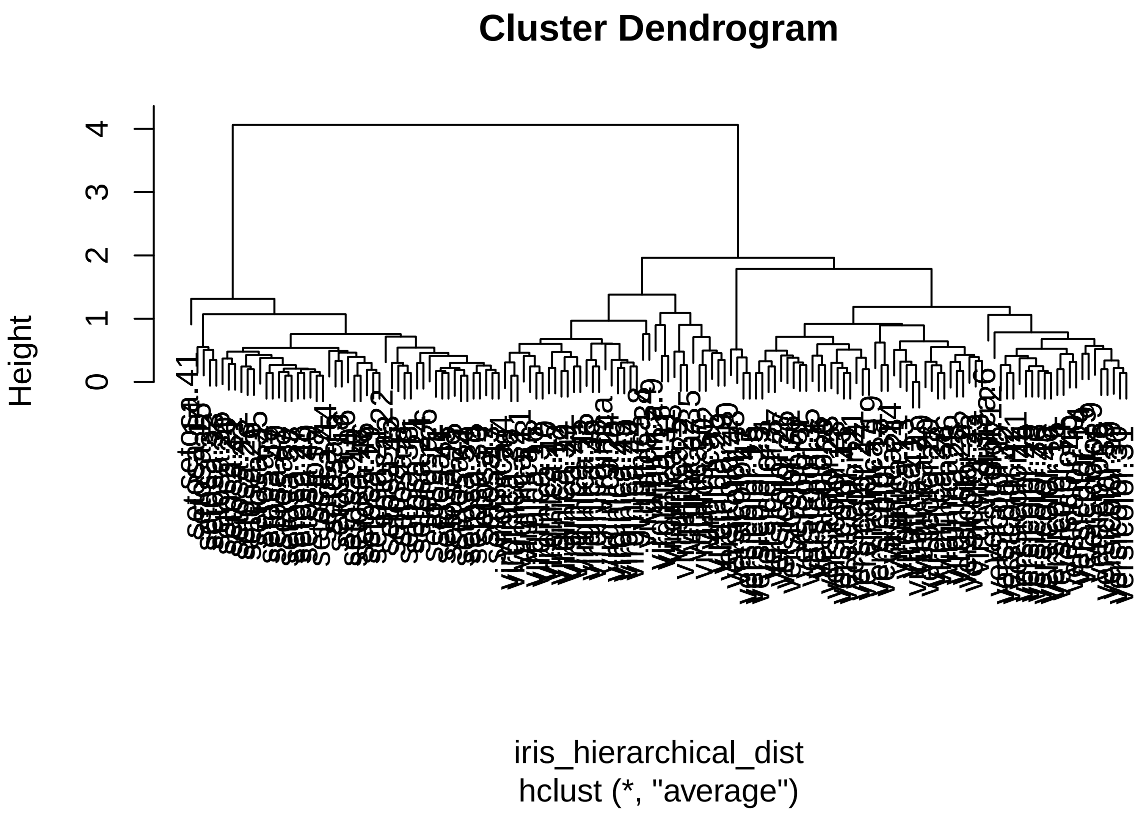
```r
#diag(hierarchical_dist) <- NA
#arrayInd(which.min(hierarchical_dist), dim(hierarchical_dist))

##postitions 5 and 4 are the most similar. Lets go back to the names of the cars
#Auto_data_names[5]
#Auto_data_names[4]
#
##postitions 5 and 4 are the most similar
#
##replace pos 4 with the average of pos 5 and 4
#hierarchical_dist[,4] <- apply((hierarchical_dist[,c(5,4)]),1,mean)
#hierarchical_dist[4,] <- apply((hierarchical_dist[c(5,4),]),2,mean)
#
##remove pos 4
#hierarchical_dist <- hierarchical_dist[-5,-5]
#
##now position 5 represents the cluster containing the ford torino and amc rebel sst
#
```

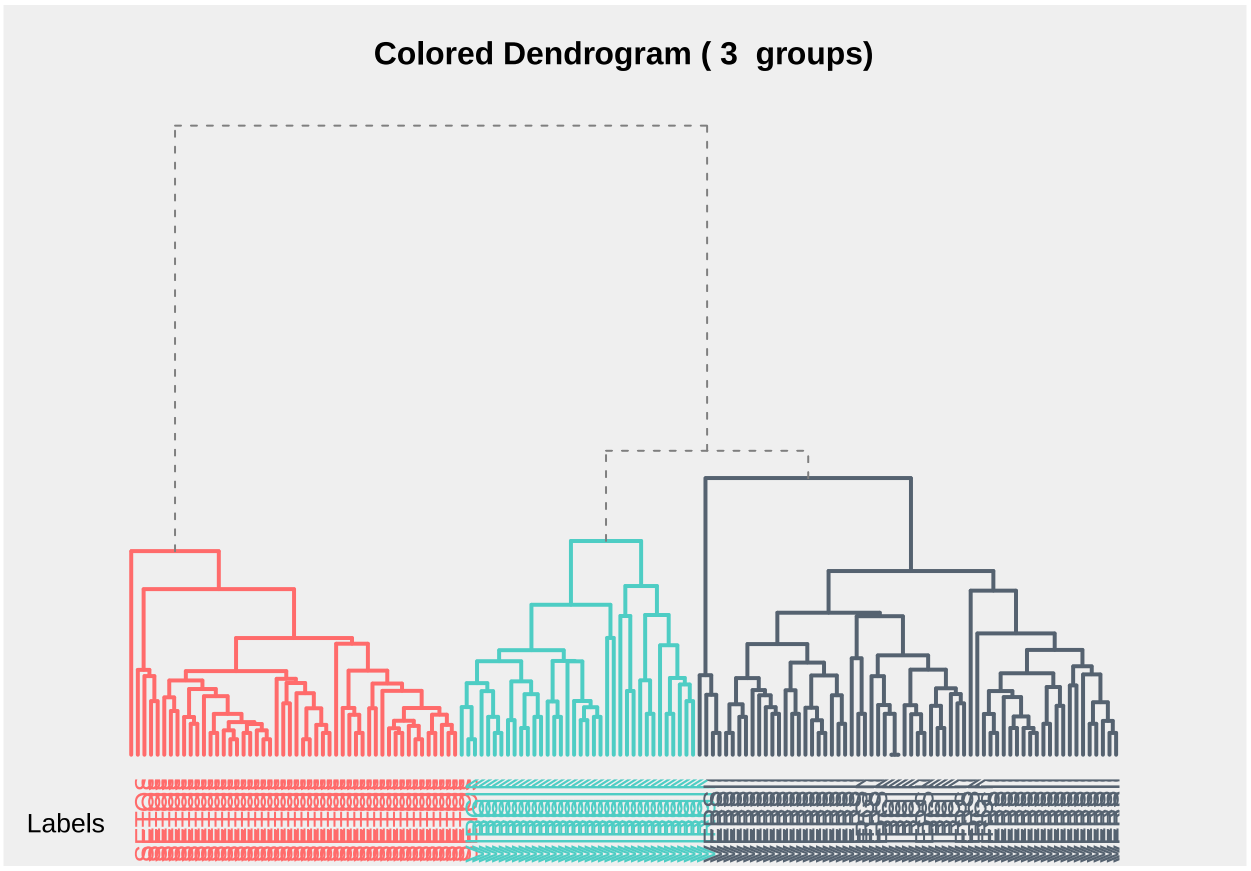
### R function

Now that we know how the algorithm works, let's use the R function hclust. Plot the Dendogram resulting from clustering the Auto_data_clust using average linkage.

