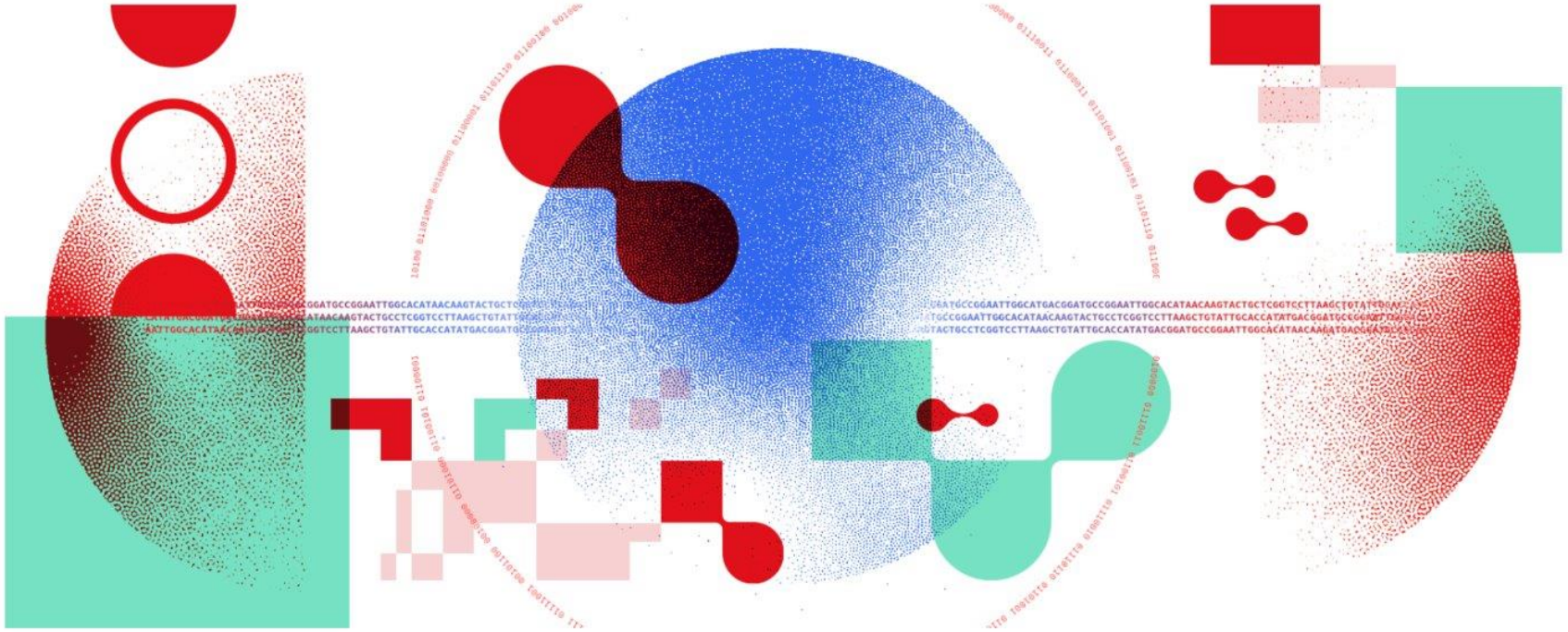


SIB
Swiss Institute of Bioinformatics

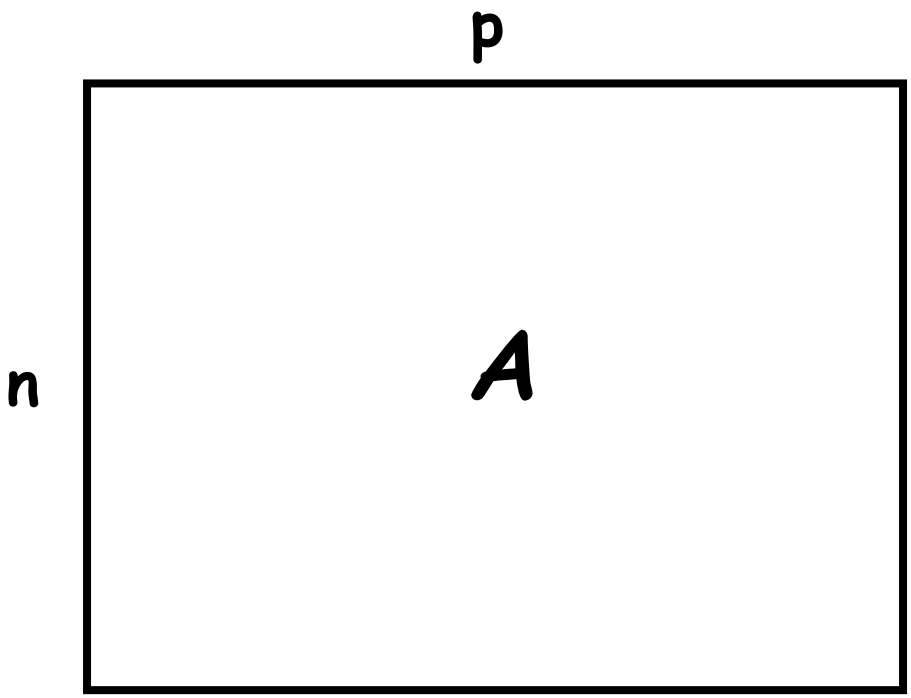
Introduction to Statistics and Data
Lausanne, January 2026

Rachel Marcone and Joao Lourenco

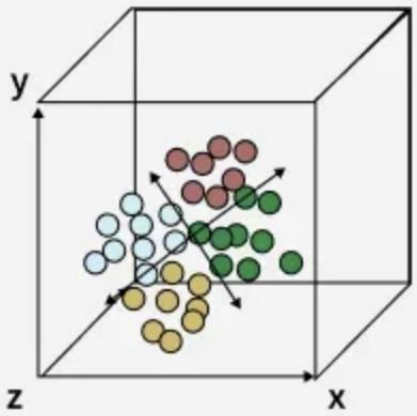
PCA and Clustering

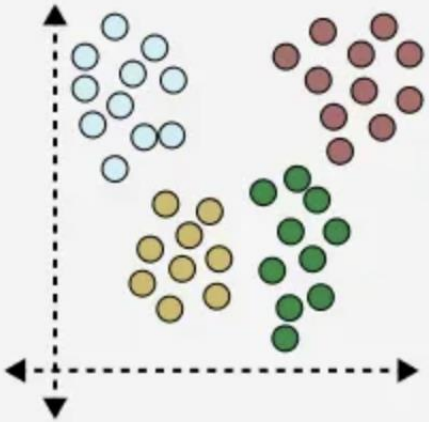
Dimension reduction (with PCA)

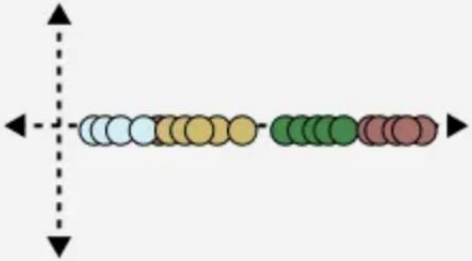
Starting point: Big Data



p=3 p=2 p=1

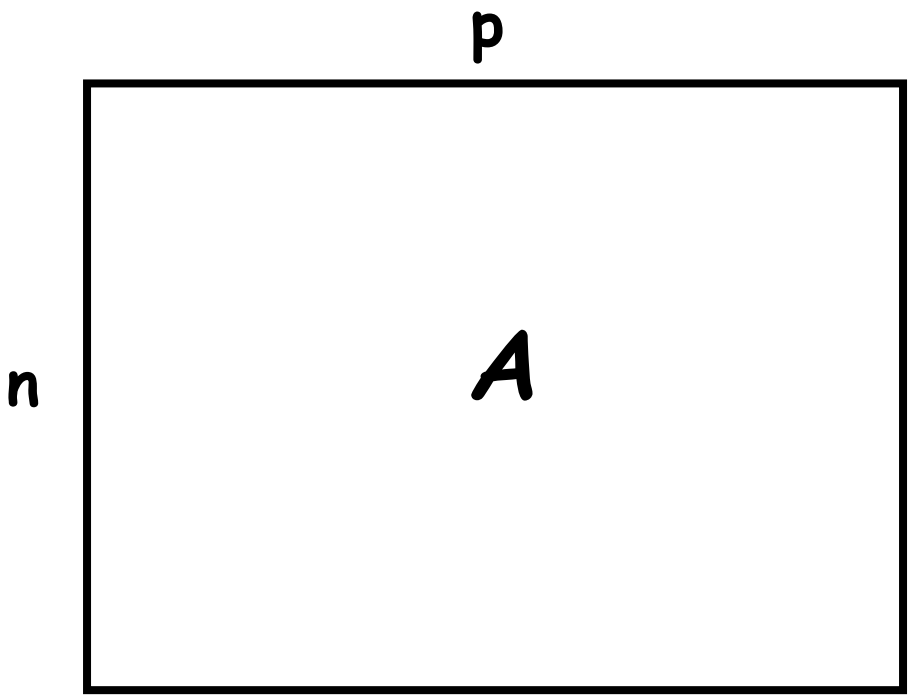




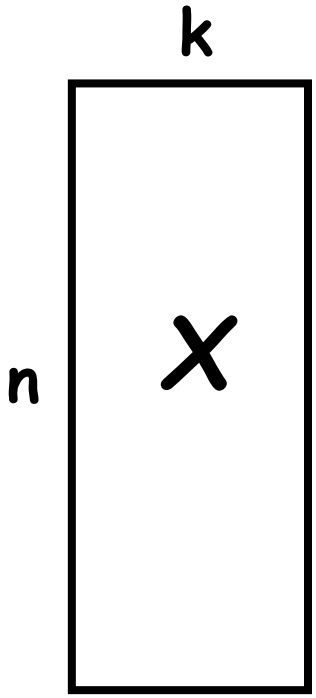


p > 3 ?