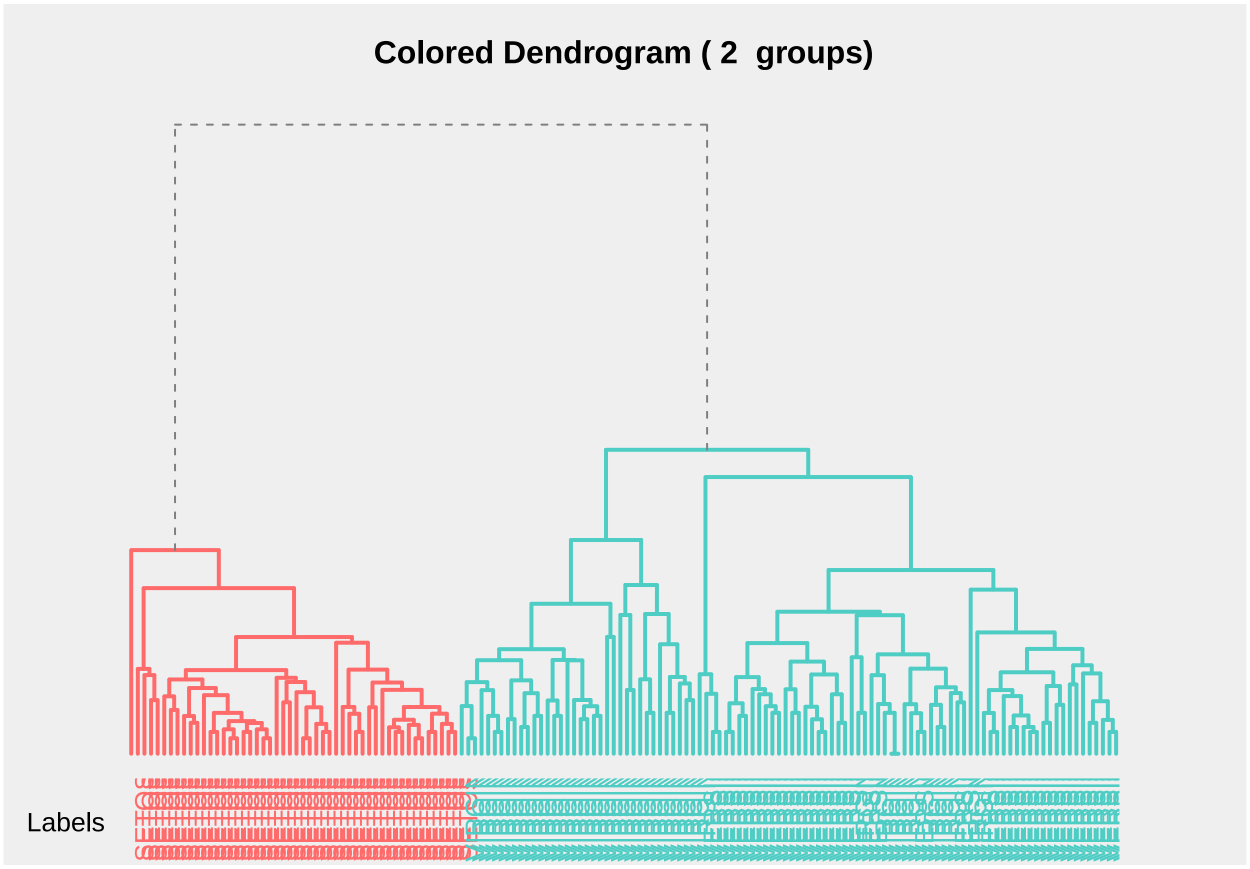
```r
iris_hierarchical_dist <- dist(iris_clust, method = "euclidean")
hc_iris <- hclust(iris_hierarchical_dist, method="average")
plot(hc_iris)
```



```
# load code of A2R function
source("http://addictedtor.free.fr/packages/A2R/lastVersion/R/code.R")
# colored dendrogram
op = par(bg = "#EFEFEF")
A2Rplot(hc_iris, k = 3, boxes = FALSE, col.up = "gray50", col.down = c("#FF6B6B",
    "#4ECDC4", "#556270"))
```



```r
A2Rplot(hc_iris, k = 2, boxes = FALSE, col.up = "gray50", col.down = c("#FF6B6B", "#4ECDC4"))
```


Colored Dendrogram ( 2  groups)
Labels

##K-Means Clustering Step 1. Choose the N number of clusters. 2 cuz the plots say so.

Step 2. Find the N items that are furthest apart and set them as cluster centroids.

Step 3. Assign one item in the dataset to the closest of the N cluster centroids.

Step 4. Recalculate the cluster centroid.

Step 5. Repeat Steps 3 and 4 until all items are in a cluster.

Step 6. Go through each item and reassess whether the item belongs in the current cluster or in a different cluster based on distance to cluster centroids. Every time an item is reassigned to a different cluster, the centroids must be recalculated.

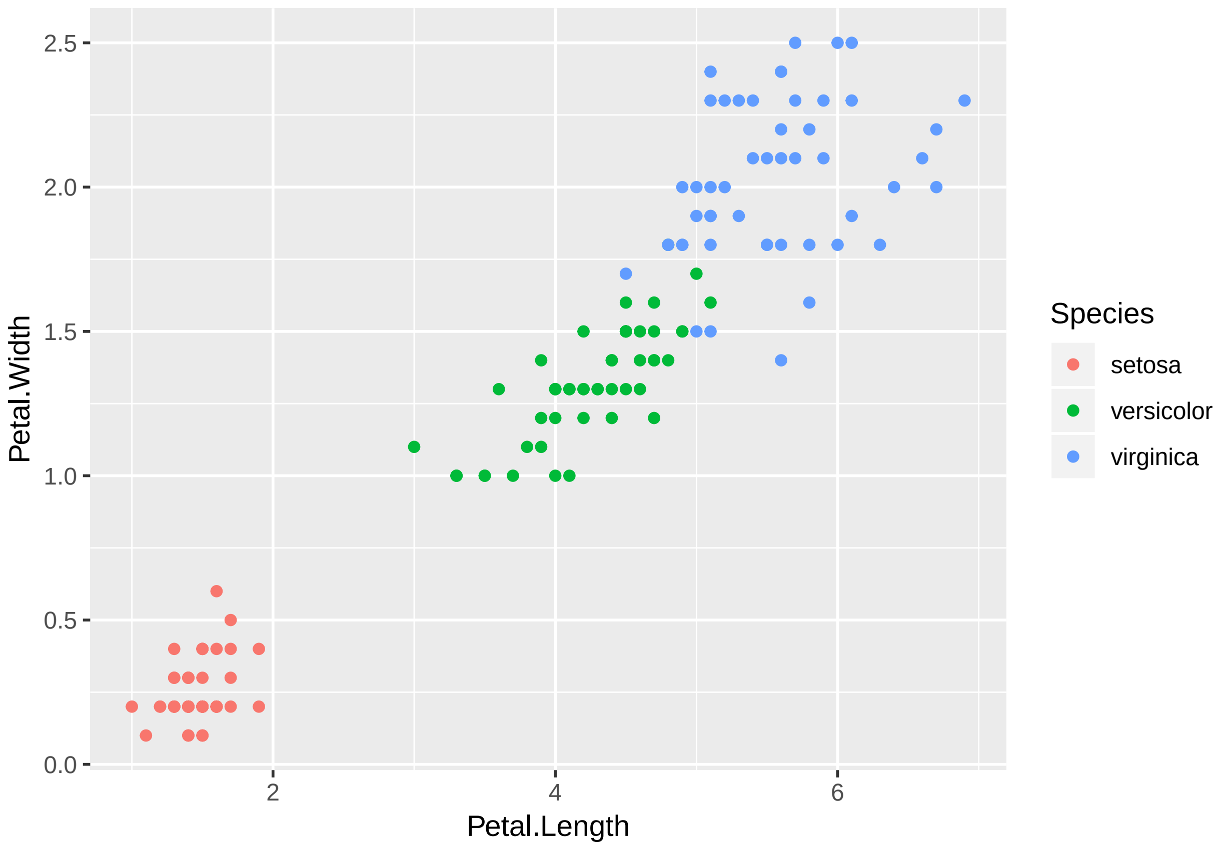
Step 7. When every item belongs firmly to a cluster, or the iterations of Step 6 are endless, the algorithm is complete.

### In practice

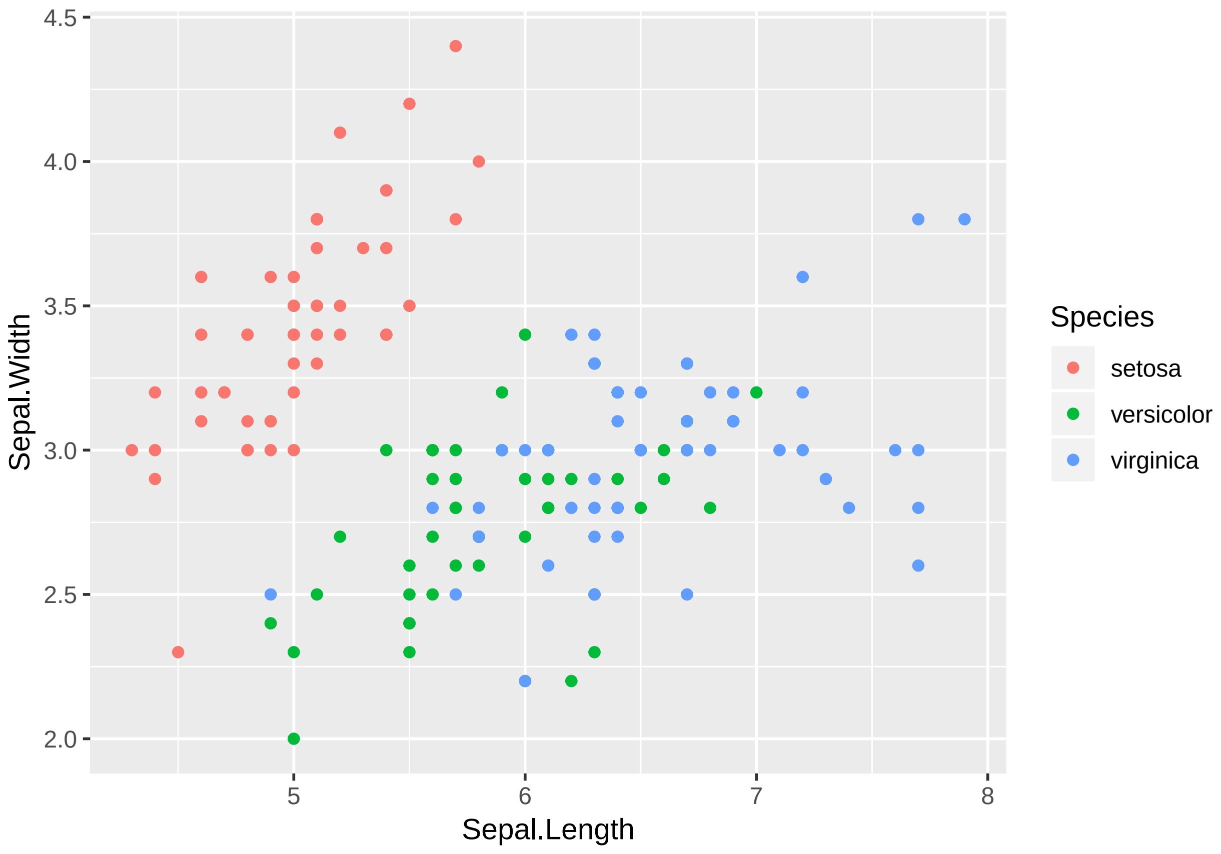
Step 1. We are going to cluster the 150 flowers into two groups.

Step 2a. Find the two flowers furthest from each other.

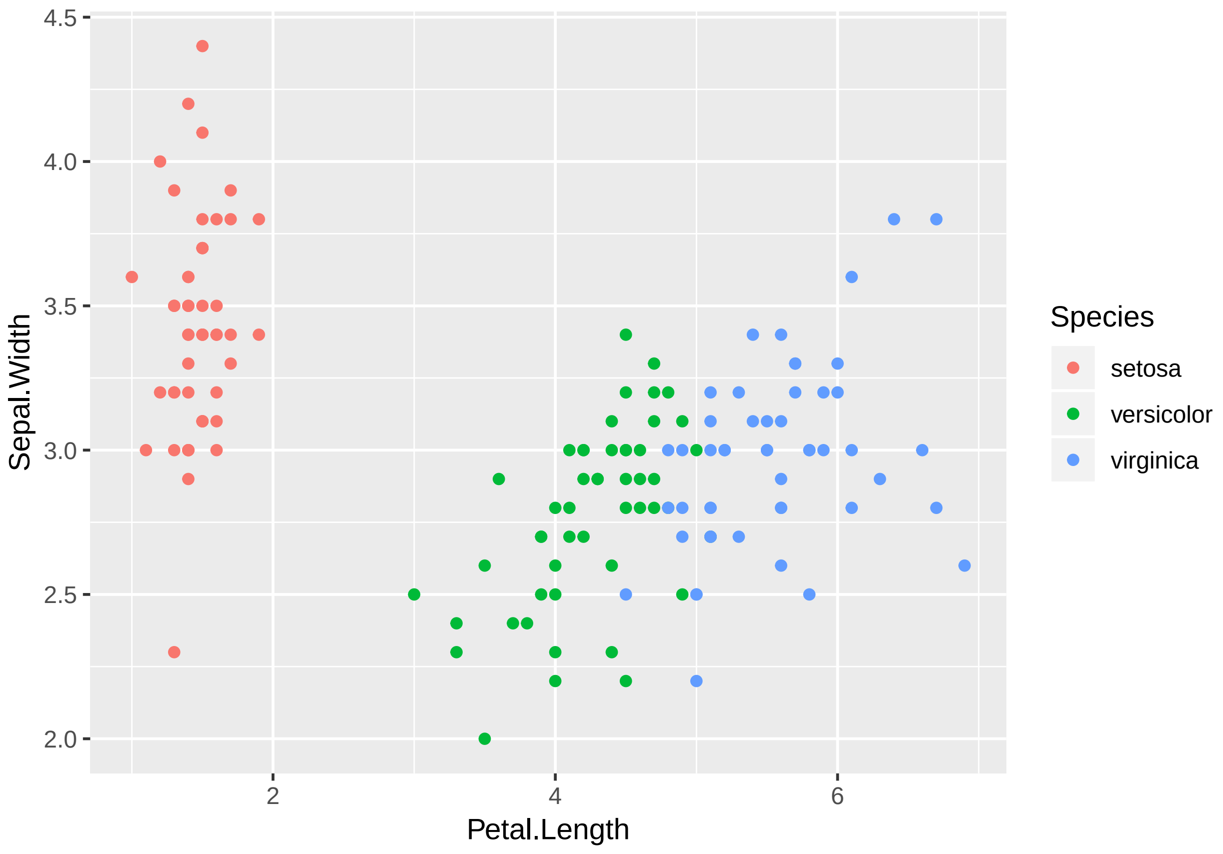
```
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
```