Starting point: Big Data

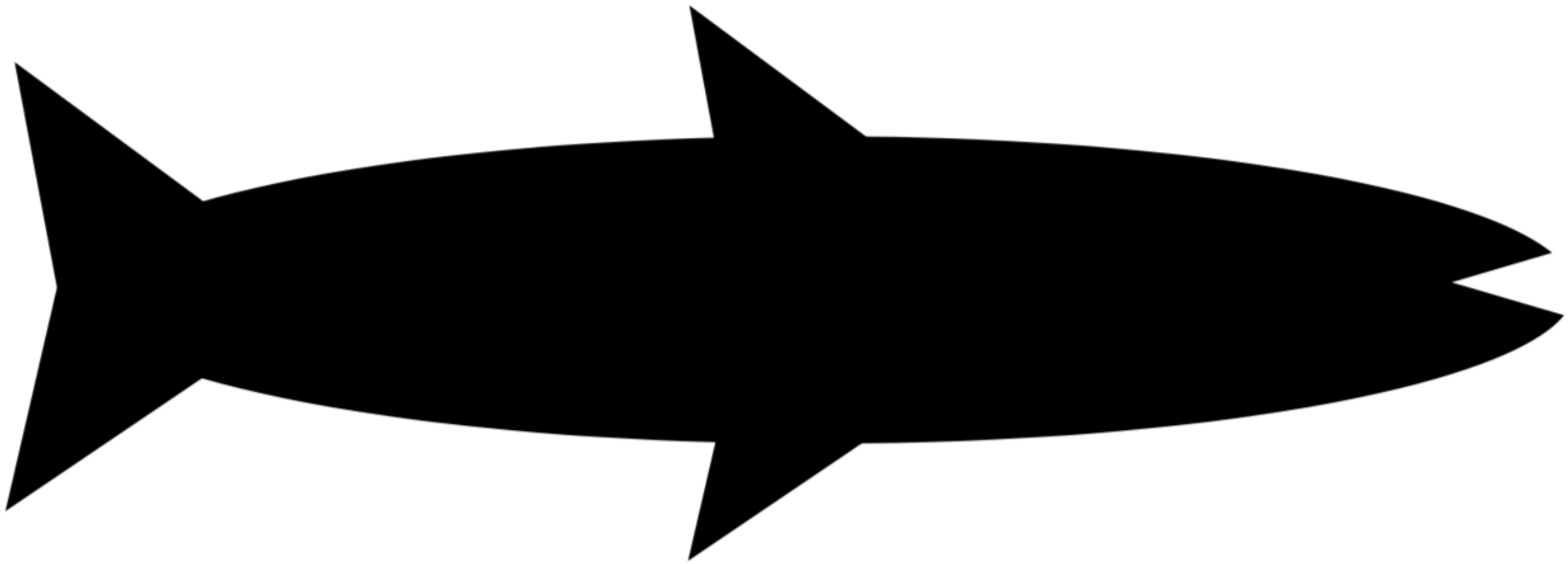


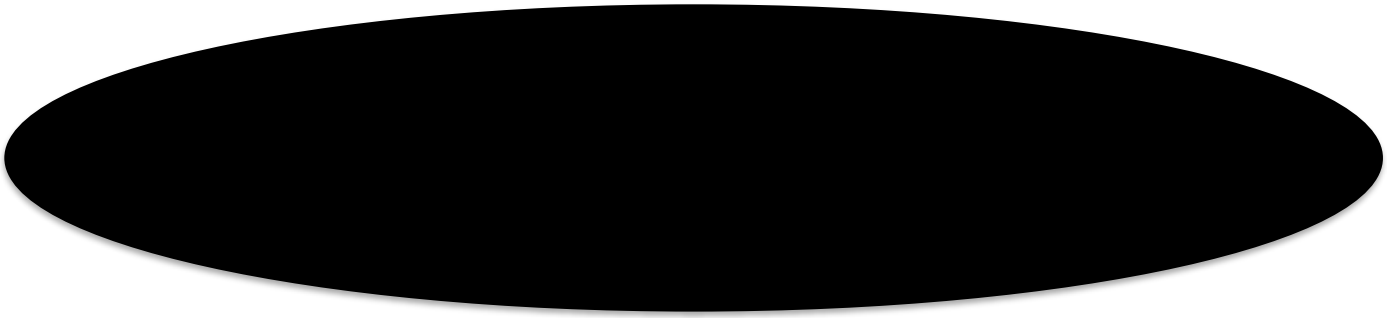
End result: human readable



Reducing the **dimension** of the variable space is called

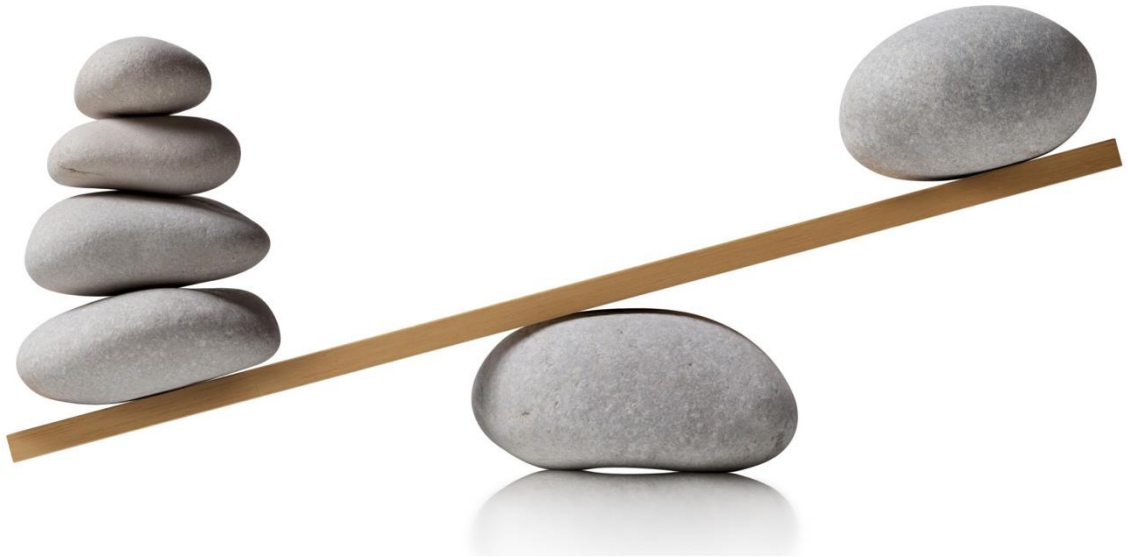
Dimensionality Reduction





clarity of representation

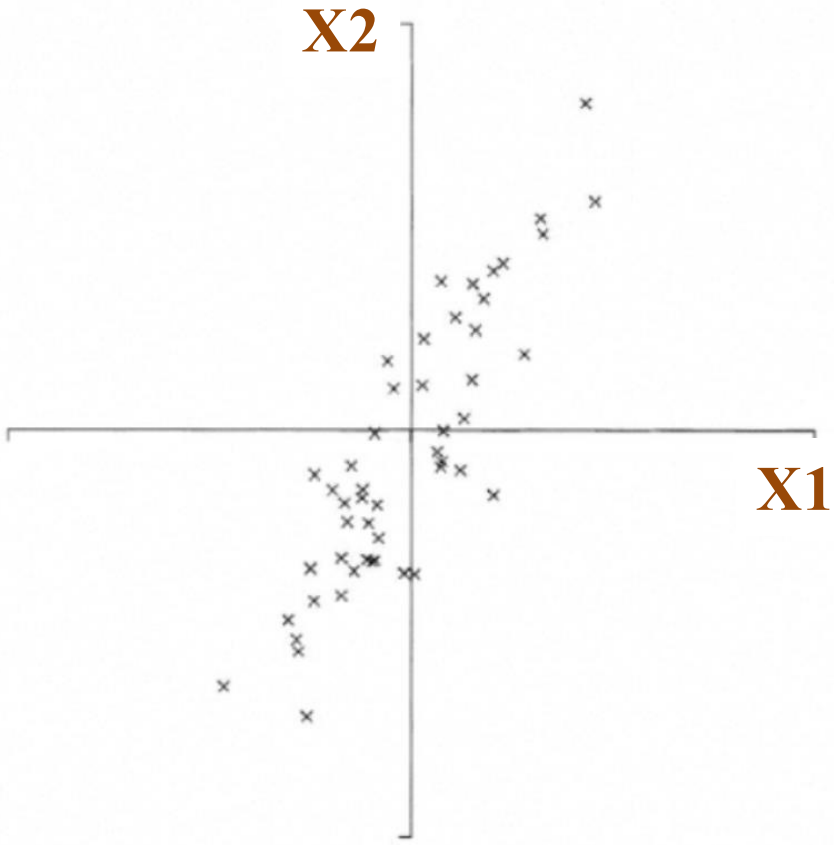
Over-simplification

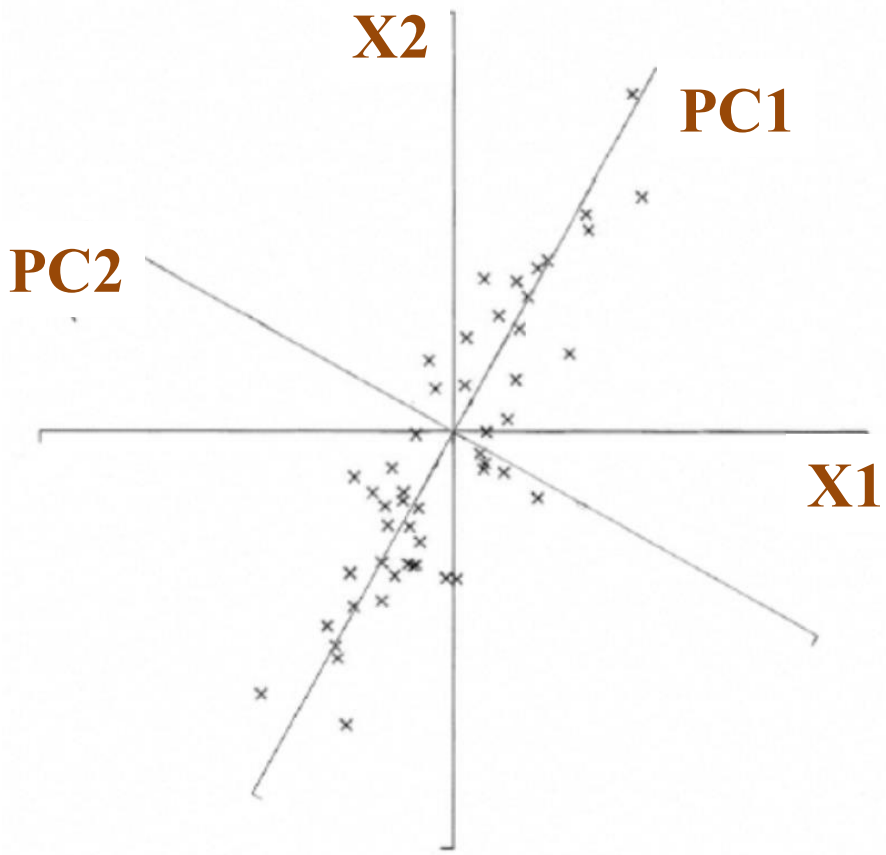


The objective of dimensionality deduction is to reduce the number of variables in a dataset while preserving as much important information as possible

Principal Component Analysis (PCA)

Pearson (1901) and Hotelling (1933)



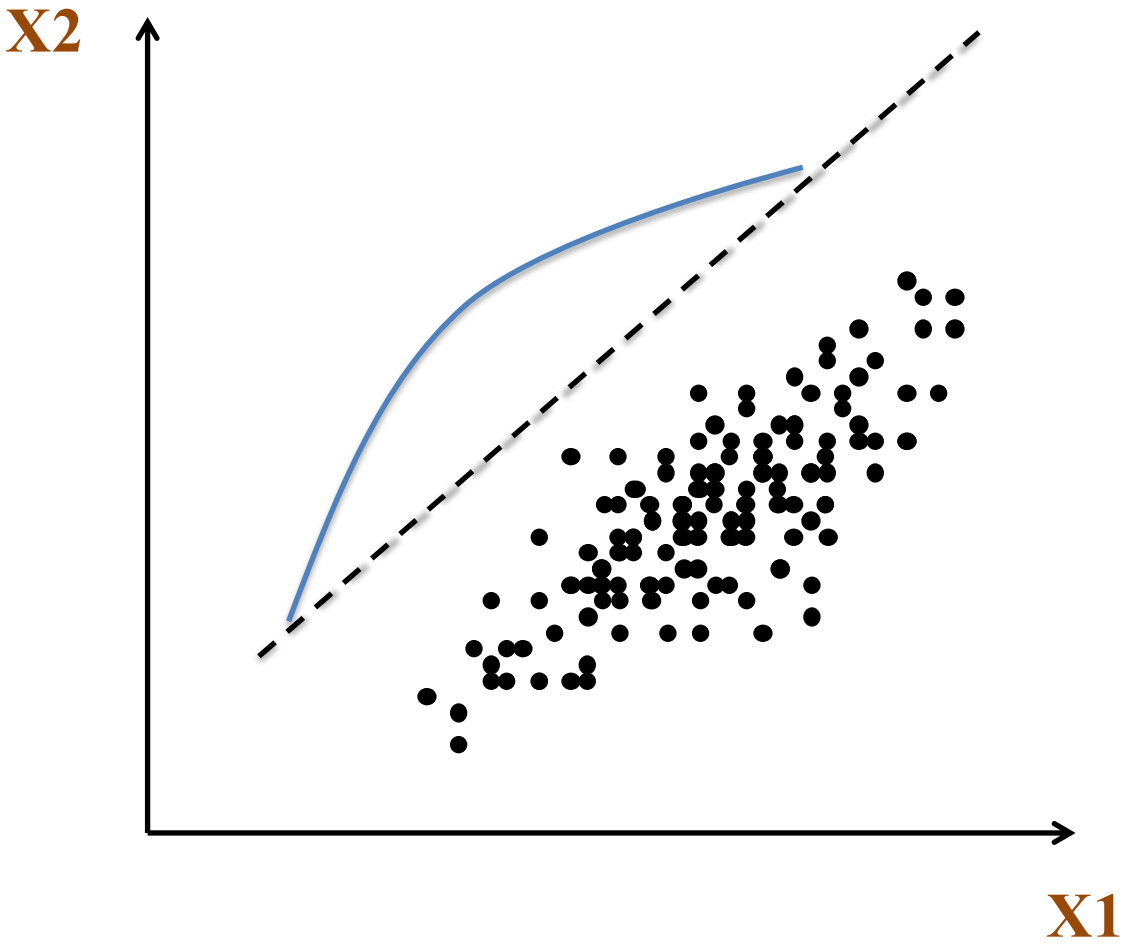


New "components" are created by rotating the existing axes (var). "Principal": components are ranked according to the proportion of variability they "capture" - the objective is to drop the least important ones.

PCA- Principal component analysis

1. Find the 1st principal component (the one with the largest possible variance)

PCA- Principal component analysis



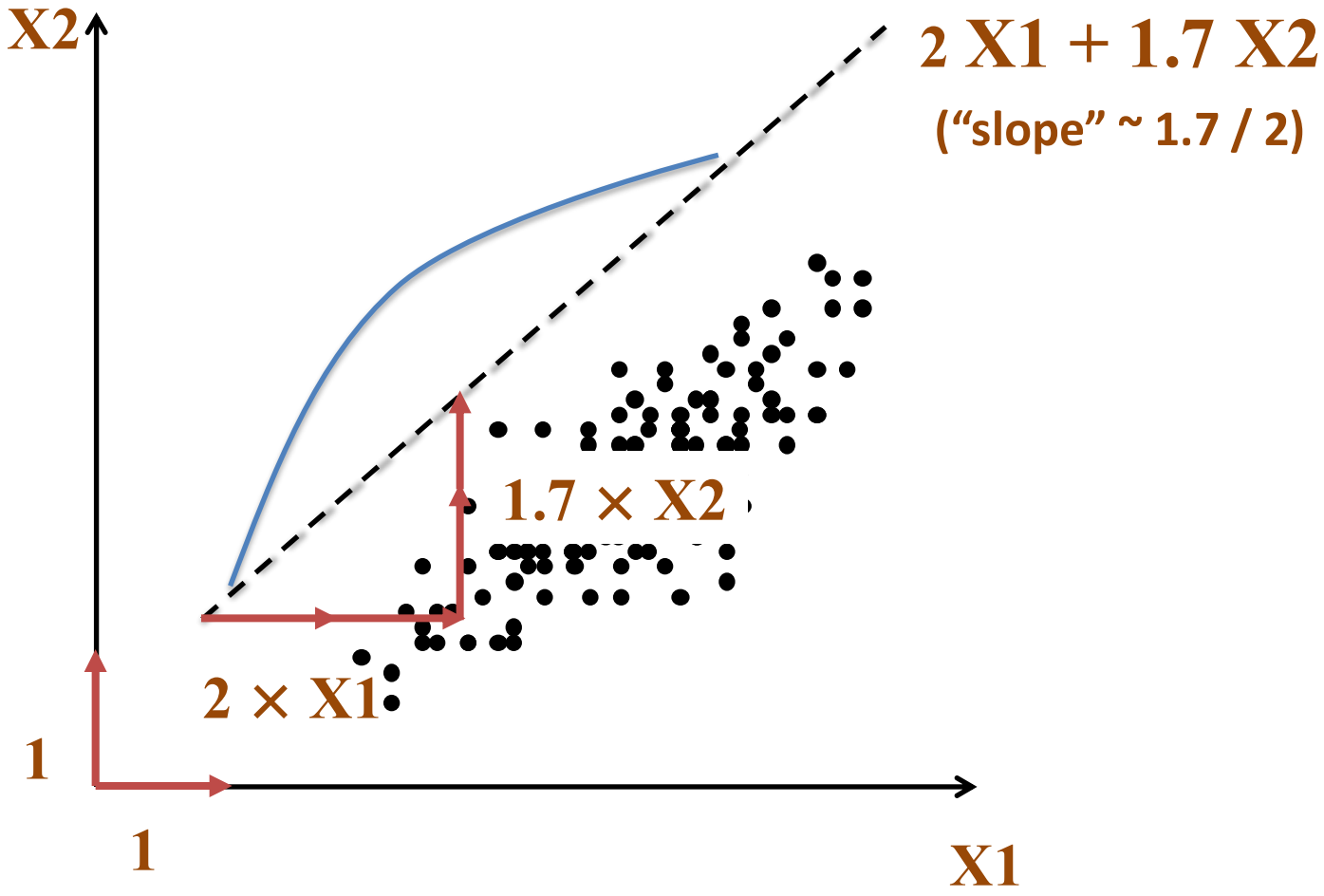
PCA- Principal Component Analysis

Data is most often scaled (and centered) before perfoming PCA!

If one variable is on a different scale than another, it will dominate the PCA procedure as the largest variance might be observed there, and the low dimension plot will really just be visualizing that dimension.