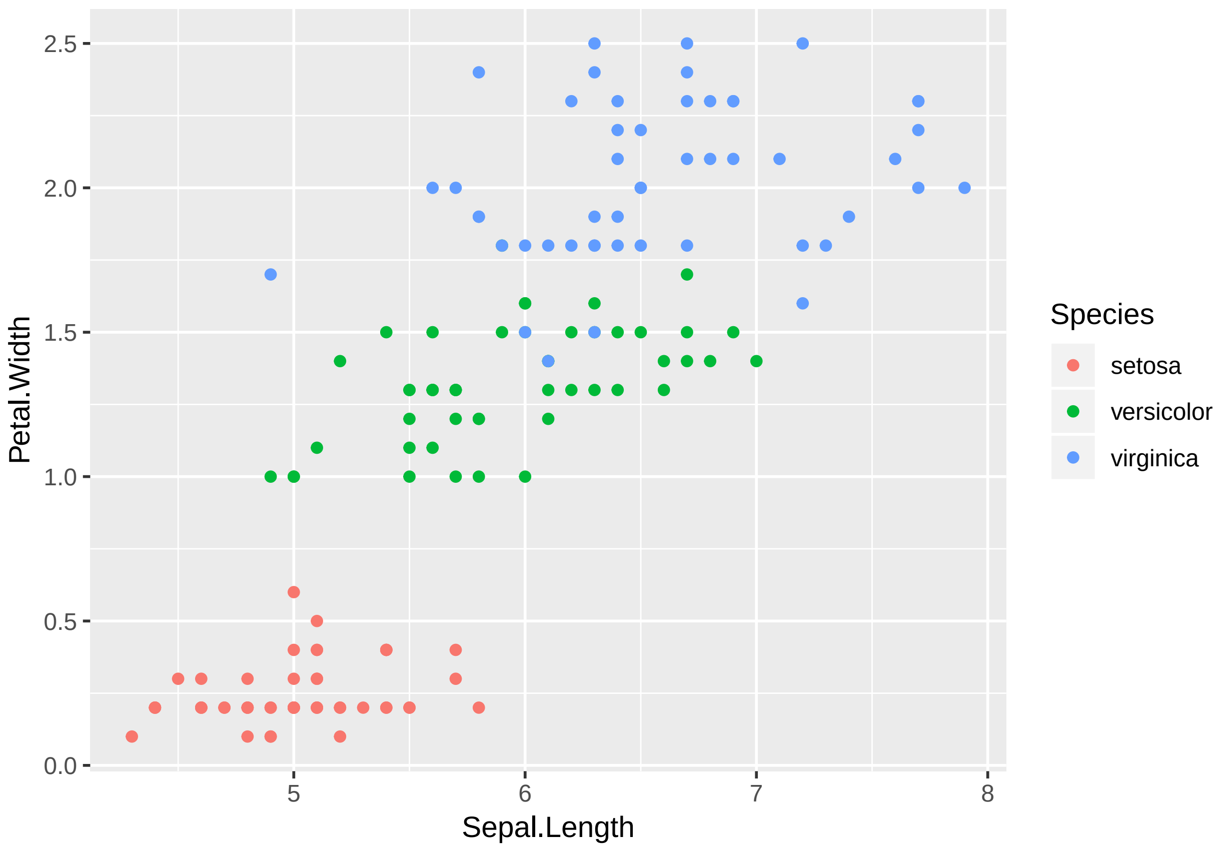
```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point()
```



```r
ggplot(iris, aes(Petal.Length, Sepal.Width, color = Species)) + geom_point()
```



```r
ggplot(iris, aes(Sepal.Length, Petal.Width, color = Species)) + geom_point()
```



```r
iris_kmeans_dist <- as.matrix(dist(iris_clust, method = "euclidean"))
diag(iris_kmeans_dist) <- NA
arrayInd(which.max(iris_kmeans_dist), dim(iris_kmeans_dist))
```

```
##      [,1] [,2]
## [1,]  119   14
```

Step 2b. Create data frames to hold each cluster, cluster names, and centroids.

```r
iris_cluster_one <- iris_clust[119,]
iris_cluster_one_names <- species[119]
iris_cluster_one_centroid <- iris_cluster_one

iris_cluster_two <- iris_clust[14,]
iris_cluster_two_names <- species[119]
iris_cluster_two_centroid <- iris_cluster_two
```

Step 3a. Sequentially put cars in either cluster one or cluster two. lets start with car 1. Is car 1 closer to cluster one or cluster two?

```r
#distance to cluster 1
dist(rbind(iris_cluster_one_centroid,iris_clust[1,]), method = "euclidean")
```

```
##        virginica.18
## setosa     6.498461
```

```
#distance to cluster 2
dist(rbind(iris_cluster_two_centroid,iris_clust[1,]), method = "euclidean")
```

```
##        setosa.13
## setosa 0.9949874
```

Step 3b. Add car 1 to cluster two and adjust the centroid value for cluster 2. The new centroid value is a mean of the values of cars in that cluster.

```
iris_cluster_two <- rbind(iris_cluster_two, iris_clust[1,])
iris_cluster_two_names <- rbind(iris_cluster_two_names, species[1])
iris_cluster_two_centroid <- apply(iris_cluster_two,2,function(x) mean(as.numeric(x)))
```

Step 3d. Do for all flowers

```
for(i in 1:nrow(iris_clust)){
  if(i == 102 | i == 143){
    next
  }
  if(dist(rbind(iris_cluster_two_centroid,iris_clust[i,]), method = "euclidean") < dist(rbind(
    iris_cluster_two <- rbind(iris_cluster_two, iris_clust[i,])
    iris_cluster_two_names <- rbind(iris_cluster_two_names, species[i])
    iris_cluster_two_centroid <- apply(iris_cluster_two,2,function(x) mean(as.numeric(x)))
  }else{
    iris_cluster_one <- rbind(iris_cluster_one, iris_clust[i,])
    iris_cluster_one_names <- rbind(iris_cluster_one_names, species[i])
    iris_cluster_one_centroid <- apply(iris_cluster_one,2,function(x) mean(as.numeric(x)))
  }
}
```

Step 4. Adjust the clusters by comparing the distance of each car to the centroid of its current cluster versus the distance to the centroid of the other cluster. Does it still belong in the current cluster? lets start with car 1. Does it belong in cluster two?

```
dist(dist(rbind(iris_cluster_one_centroid,iris_clust[1,])))
```

```
##               1
## setosa 5.754075
```

```
dist(dist(rbind(iris_cluster_two_centroid,iris_clust[1,])))
```

```
##                1
## setosa 0.2540102
```

Alright, so car 1 does belong in cluster two. To complete the algorithm, iterate over the cars until none of the cars switch clusters.