If we scale -> all variables have the same weight
If we don't scale -> some variables will have more weight than others

PCA- Principal component analysis

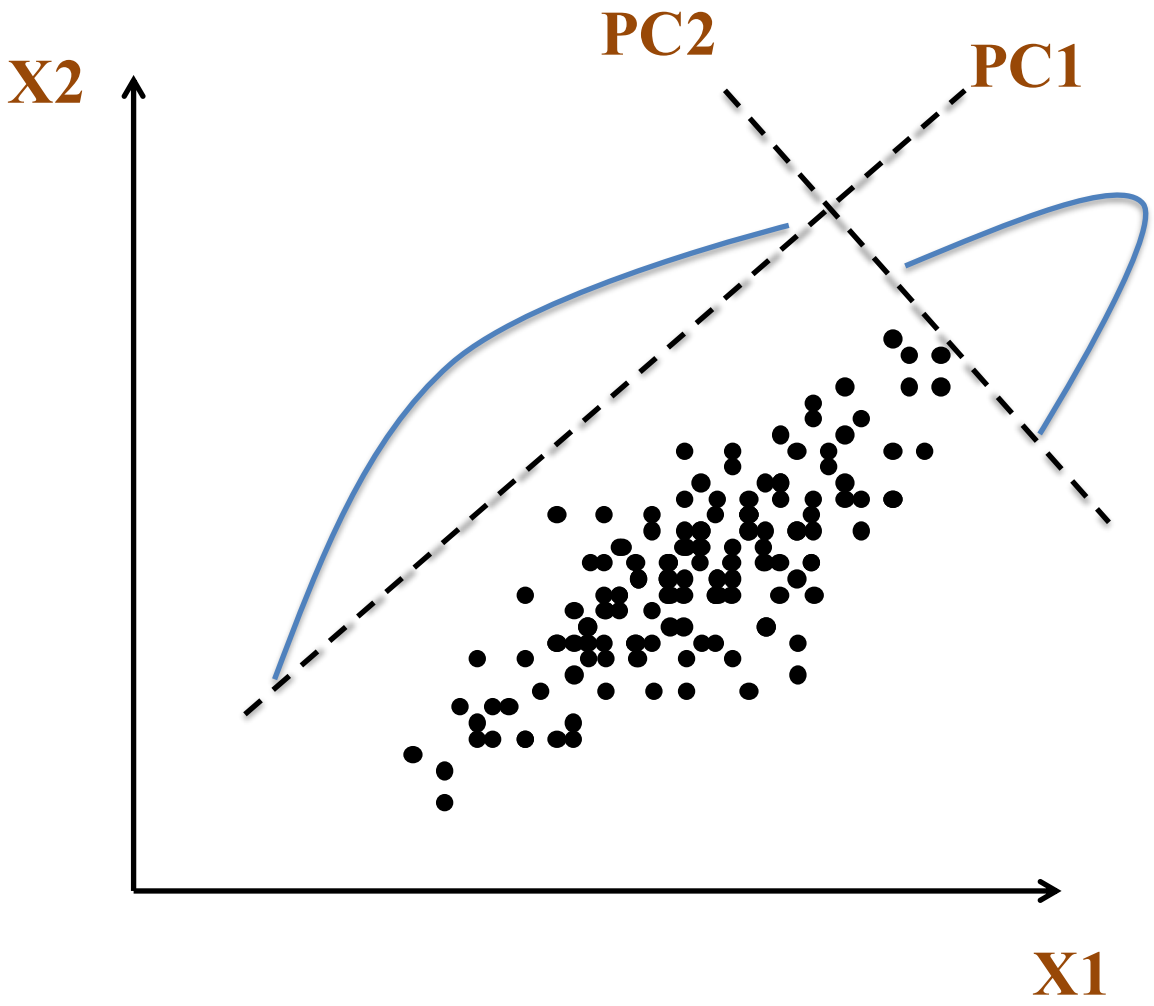


PCA- Principal Component Analysis

1. Find the 1st principal component (the one with the largest possible variance)

2. Select uncorrelated (orthogonal) axes as the next principal component

PCA- Principal Component Analysis

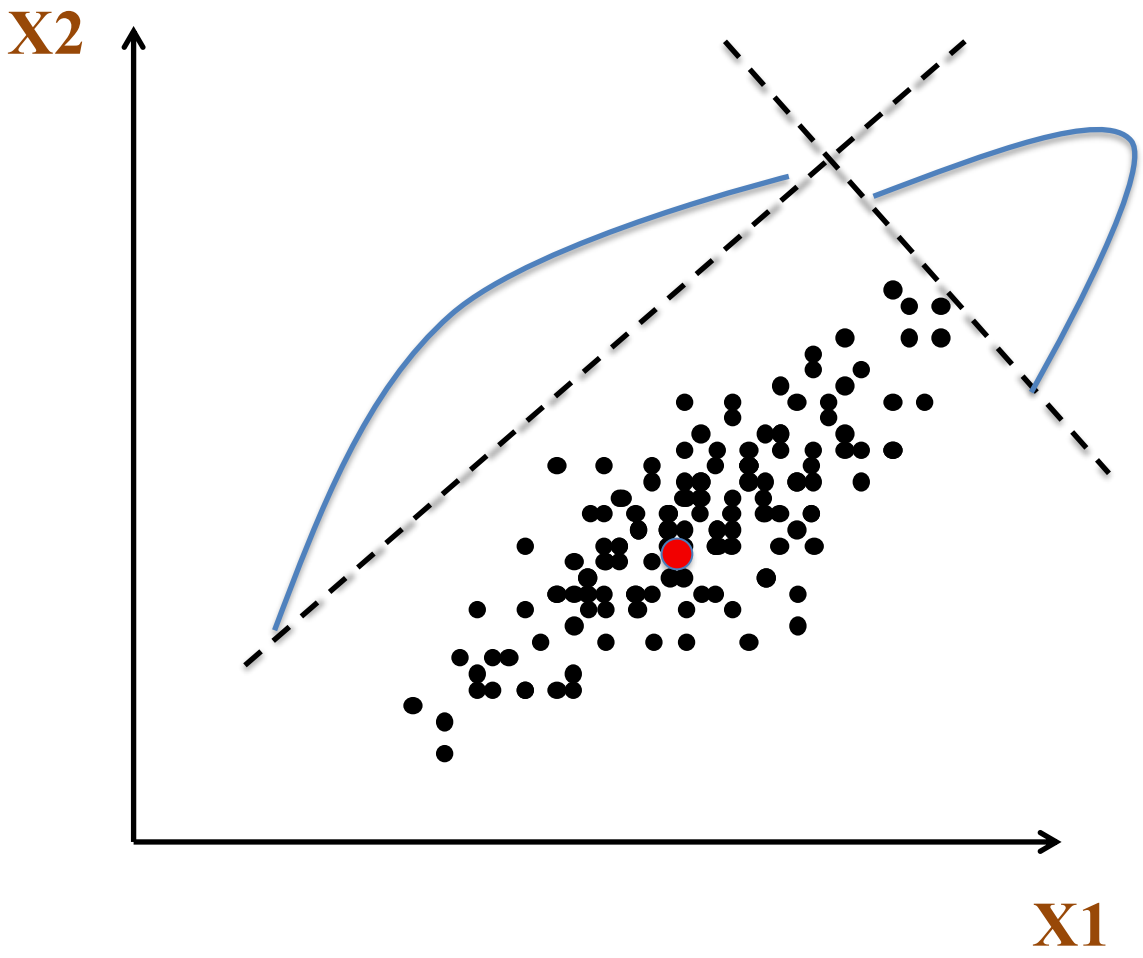


PCA- Principal Component Analysis

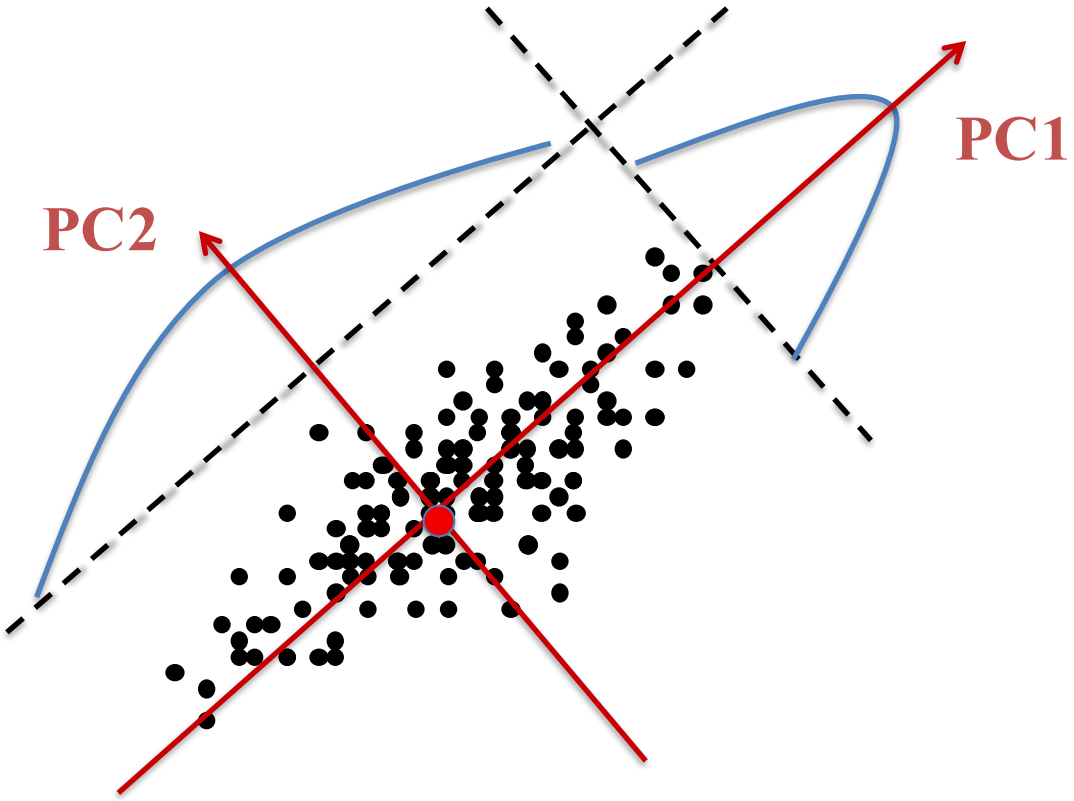
1. Find the 1st principal component (the one with the largest possible variance)

2. Select uncorrelated (orthogonal) axes as the next principal component

3. Make the origin coincide with the centroid


X2
X1

PCA- Principal Component Analysis



PCA- Principal Component Analysis

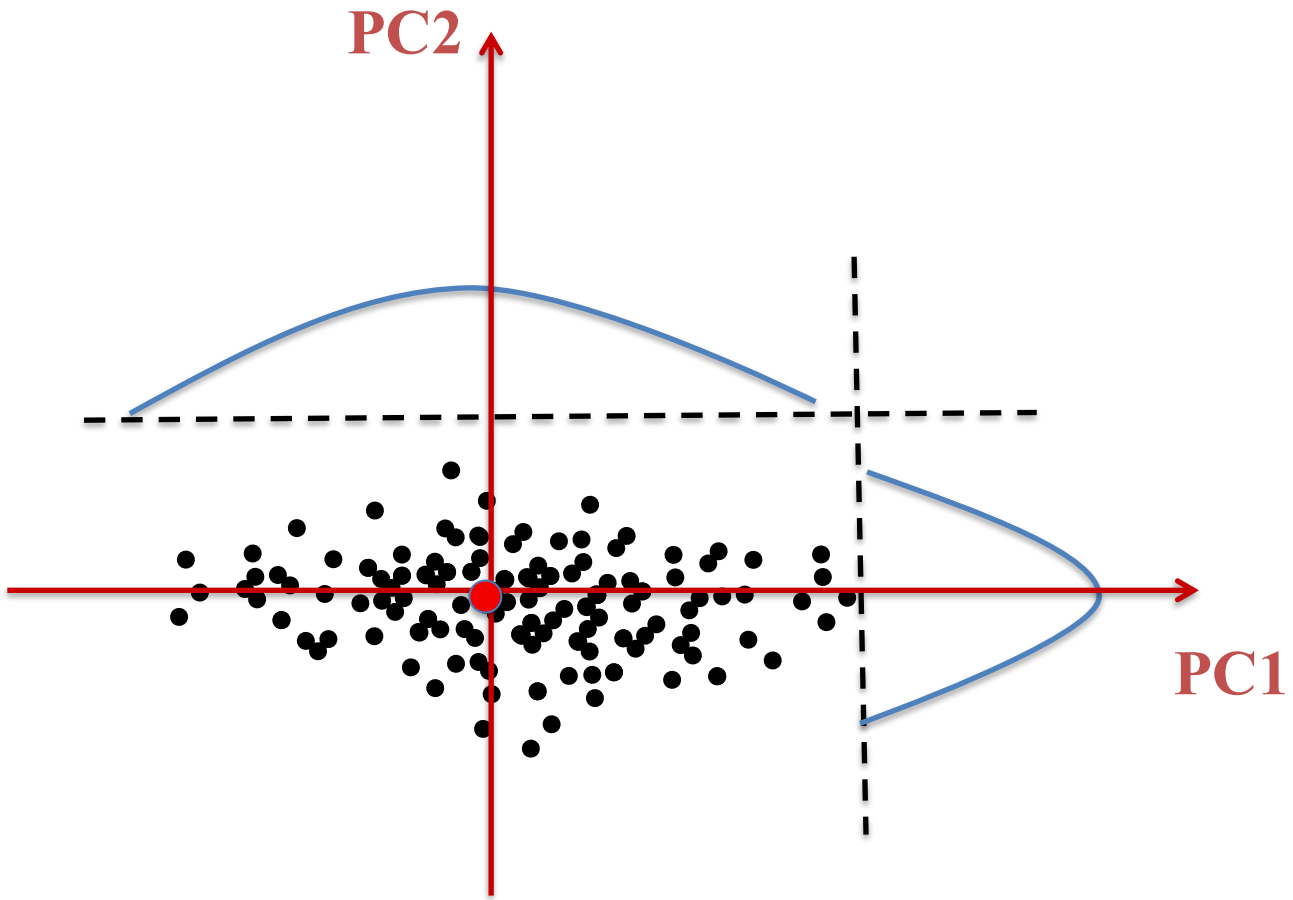
1. Find the 1st principal component (the one with the largest possible variance)