### R function

Now we know how the algorithm works, lets use the R function kmeans.

```r
help(kmeans)
iris_kmeans_put <- kmeans(iris_clust,2)
str(iris_kmeans_put)
```

```
## List of 9
##  $ cluster     : Named int [1:150] 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:150] "setosa" "setosa.1" "setosa.2" "setosa.3" ...
##  $ centers     : num [1:2, 1:4] 5.01 6.3 3.37 2.89 1.56 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
##  $ totss       : num 681
##  $ withinss    : num [1:2] 28.6 123.8
##  $ tot.withinss: num 152
##  $ betweenss   : num 529
##  $ size        : int [1:2] 53 97
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

## Principal Components Analysis (PCA)

Principal Components Analysis is a linear dimensionality reduction algorithm. If you want to learn more about linear algebra, I suggest the MIT Open Courseware class here : https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/ There are two ways of doing PCA, Single Value Decomposition (SVD), and the method we will use today, using the covariance matrix of the data.

Step 1. Center data by subtracting the mean.

Step 2. Calculate covariance matrix of data.

Step 3. Perform Eigendecomposition of the covariance matrix. i.e. represent the matrix in terms of it's eigenvalues and eigen vectors

Step 4. Multiply the eigen vectors by the original data to express the data in terms of the eigen vectors.

Step 1. Center the data by subtracting the mean of the each column from the values in that column

```r
#we need a square matrix
iris_clust_pca <- data.matrix(iris_clust[1:4,])

Center_iris <- apply(iris_clust_pca, 1, function(x) x - mean(x))
```

Step 2. Calculate covariance matrix of the Auto data

```r
Covariance_iris <- cov(Center_iris)
Covariance_iris
```

```
##            setosa setosa.1 setosa.2 setosa.3
## setosa   4.750000 4.421667 4.353333 4.160000
## setosa.1 4.421667 4.149167 4.055000 3.885000
## setosa.2 4.353333 4.055000 3.990000 3.813333
## setosa.3 4.160000 3.885000 3.813333 3.656667
```

Step 3. Calculate eigen values and vectors

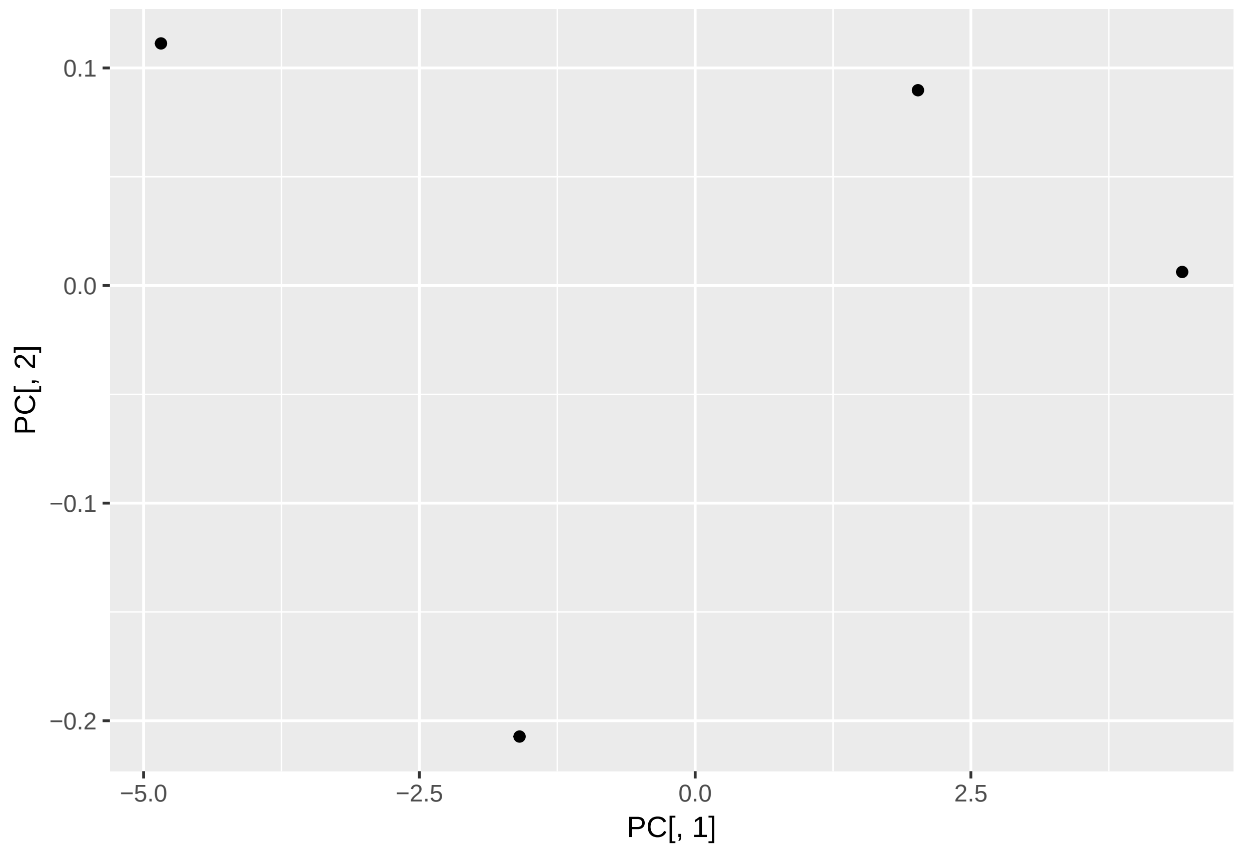
```r
Eigen_value_iris <- eigen(Covariance_iris)$value

#columns are the eigen vectors
Eigen_vector_iris <- eigen(Covariance_iris)$vector
```

Step 4. Multiply the eigen vector matrix by the original data.

```r
PC <- as.data.frame(data.matrix(Center_iris) %*% Eigen_vector_iris)

ggplot(PC, aes(PC[,1], PC[,2])) + geom_point(aes(PC[,1], PC[,2]))
```



```
#+ geom_text(aes(label=Auto_data_names[1:8]), nudge_x = -2.5, nudge_y = 400)
```

Step 5. Find out which principal components explain the variance in the data.