2. Select uncorrelated (orthogonal) axes as the next principal component

3. Make the origin coincide with the centroid

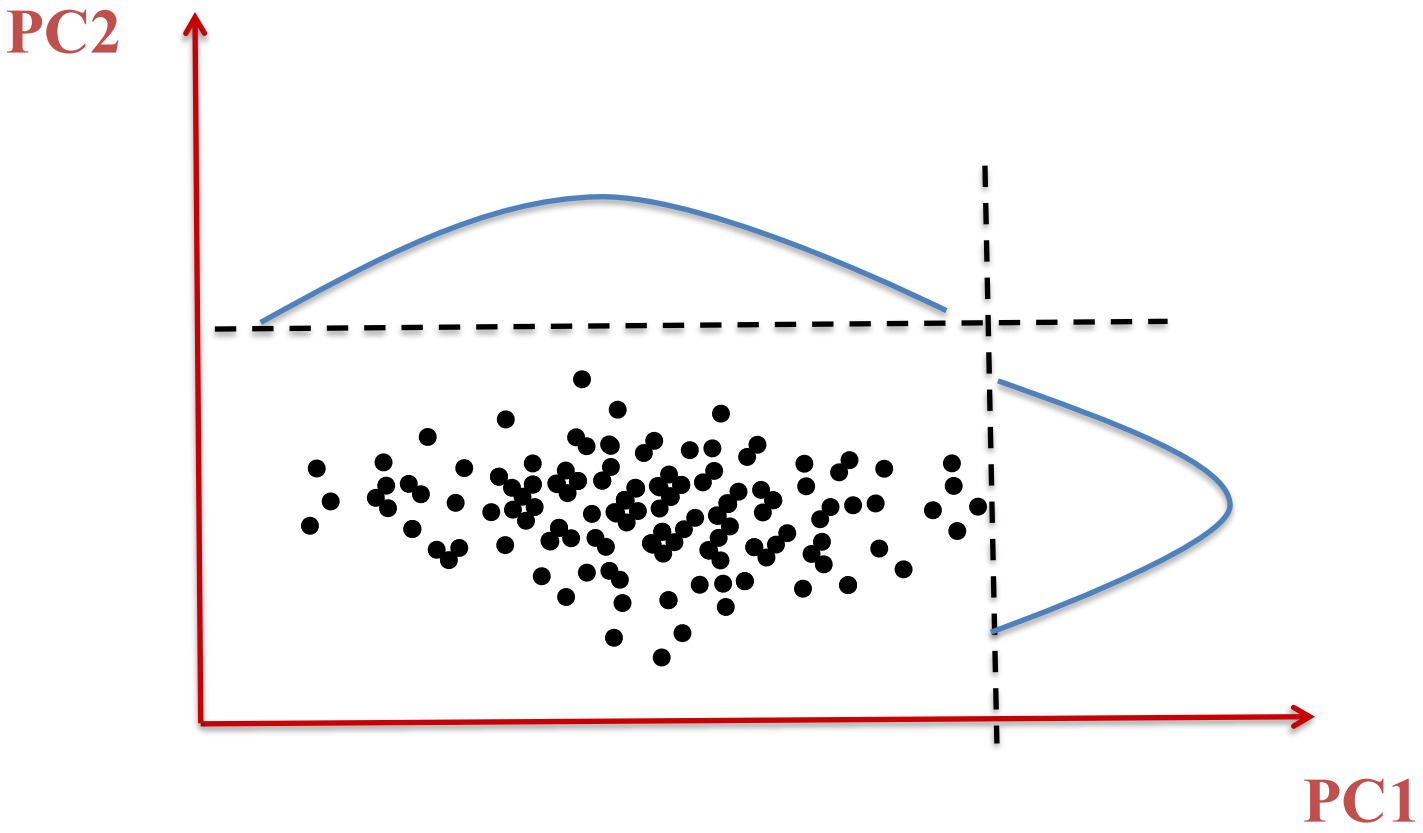
4. Rotate

PCA- Principal Component Analysis

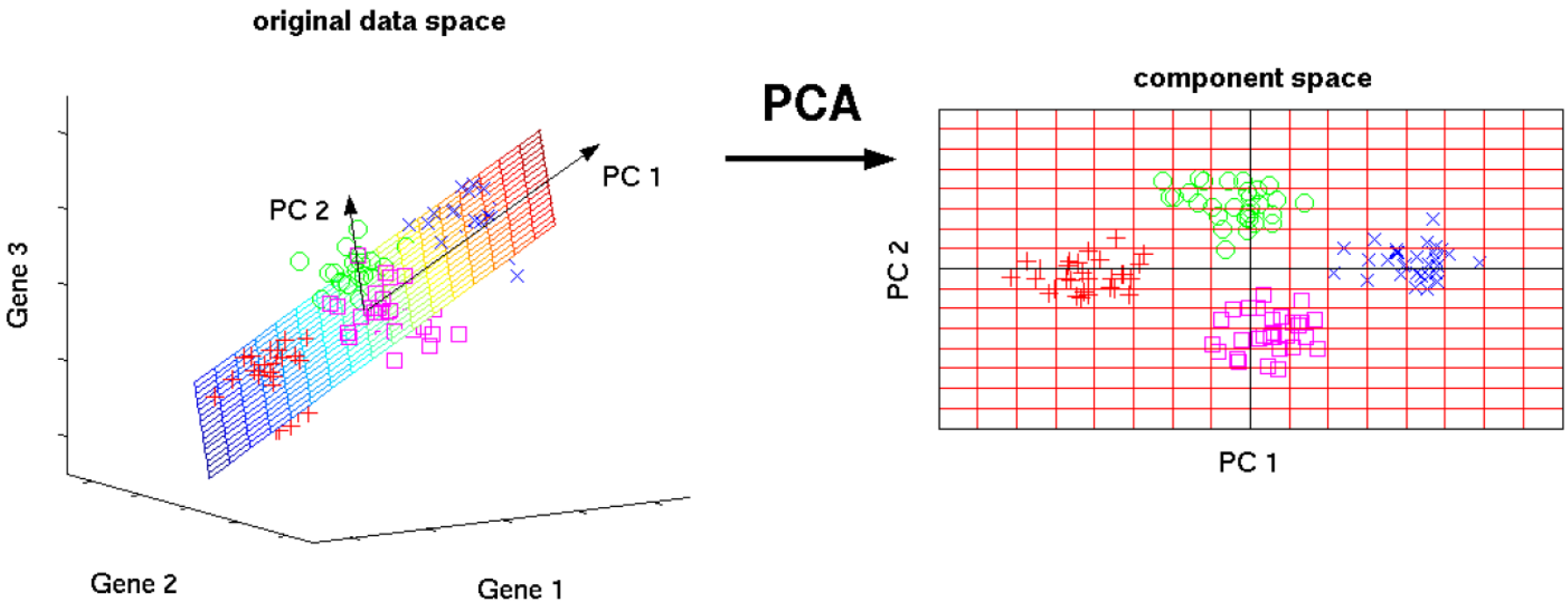


Transform the original data to align with the new axes

PCA- Principal Component Analysis



Without centering


original data space
PC 2
PC 1
Gene 3
Gene 2
Gene 1
PCA
component space
PC 2
PC 1

Mathematically

You calculate the **covariance matrix** meaning a matrix containing two-by-two covariances

- if positive then : the two variables increase or decrease together (correlated)
- if negative then : One increases when the other decreases (Inversely correlated)

If data was standardized (centered and scaled), then the **correlation matrix** is used instead of the covariance, Corr(X,Y) = Cov(X,Y)/sd(x)*sd(y)

Decompose the Cov matrix in $\mathbf{PDP}^{-1}$, where **P** is the matrix of eigenvectors and **D** is the diagonal matrix with eigenvalues on the diagonal and values of zero everywhere else.

Mathematically

- Eigenvectors are the directions of the axes where there is the most variance (this is something you can prove mathematically!)
- Eigenvalues are the coefficients attached to eigenvectors, which give the amount of variance carried in each Principal Component (the magnitude, or how important each PC is).
- Eigenvectors and eigenvalues define the PCs

PCA – Principal Component Analysis

1. Find the 1st principal component (the one with the largest possible variance)

2. Select uncorrelated (orthogonal) axes as the next principal component

3. Make the origin coincide with the centroid

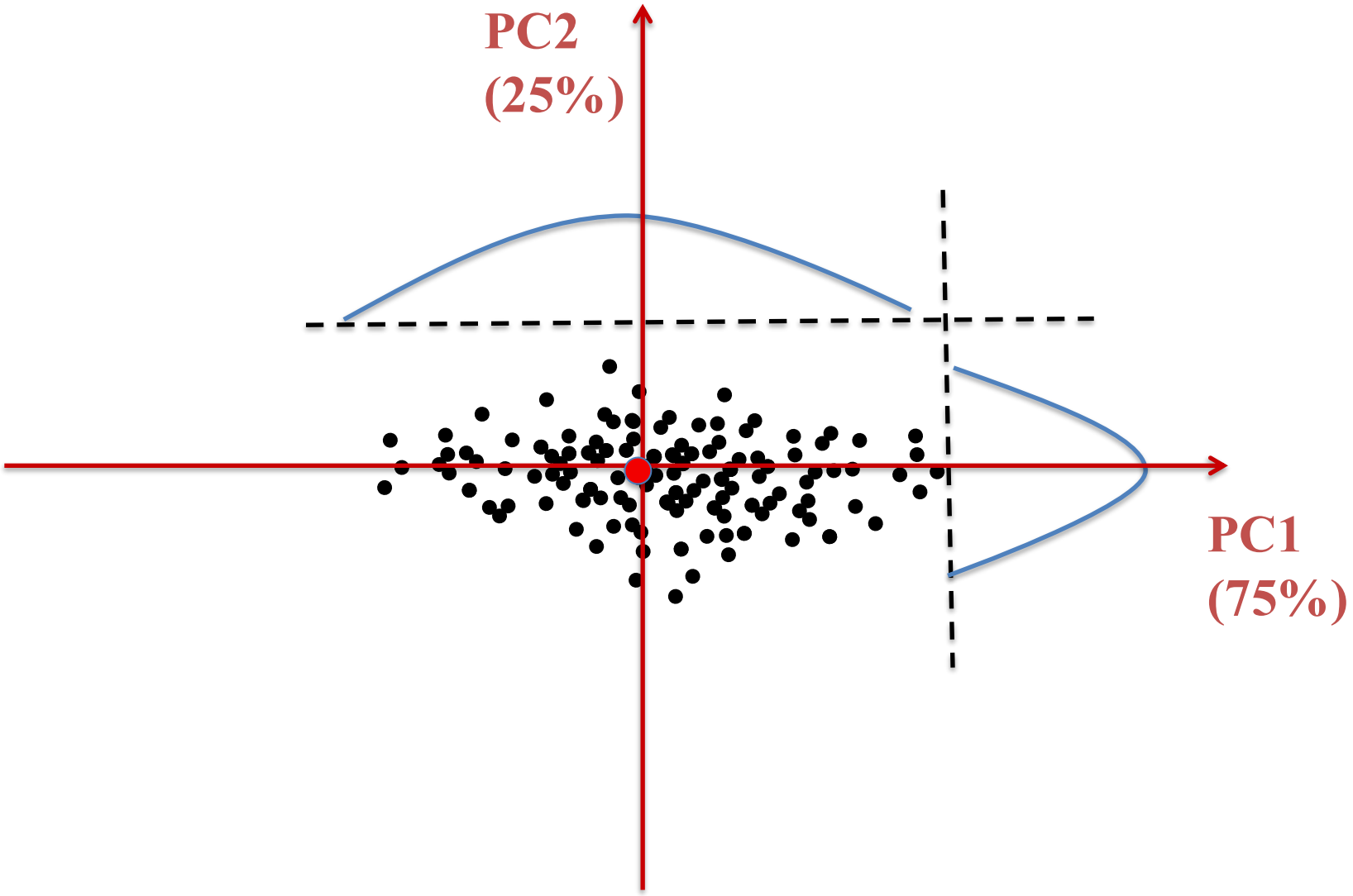
4. Rotate

5. Determine the proportion of the variation that is explained by each PC

Mathematically

- To compute the percentage of variance accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues.

PCA- Principal Component Analysis

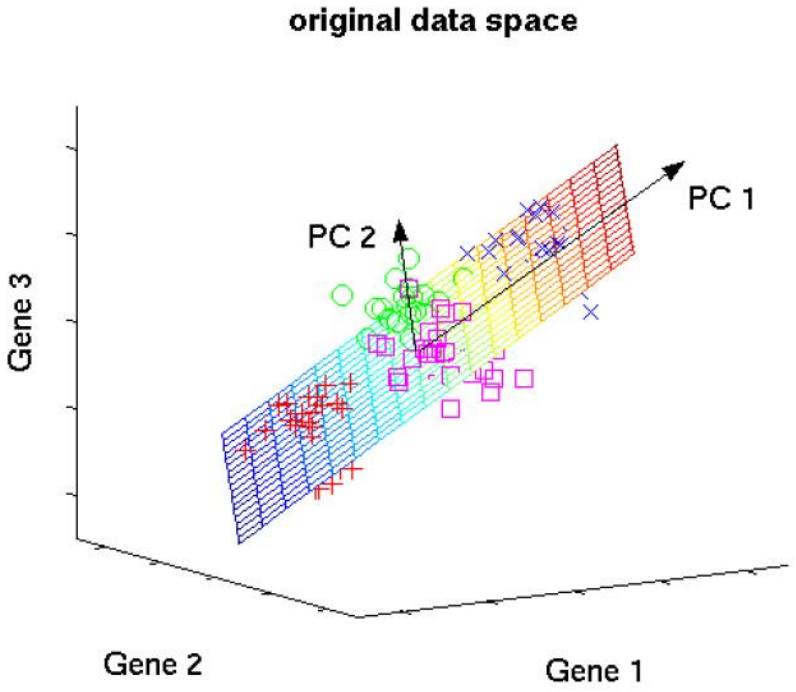


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2

...

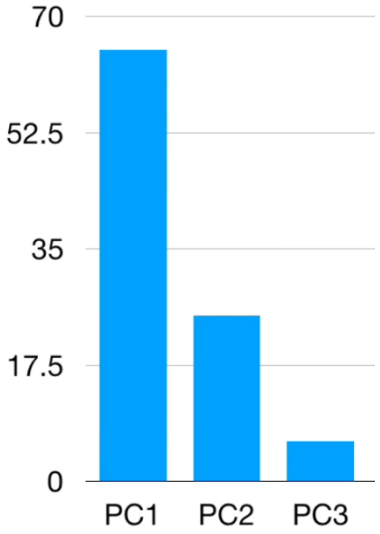
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2

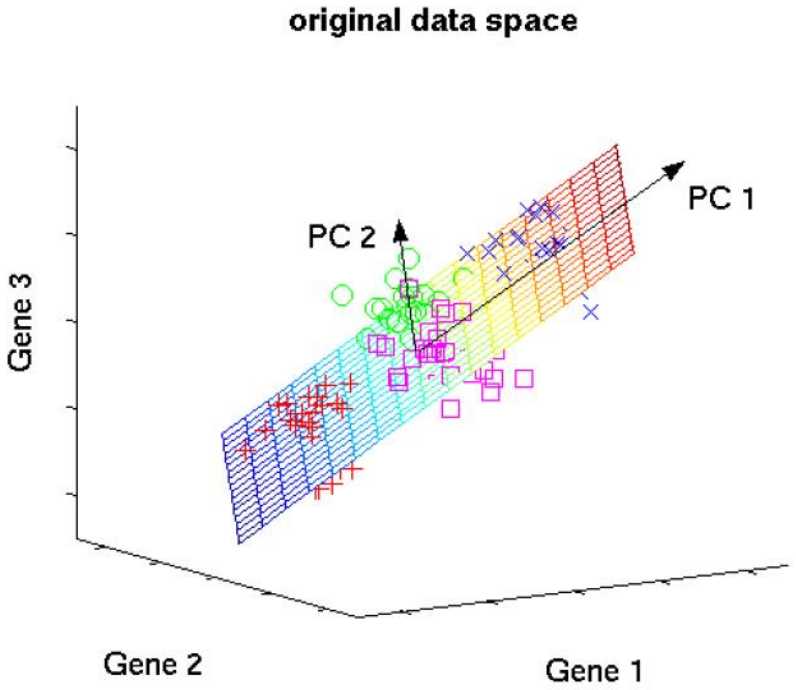
...



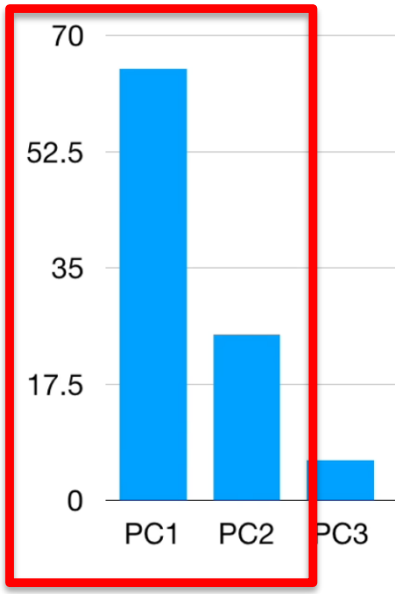
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2

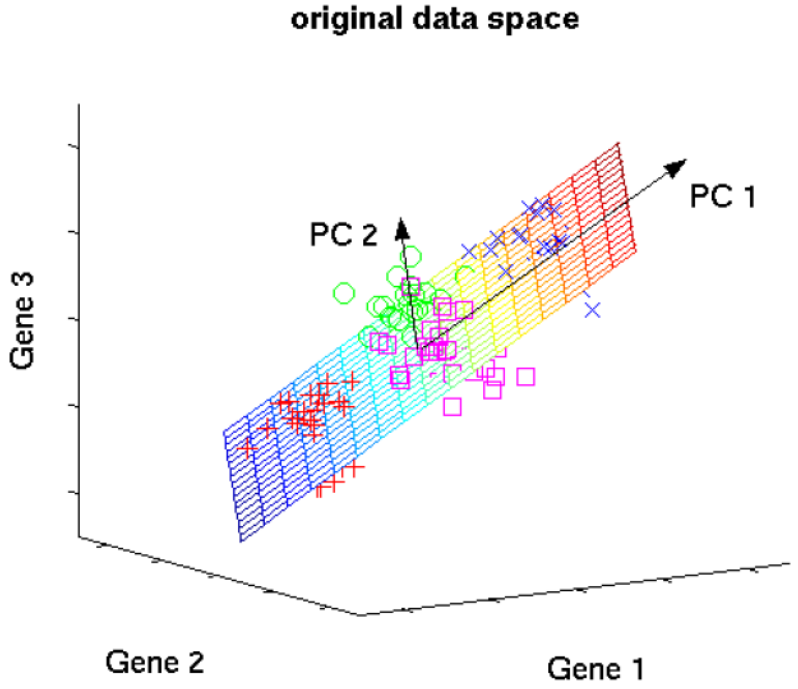
...



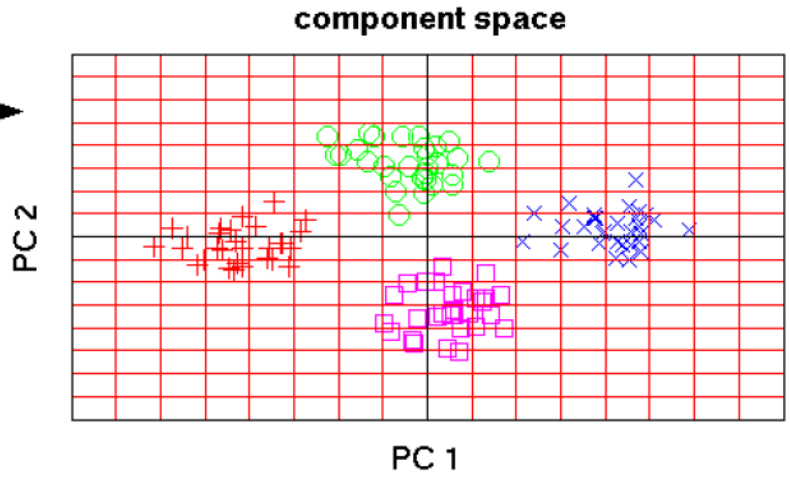


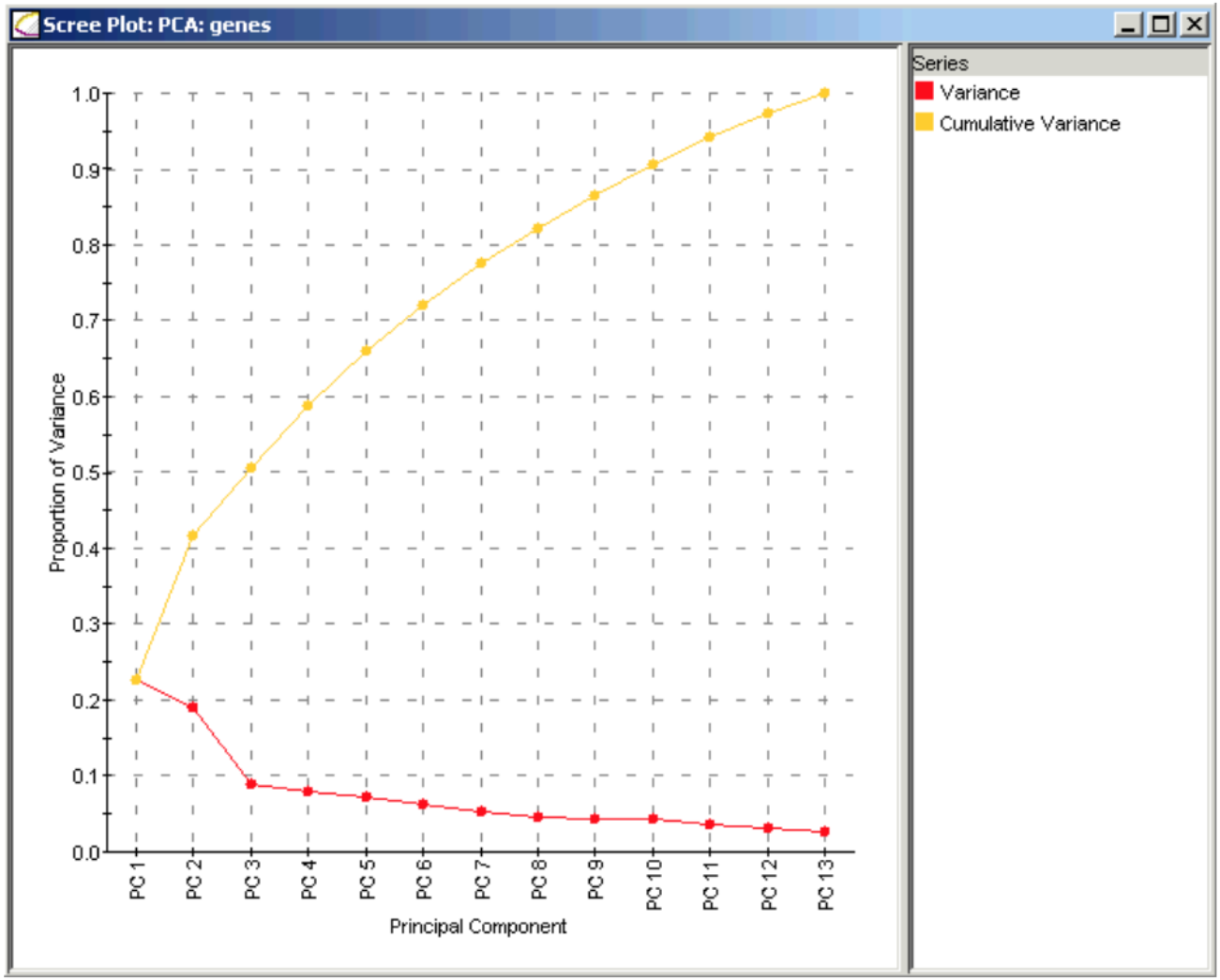
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2





PCA





Scree Plot for Genetic Data. (Source.)

https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

How many PCs?

- Method 1: We **arbitrarily select** a number of principal components to include. Suppose I wanted to keep five principal components in my model. In the genetic data case above, these five principal components explains about 66% of the total variability that would be explained by including all 13 principal components.
- Method 2: Suppose I wanted to include **enough principal components to explain 90%** of the total variability explained by all 13 principal components. In the genetic data case above, I would include the first 10 principal components and drop the final three variables from ***Z****.
- Method 3: Here, we want to "**find the elbow**." In the scree plot above, we see there's a big drop in proportion of variability explained between principal component 3 and the following. In this case, we'd likely include the first three features and drop the remaining features. As you can see, this method is a bit subjective as "elbow" doesn't have a mathematically precise definition and, in this case, we'd include a model that explains only about 42% of the total variability.

In R

```
> data(iris)

> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

> ?prcomp

> pca.iris.cov <- prcomp(iris[,1:4], center = TRUE, scale. = FALSE)
```

Center and Scale

- Why should we center, why should we scale ?
- Will see this through the exercises
- Centering or not centering, it will not change the result
- In prcomp, however, "PCA" is defined as computing the eigenvalues of the X^TX / (n-1) matrix, which in a centered data is exactly the covariance matrix, otherwise not.

Center and Scale

- Although proven not to be exactly true *, this will result in first PCs capturing the mean of the data as this "explains" most of the variance in the model.
- The scaling will determine if you compute eigenvectors on the covariance matrix (if unscaled) or on the correlation matrix (if scaled).
- This again (mostly) means that what you will capture in the first PCs is mostly what is in a bigger scale.

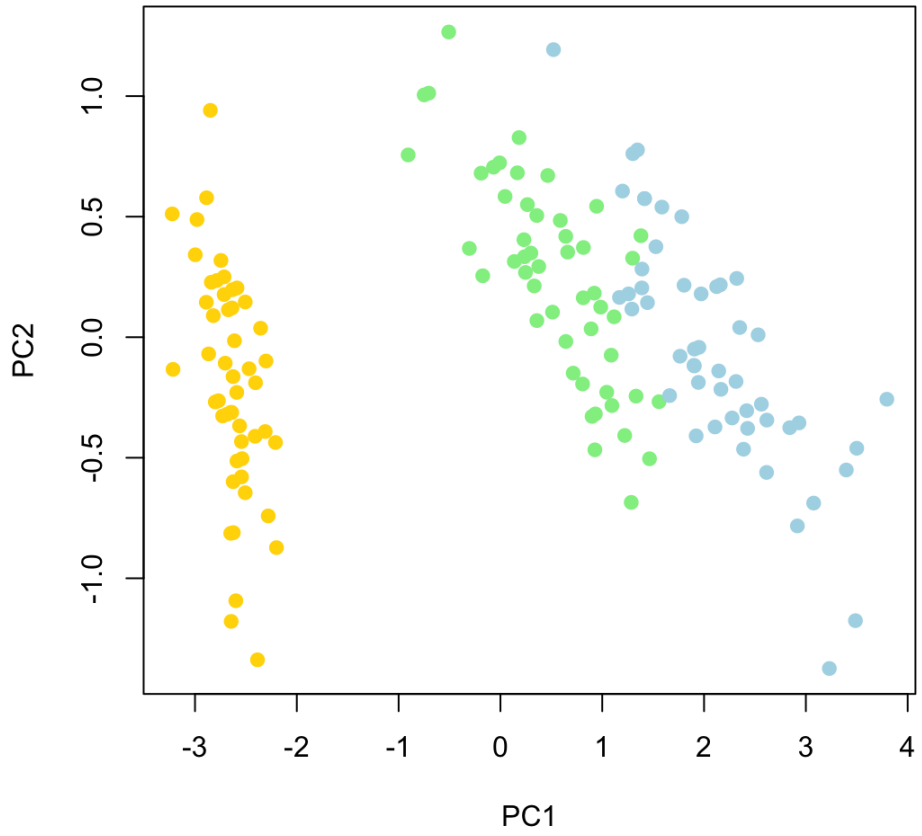
- * The Effect of Data Centering on PCA Models Neal B. Gallagher, Donal O'Sullivan, Manuel Palacios

In R

```
> head(pca.iris.cov$x)
            PC1        PC2         PC3          PC4
[1,] -2.684126 -0.3193972  0.02791483  0.002262437
[2,] -2.714142  0.1770012  0.21046427  0.099026550
[3,] -2.888991  0.1449494 -0.01790026  0.019968390
[4,] -2.745343  0.3182990 -0.03155937 -0.075575817
[5,] -2.728717 -0.3267545 -0.09007924 -0.061258593
[6,] -2.280860 -0.7413304 -0.16867766 -0.024200858
```

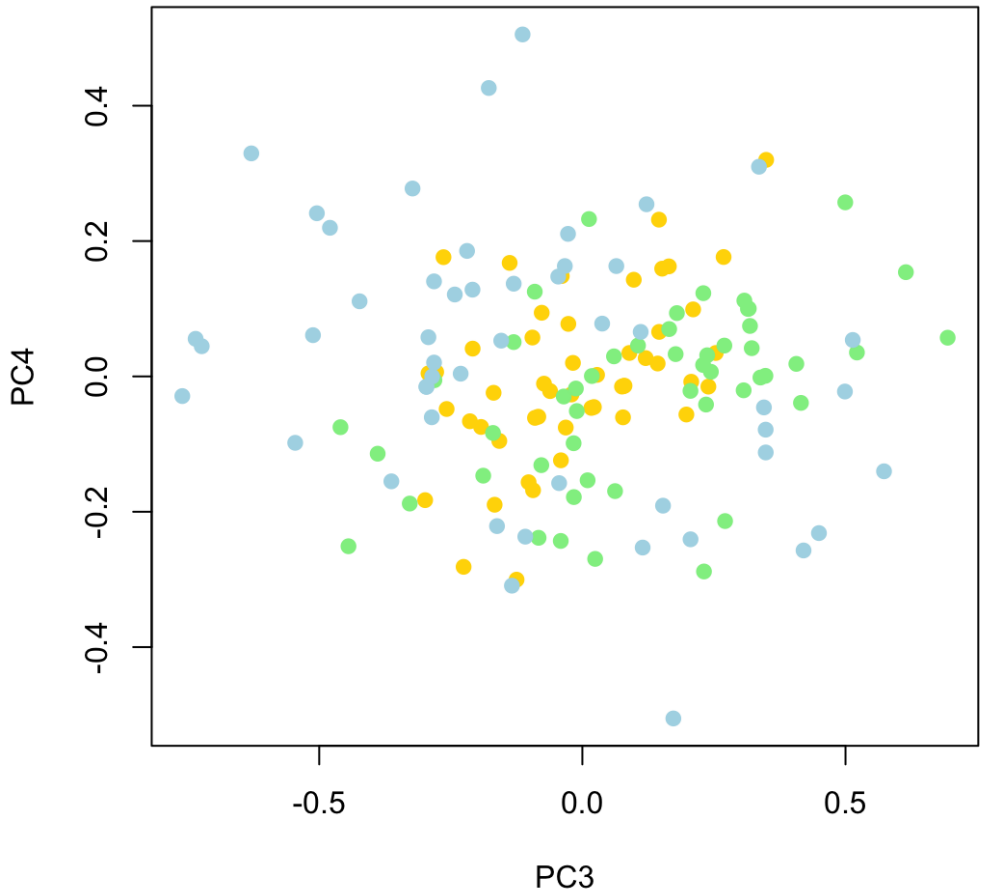
In R

```
> cols <- c(setosa = "gold", versicolor = "lightgreen", virginica = "lightblue")
> plot(pca.iris.cov$x, col = cols[iris$Species], pch = 19)
```



In R

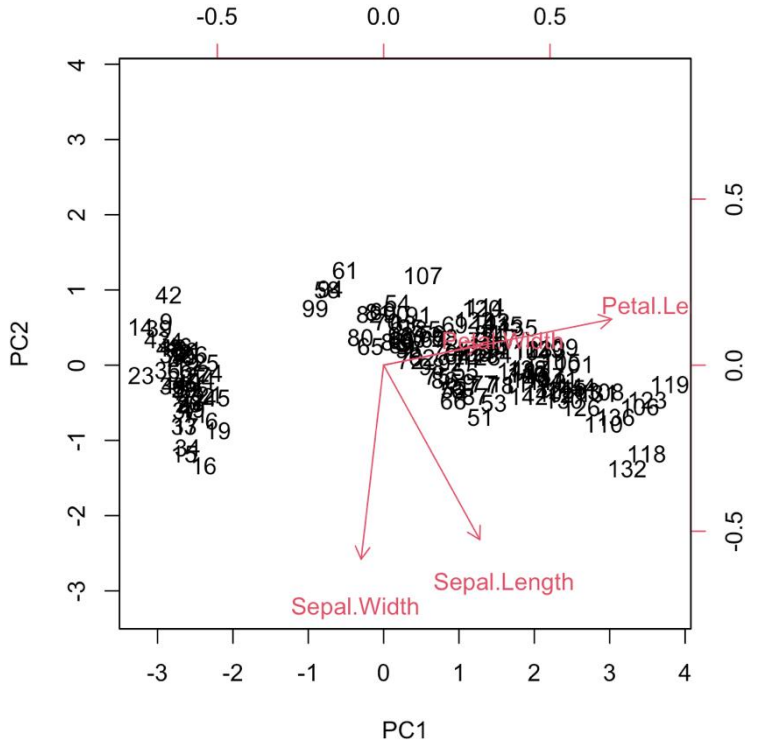
```
> plot(pca.iris.cov$x[,c("PC3","PC4")],col = cols[iris$Species], pch = 19)
```