```
#for each component, take the cumulative sum of eigen values up to that point and and divide b
round(cumsum(Eigen_value_iris)/sum(Eigen_value_iris) * 100, digits = 2)
```

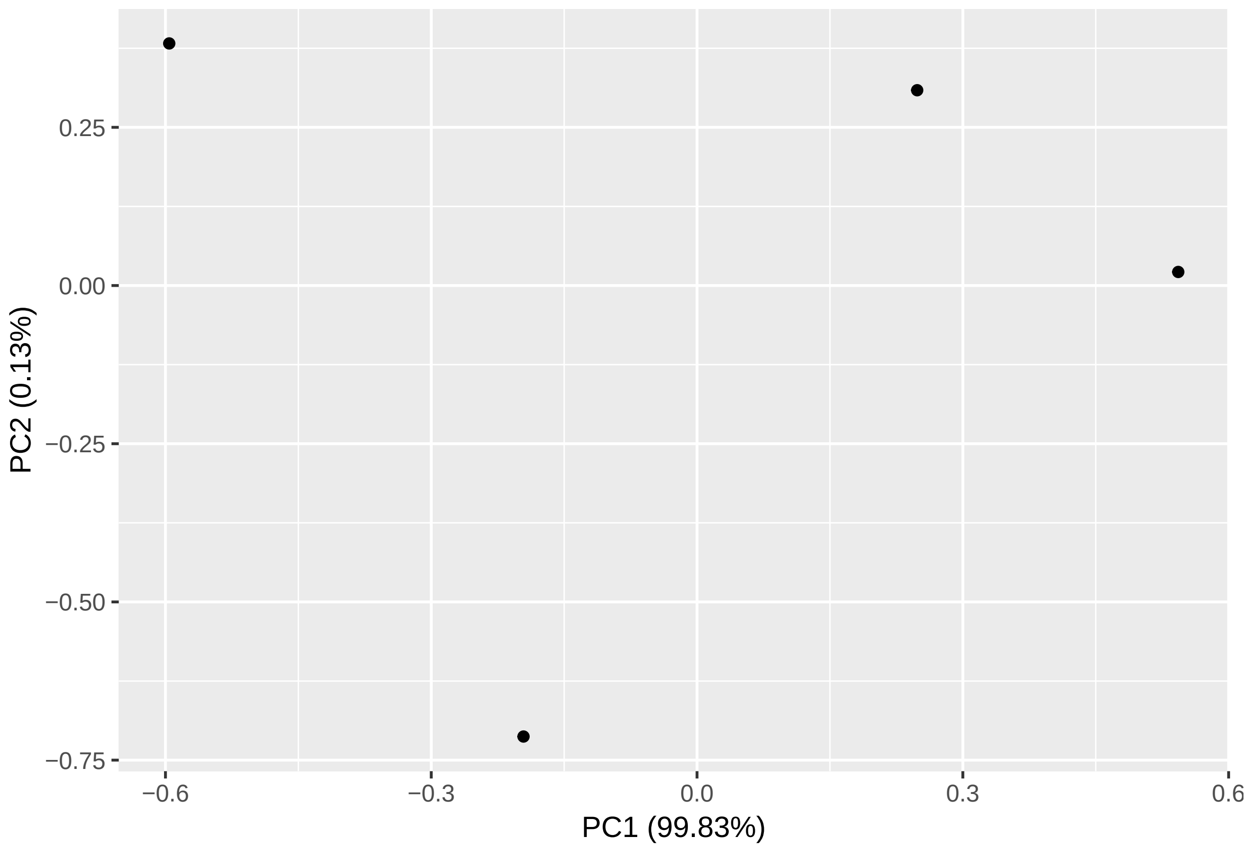
```
## [1]  99.83  99.96 100.00 100.00
```

Principal component 1 and 2 explain 99.99 percent of the variance. Principal component 1,2, and 3 together explain 100% of the variance in the data.

### R function

Now that we know how PCA works, lets use the R funtion prcomp.

```
help("prcomp")
autoplot(prcomp(t(iris_clust_pca)))
```



*Honestly, I'm not convinced that this PCA plot is correct..., but maybe some feedback would help. I'm guessing that because 99.83% and 0.13% add up to 99.96%, component 3 is still explains 0.04%?*

## Independent Component Analysis (ICA)

ICA is an algorithm that finds components that are independent, or subcomponents of the data.

Step 1. Whiten the data by projecting the data onto the eigen vectors (PCA).

Step 2. Solve the X=AS equation by maximizing non-gaussianty in the variables(components) in S.

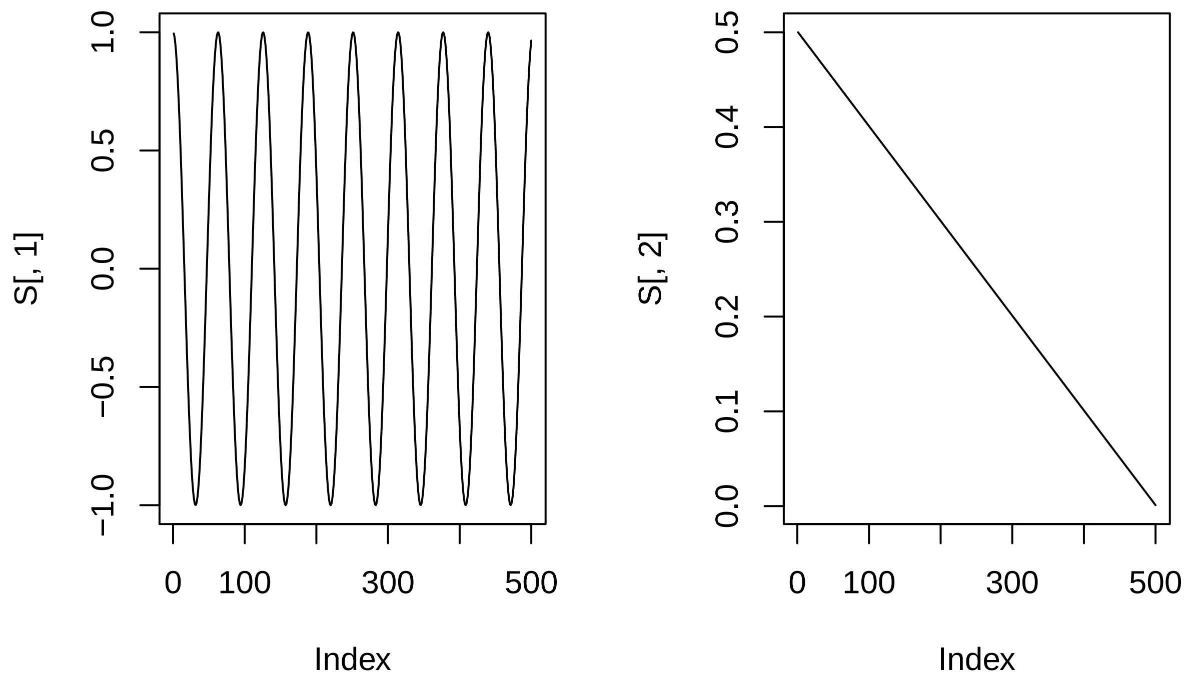
This results in a matrix S with components that are independent from each other.

We will use the fastICA algorithm.

First we will go backwards. Create a matrix S with the independent components

```r
#create two signals
S <- cbind(cos((1:500)/10), ((500:1)/1000))

par(mfcol = c(1, 2))
plot(S[,1], type="l")
plot(S[,2], type="l")
```

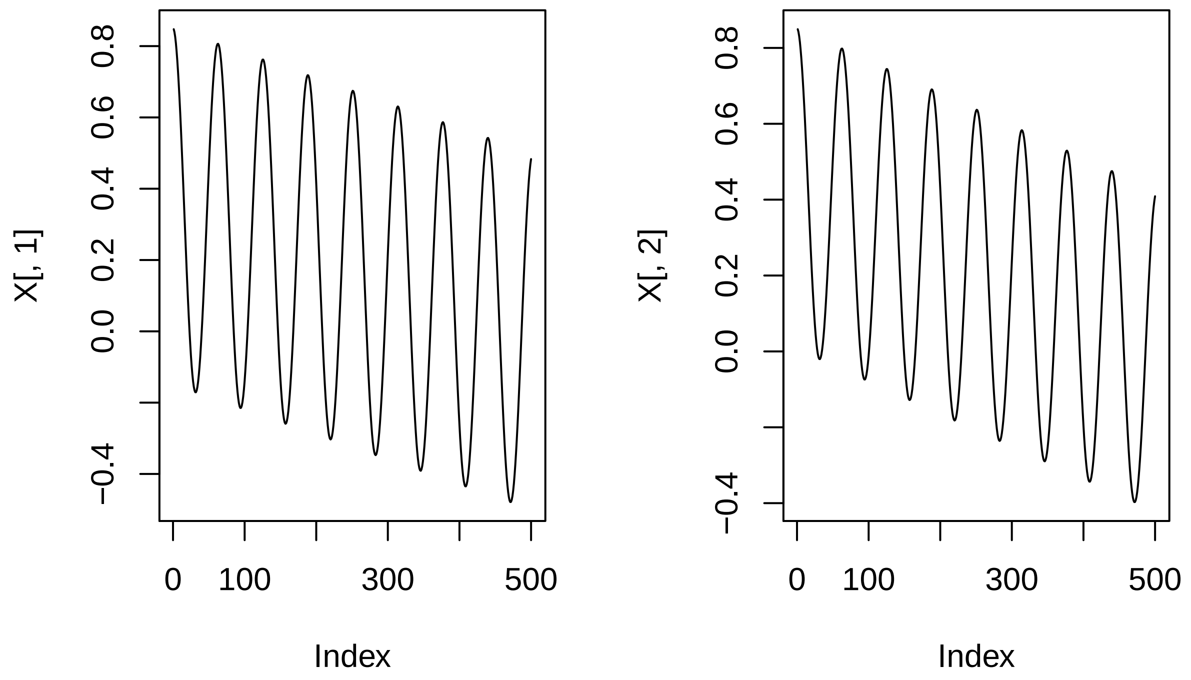


Create a mixing matrix A