In R

```
> pca.iris.cov$rotation
                    PC1         PC2         PC3        PC4
Sepal.Length  0.36138659 -0.65658877  0.58202985  0.3154872
Sepal.Width  -0.08452251 -0.73016143 -0.59791083 -0.3197231
Petal.Length  0.85667061  0.17337266 -0.07623608 -0.4798390
Petal.Width   0.35828920  0.07548102 -0.54583143  0.7536574
```

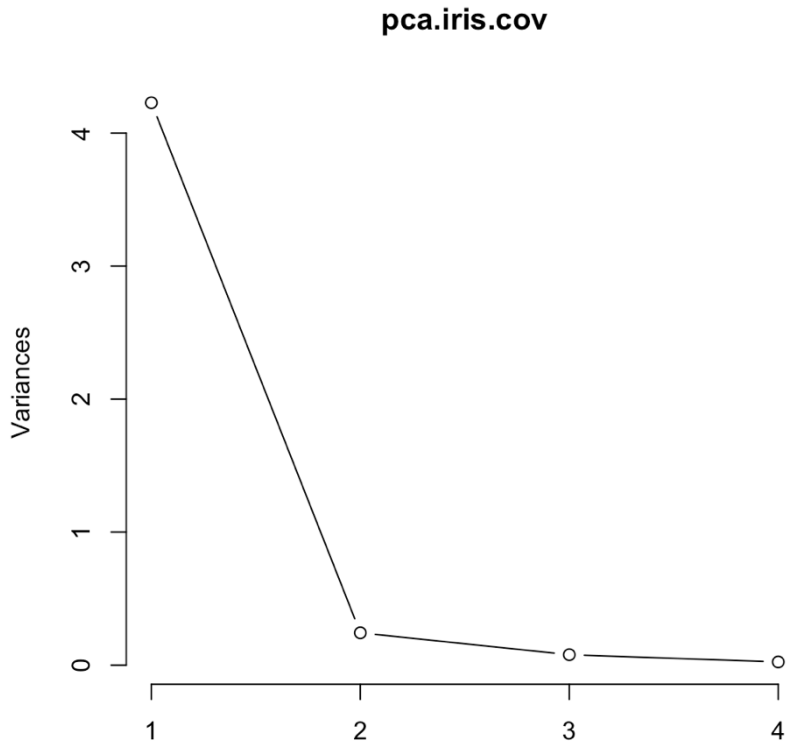
```
> biplot(pca.iris.cov, scale = 0)
```



In R

```
> summary(pca.iris.cov)
Importance of components:
                          PC1     PC2    PC3     PC4
Standard deviation     2.0563 0.49262 0.2797 0.15439
Proportion of Variance 0.9246 0.05307 0.0171 0.00521
Cumulative Proportion  0.9246 0.97769 0.9948 1.00000
```

```
> screeplot(pca.iris.cov, type = "line")
```

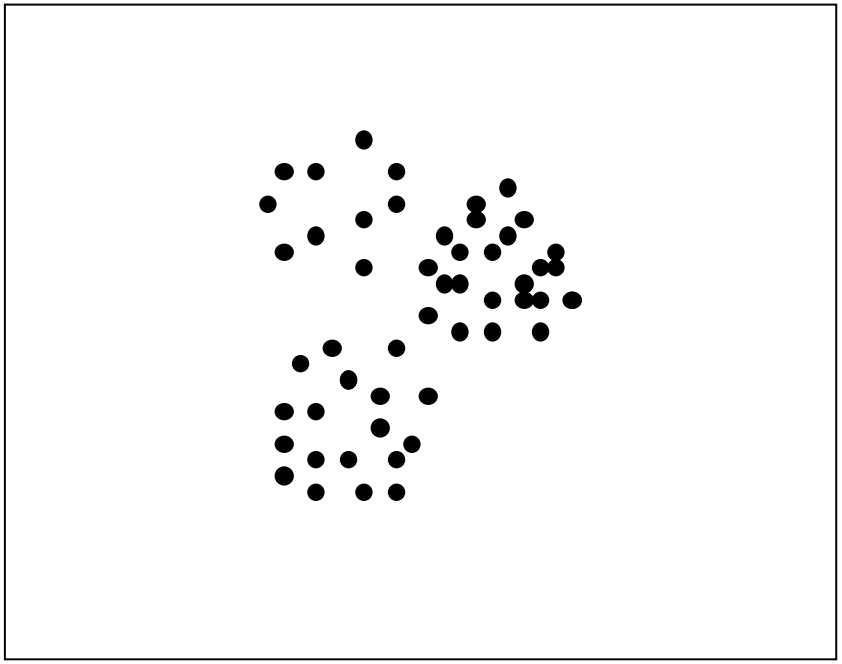


Exercise:

Rerun the Iris PCA with standardized data.
Did the results change?

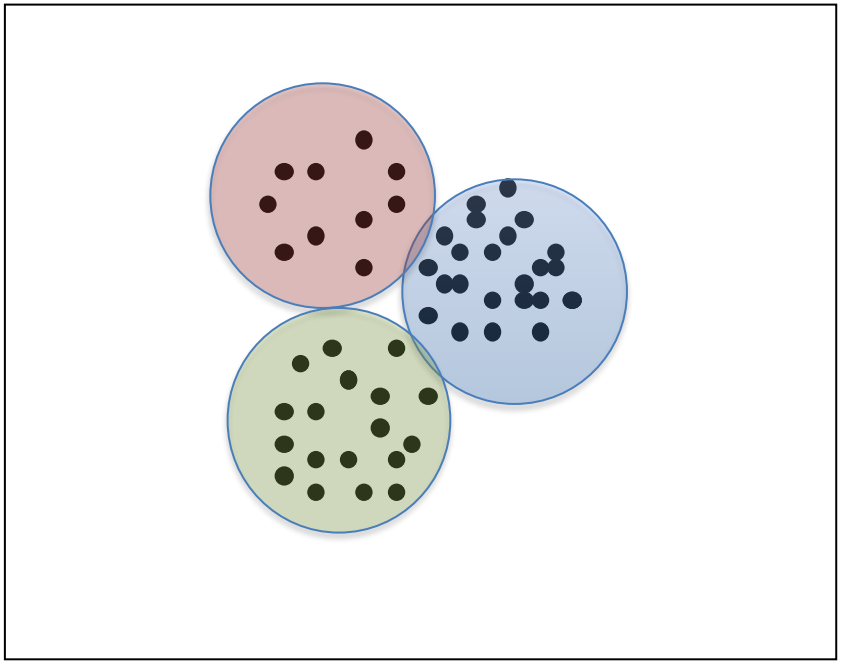
Clustering

Clustering



Point cloud

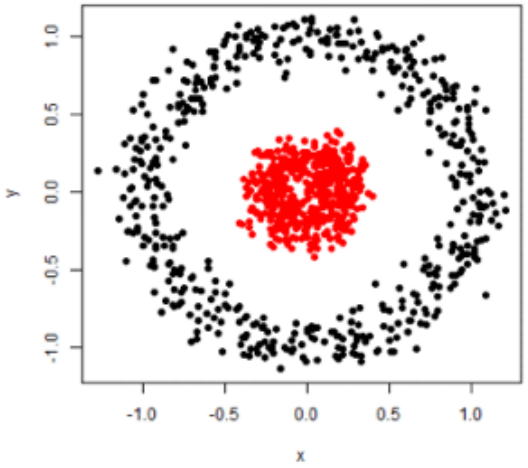
Clustering

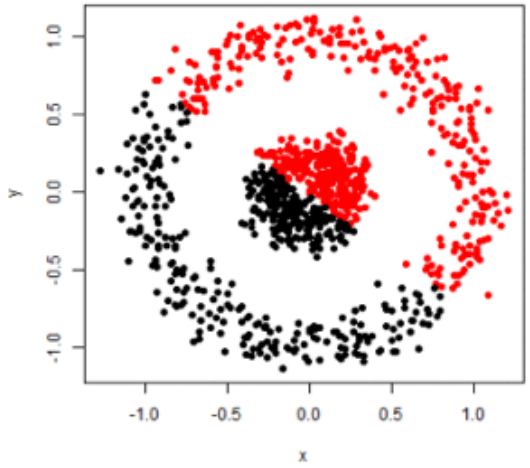


Clustering

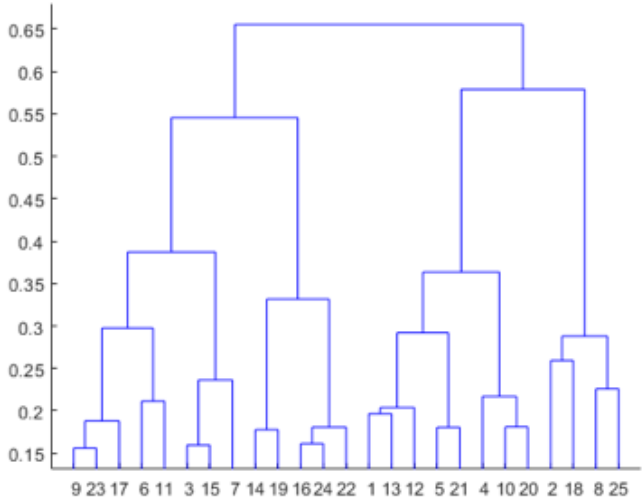
Clustering method are divided into two categories :

Partitioning clustering





Hierarchical clustering



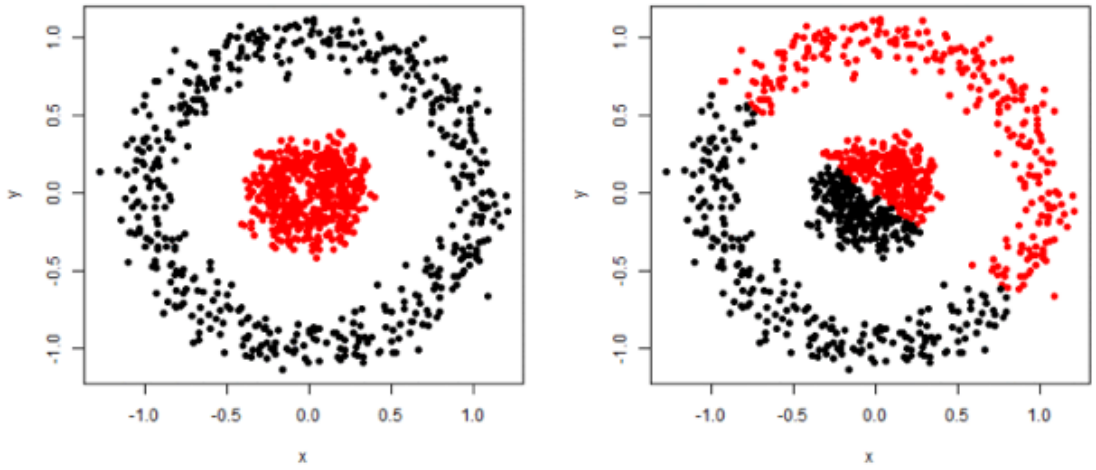
***Handbook of cluster analysis, Hennig C. et al.**

Partitioning clustering

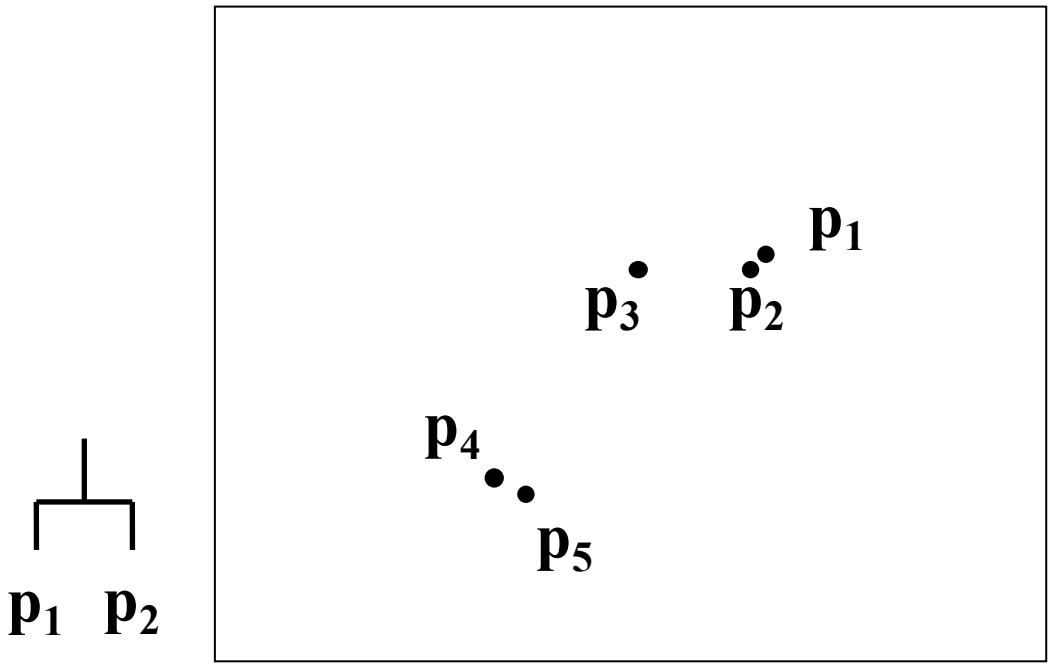
Convex partitioning. Example: K-means

Density based approaches. Example: DBSCAN

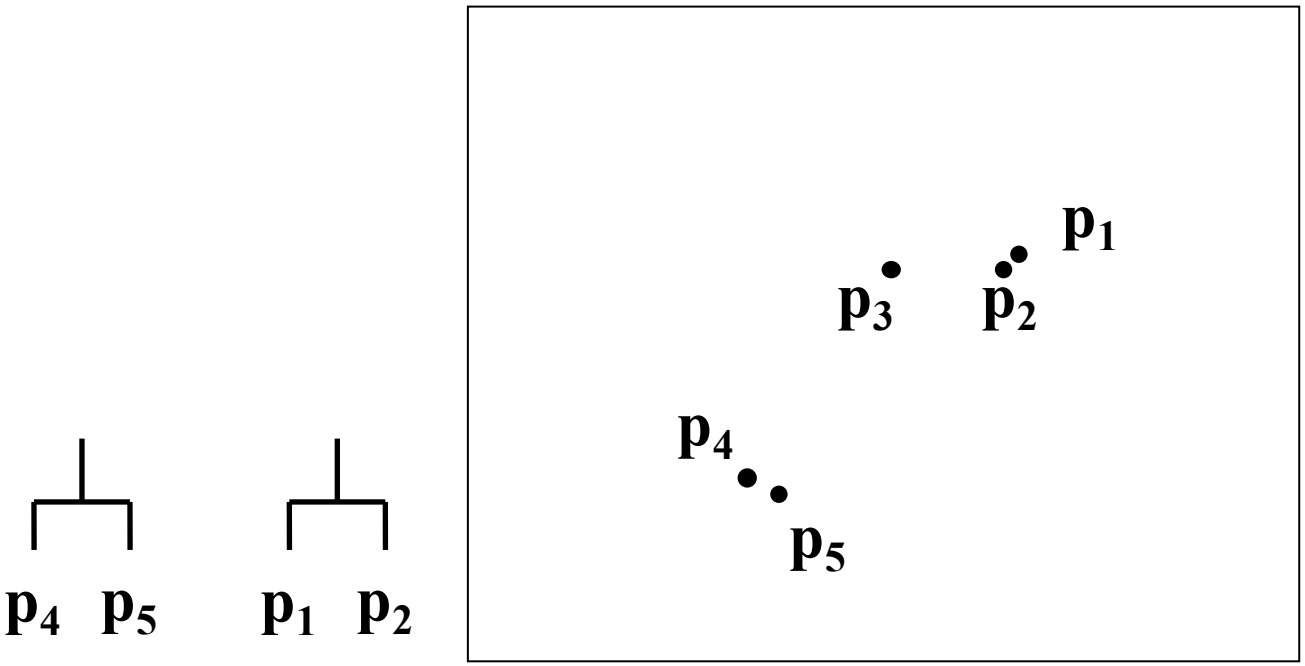
Model-based approaches. Example: Mclust



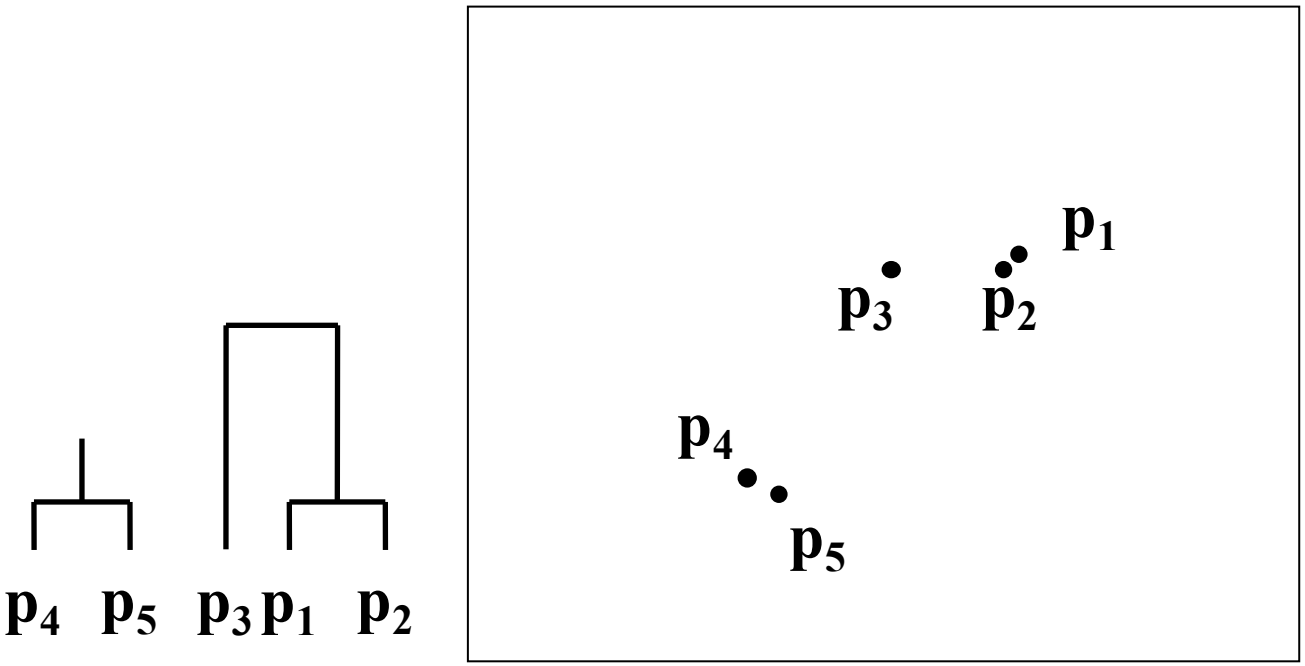
Hierarchical Clustering



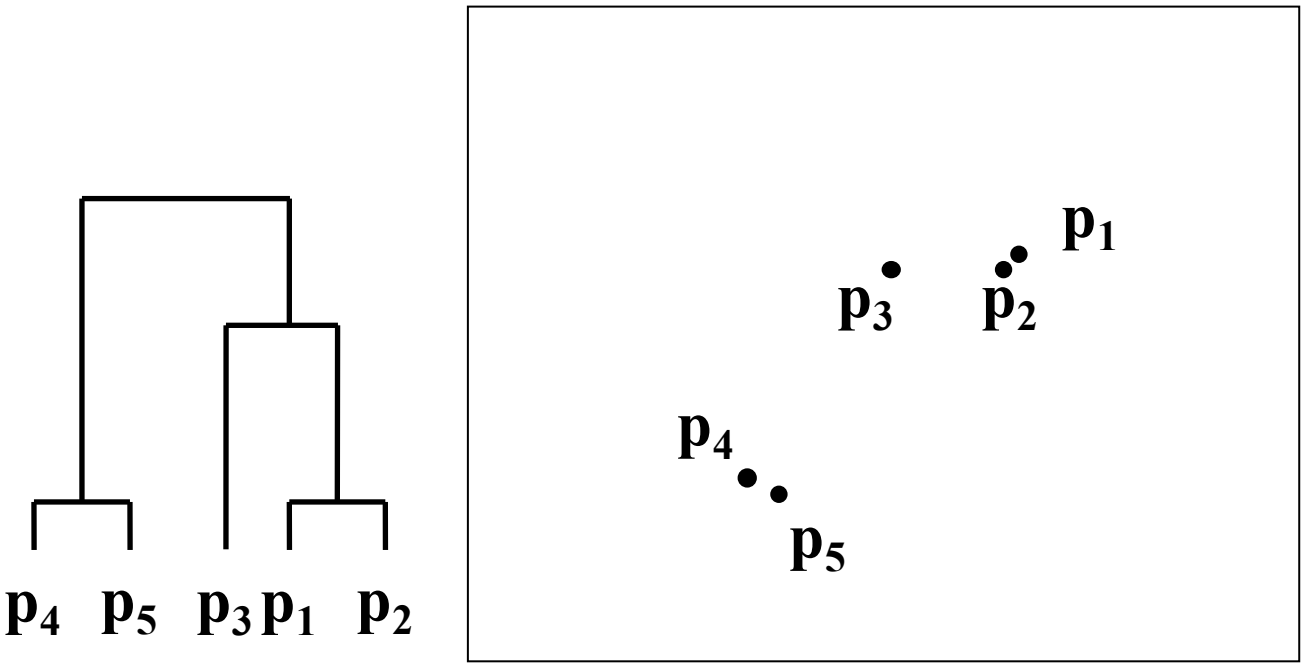
Hierarchical Clustering



Hierarchical Clustering



Hierarchical Clustering



Distance

Euclidean

X = (2, 0)

Y = (-2, -2)

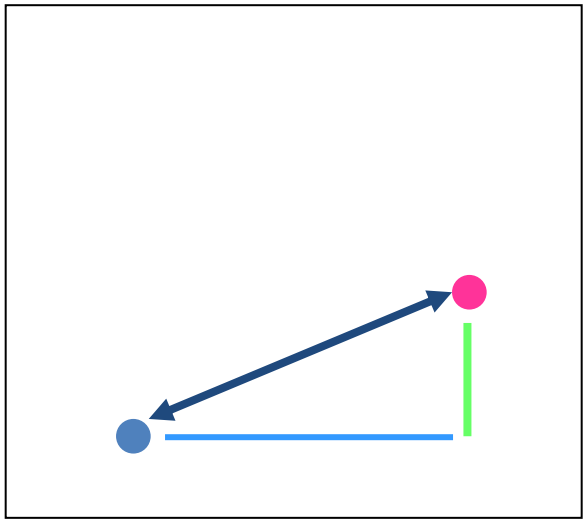
$\sqrt{\ [\Sigma\ (y - x)^2\ ]}$

$= \sqrt{([-2 - 2]^2 + [-2 - 0]^2)}$

$= \sqrt{(4^2 + 2^2)}$

$= \sqrt{20}$

$= 4.47$



It represents the “multivariate dissimilarity” of X & Y

Squared Euclidean

$X = (2, 0)$

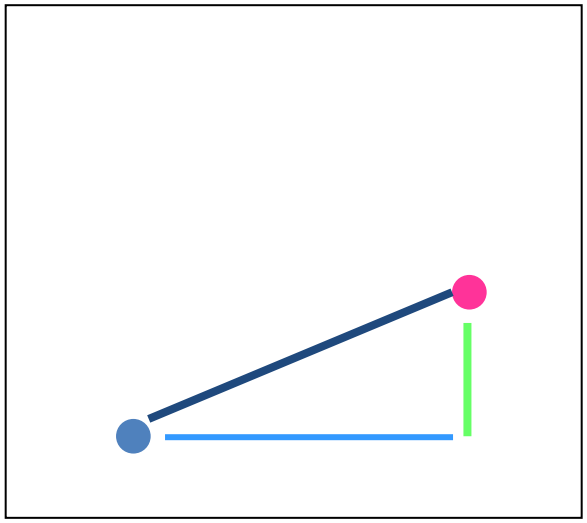
$Y = (-2, -2)$

$\Sigma (y - x)^2$

$= ([-2 - 2]^2 + [-2 - 0]^2)$

$= (4^2 + 2^2)$

$= 20$



It represents the “multivariate dissimilarity” of X & Y

City Block (Manhattan)

X = (2, 0)

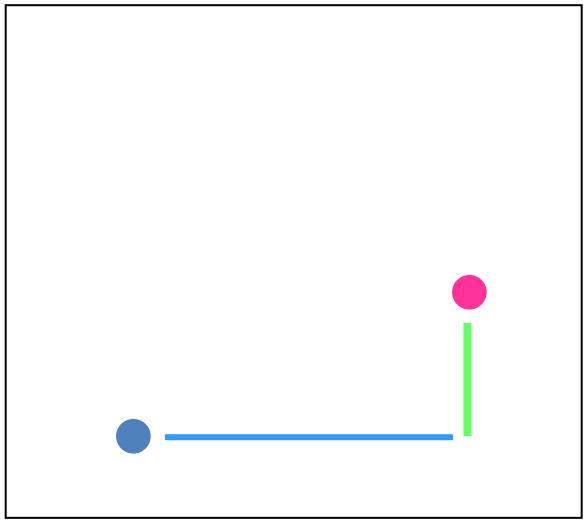
Y = (-2, -2)

$\Sigma |y - x|$

$= (|-2 - 2| + |-2 - 0|)$

$= |-4| + |-2|$

$= 6$



Distance Measures in 2D

- Euclidean $\sqrt{[\Sigma (y - x)^2]}$
- Squared Euclidean $\Sigma (y - x)^2$
- City-Block $\Sigma | y - x |$

Distance Measures in nD

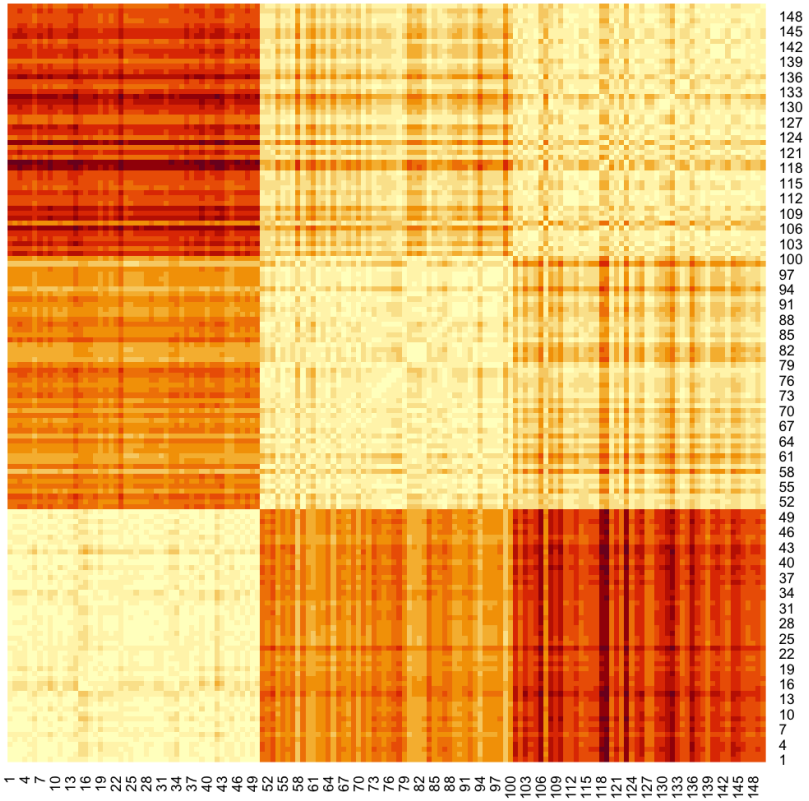
- Euclidean $$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$
- Squared Euclidean $$d = (a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2$$
- City-Block $$d = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n|$$