```r
A <- matrix(c(0.5, 0.7, 0.423, 0.857), 2, 2)
```

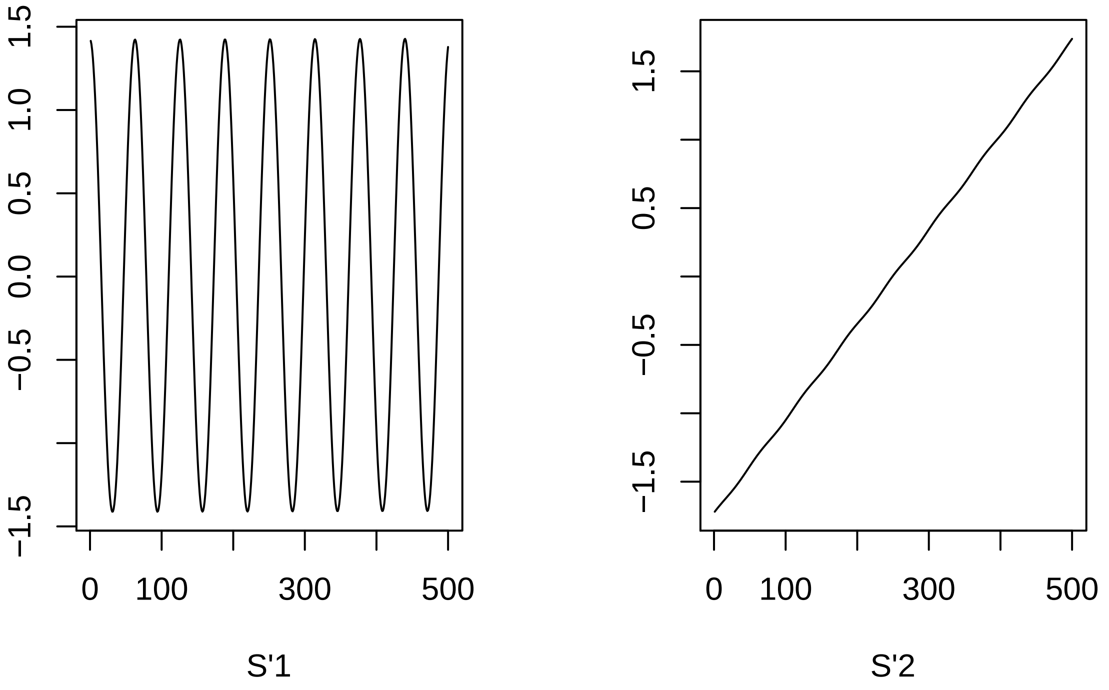
Mix S using A

```r
X <- S %*% A
par(mfcol = c(1, 2))
plot(X[,1], type="l")
plot(X[,2], type="l")
```



Unmix using fastICA

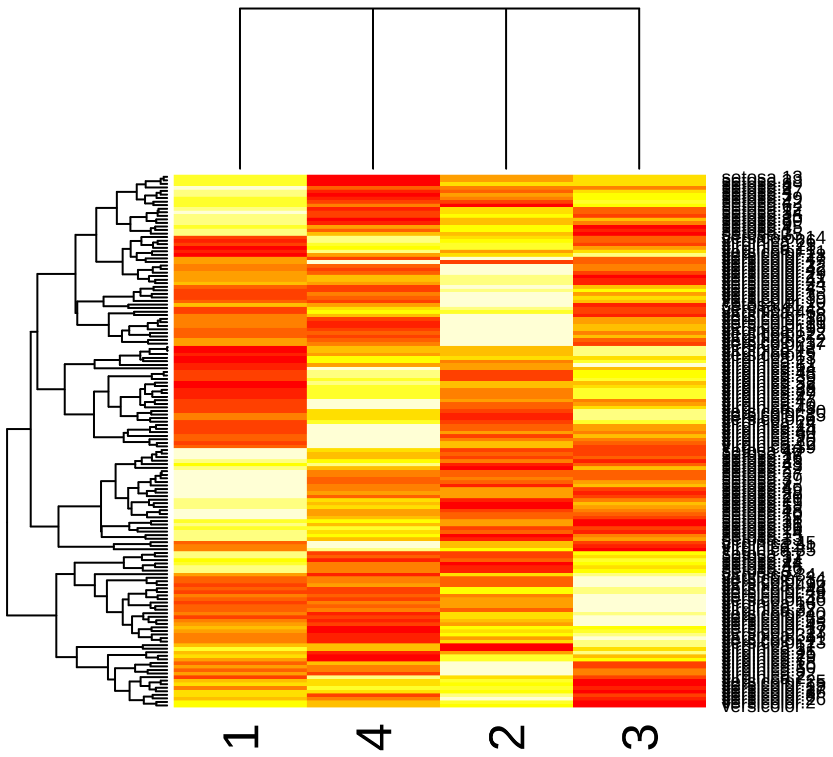
```
par(mfcol = c(1, 2))
plot(1:500, a$S[,1], type = "l", xlab = "S'1", ylab = "")
plot(1:500, a$S[,2], type = "l", xlab = "S'2", ylab = "")
```



**ICA on the iris**

plot the independent components as a heatmap

```r
heatmap(a$S)
```



### Non-negative Matrix Factorization

NMF is an algorithm that factorizes the given matrix into two matrices. All three matrices must have no negative values.

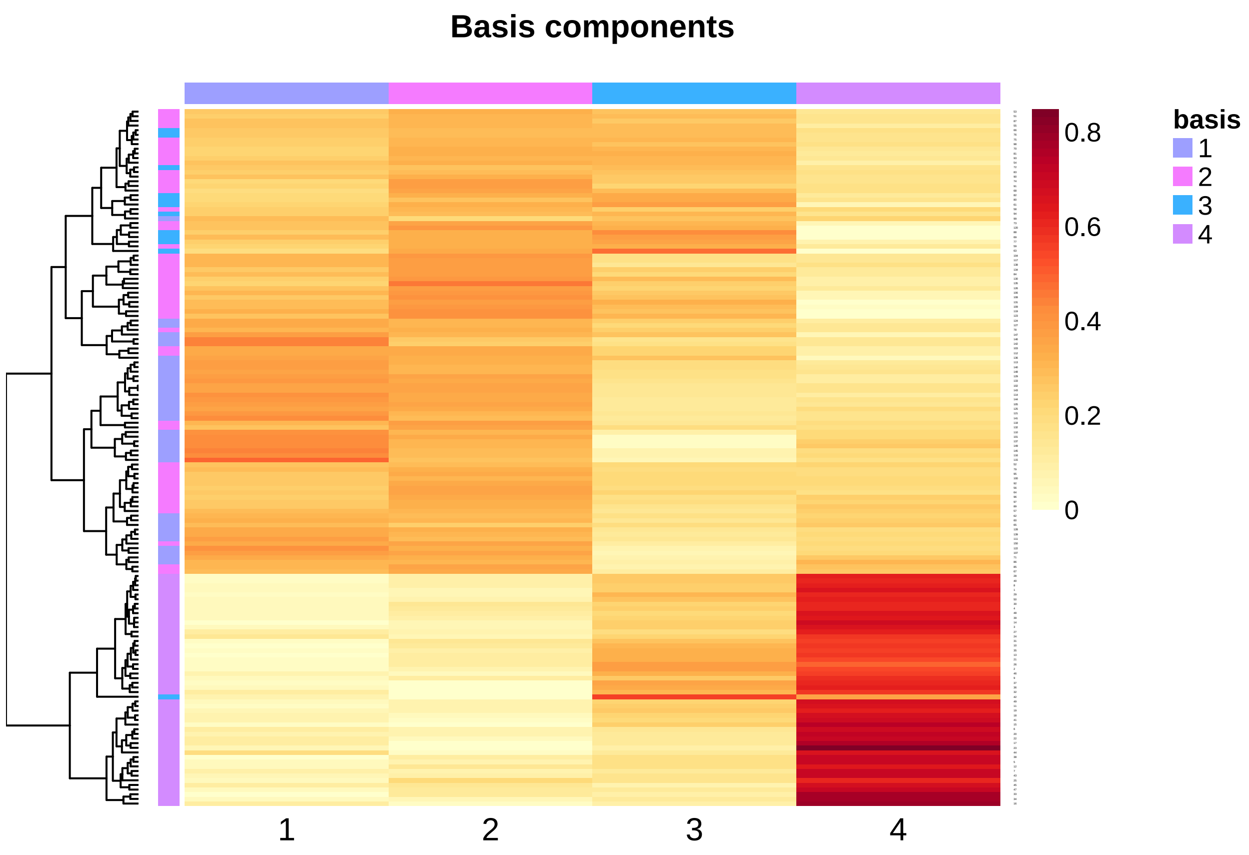
$$V_{mxn} = W_{mxp}H_{pxn}$$

Where p is specified to the algorithm. p can be thought of as the number of features to search for. The column vector W can be thought of as the features, and the vector H van be thought of as the weights for these features.

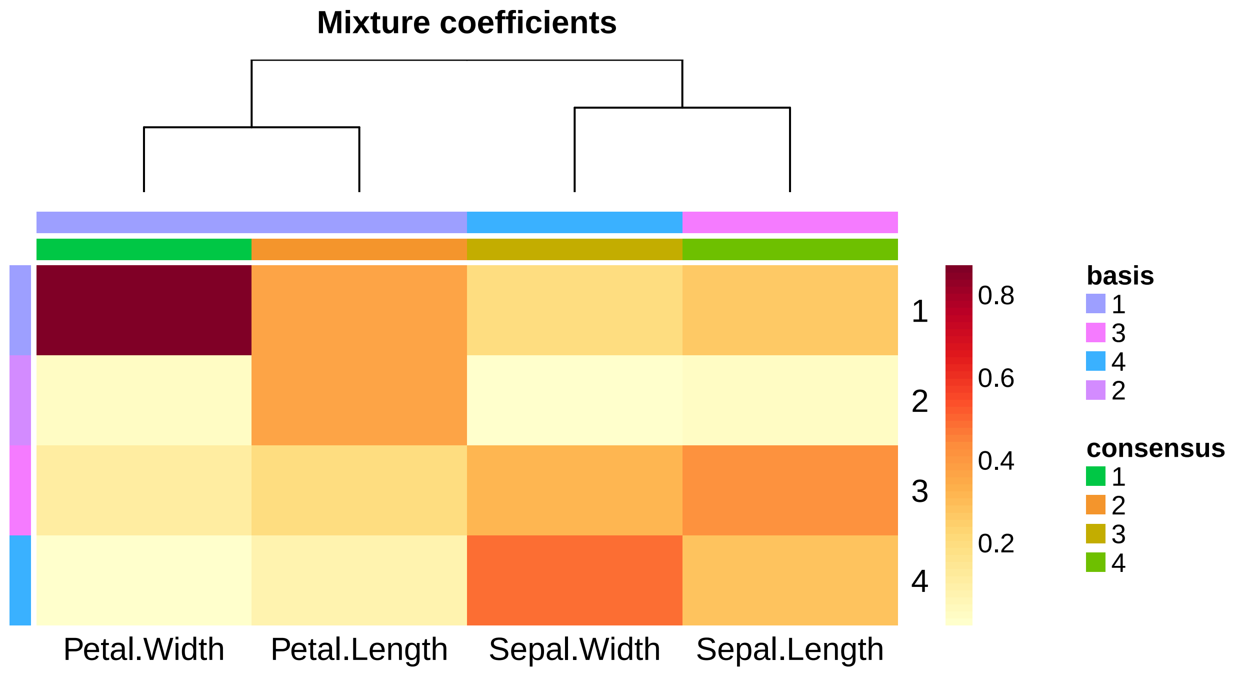
```
iris_clust2 <- iris[,1:4]

nmf_out <- nmf(iris_clust2, 4, set.seed(304543), nrun= 100)
W <- nmf_out@fit@W
H <- nmf_out@fit@H

#W
basismap(nmf_out)
```



```
#H
coefmap(nmf_out)
```


Mixture coefficients
Petal.Width
Petal.Length
Sepal.Width
Sepal.Length
1
2
3
4
0.8
0.6
0.4
0.2
basis
1
3
4
2
consensus
1
2
3
4

###Homework

```r
data(iris)
iris_subs <- iris[,c(1, 2, 3, 4)]
species <- iris[,5]
```

1. Run PCA, ICA, and NMF on the iris dataset.

a. Explain your inputs and outputs from each algorithm. For instance, in the input for the NMF example above, out inputs were a 25 x 8 matrix, and a rank of 4. The output was a matrix W with dimensions 25x4, and a matrix H with dimensions 4x8. We plotted the basis matrix (W), where each columns corresponds to a feature… etc.

2. Use the silhouette function in the cluster package to find the optimal number of clusters for kmeans for the iris dataset. Then cluster using hierarchical clustering and kmeans clustering. Does the data cluster by species?

## Optional material

On PCA:

Eigen Vectors and Eigen Values http://www.visiondummy.com/2014/03/eigenvalues-eigenvectors/ Linear Algebra by Prof. Gilbert Strang https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/ video-lectures/ http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenva

On ICA:

Independent Component Analysis: Algorithms and Applications https://www.cs.helsinki.fi/u/ ahyvarin/papers/NN00new.pdf Tutorial on ICA taken from http://rstudio-pubs-static.s3.amazonaws. com/93614_be30df613b2a4707b3e5a1a62f631d19.html