In R

```
>?dist
```

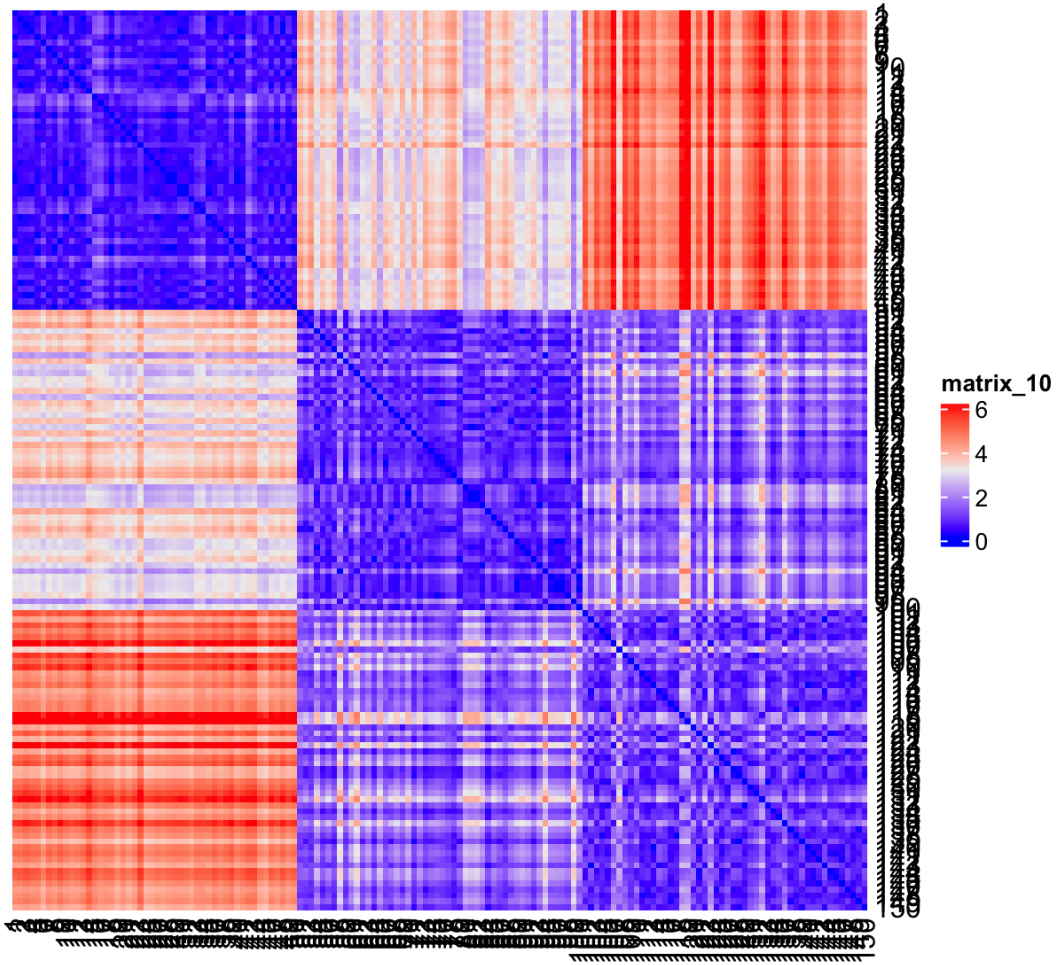
In R

```
> ?dist
> distanceMatrix <- as.matrix(dist(iris[, 1:4],
+                                  method = "euclidean", upper = TRUE, diag = TRUE))
> heatmap(distanceMatrix,Rowv = NA, Colv = NA, scale="none")
```



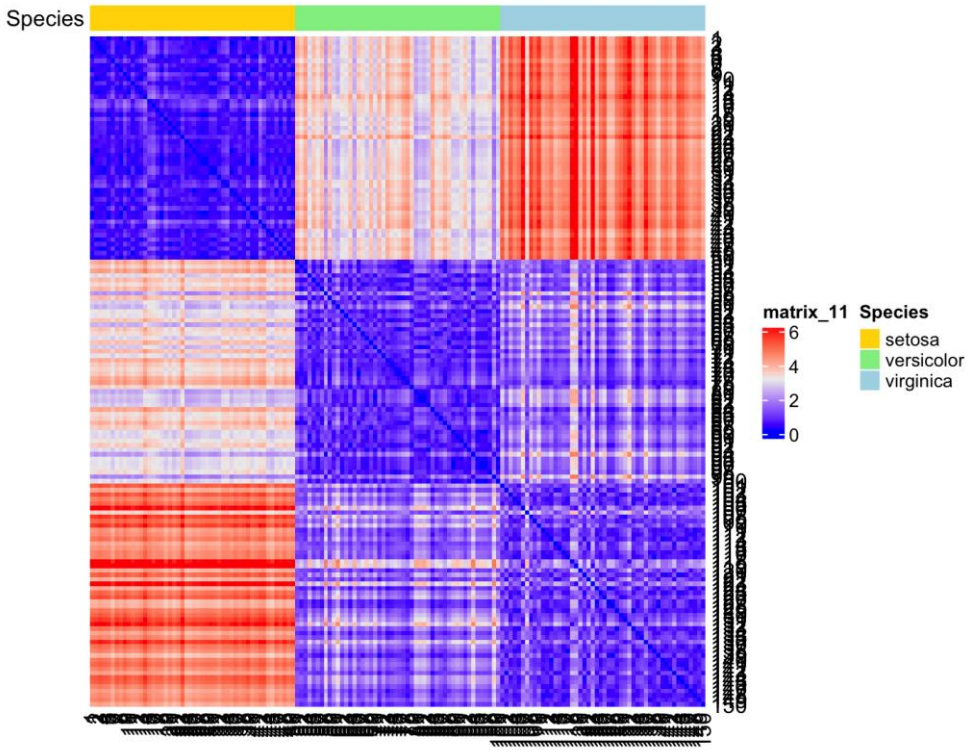
In R

```
> #BiocManager::install("ComplexHeatmap")
> library(ComplexHeatmap)
> Heatmap(distanceMatrix, cluster_rows = F, cluster_columns = F)
```

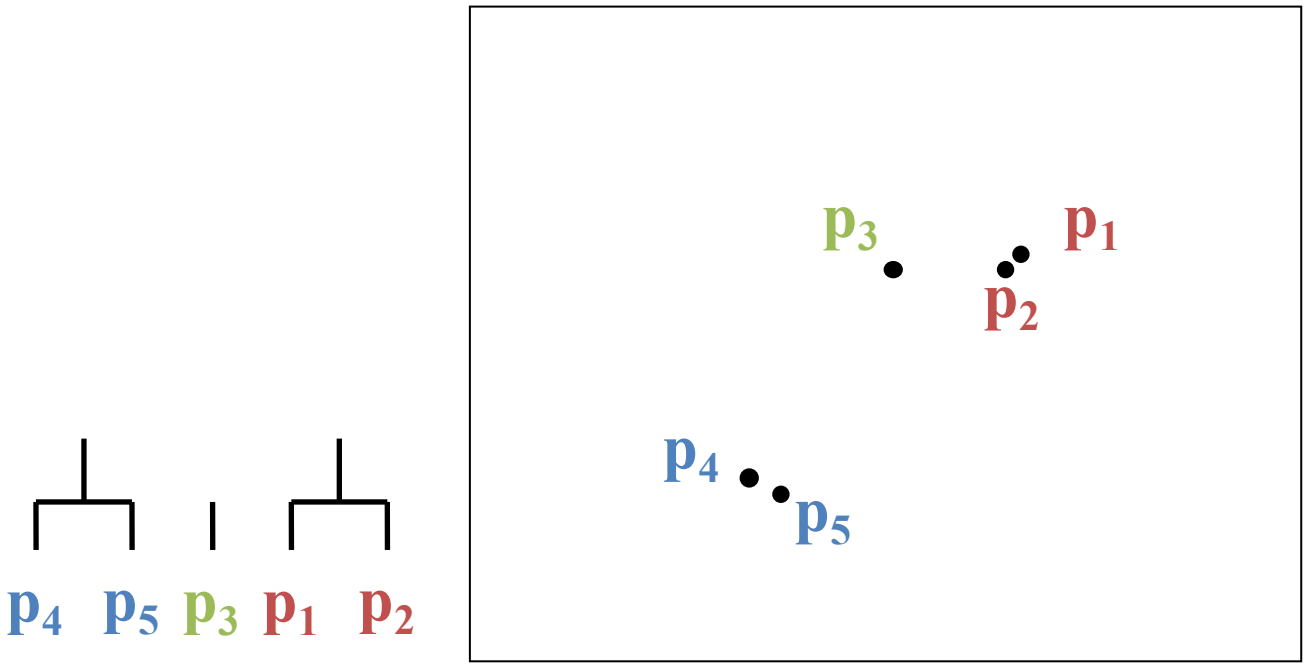


In R

```
> species_annot <- HeatmapAnnotation(Species = iris$Species,
+                                    col = list(Species = c(
+                                      setosa = "gold",
+                                      versicolor = "lightgreen",
+                                      virginica = "lightblue")),
+                                    annotation_name_side = "left")
> Heatmap(distanceMatrix,cluster_rows = F, cluster_columns = F,top_annotation = species_annot)
```

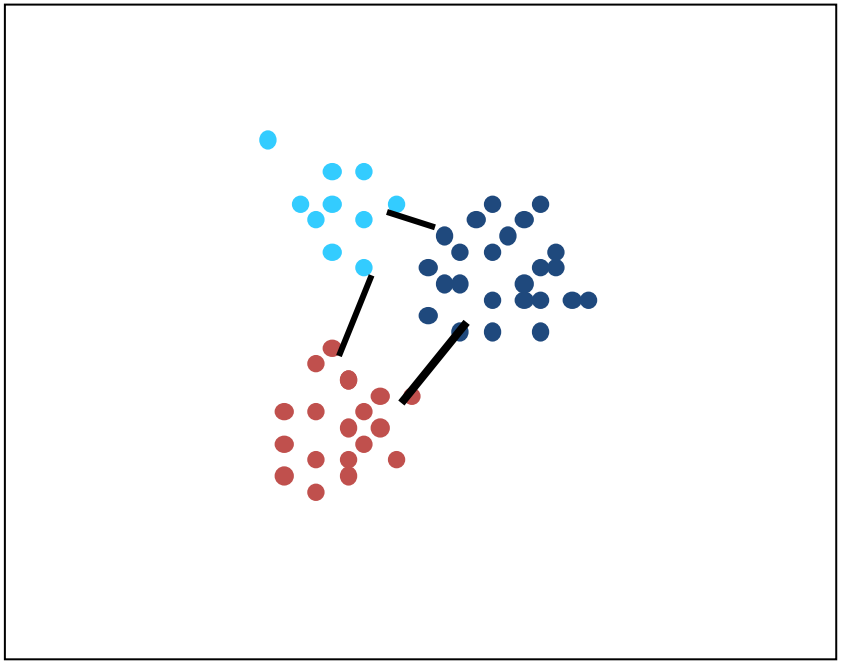


How to aggregate clusters?


p3
p1
p2
p4
p5
p4 p5 p3 p1 p2

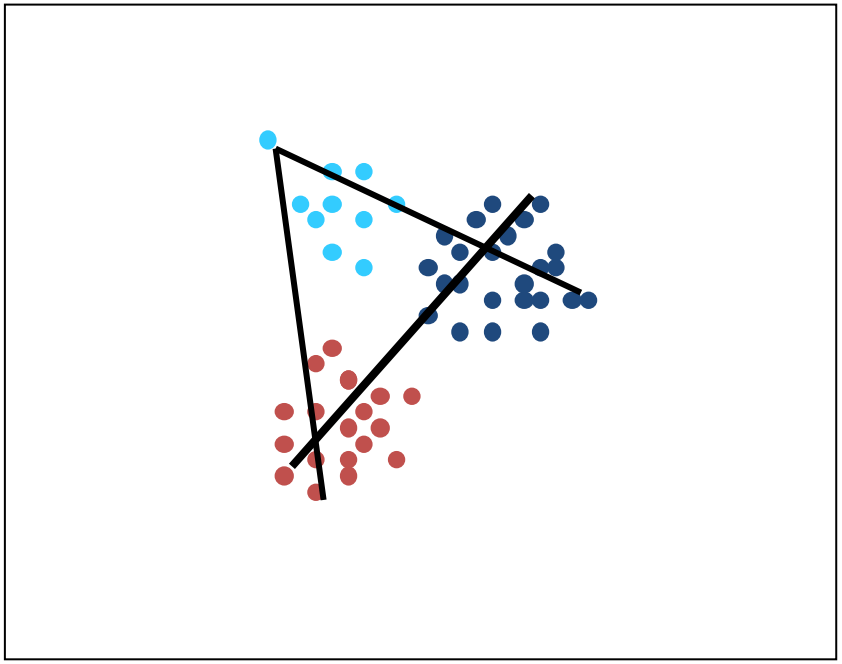
Which clusters to combine?

Single linkage



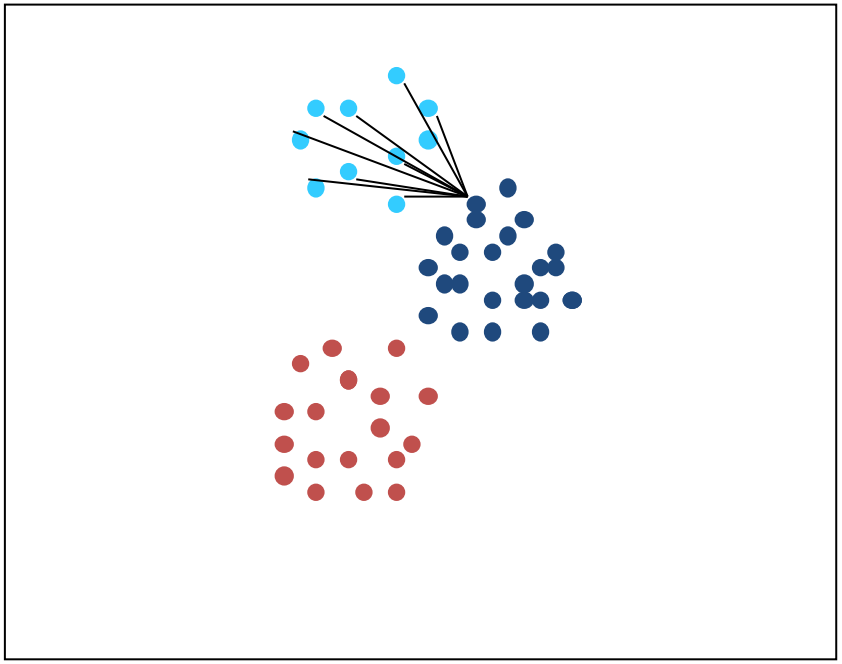
Distance between closest elements in clusters

Complete Linkage



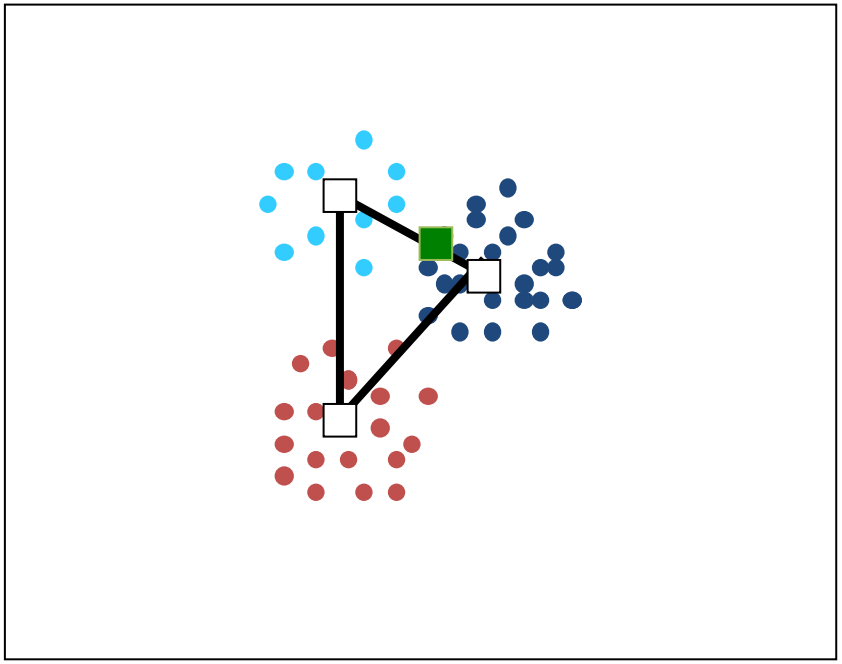
Distance between furthest elements in clusters

Average Linkage



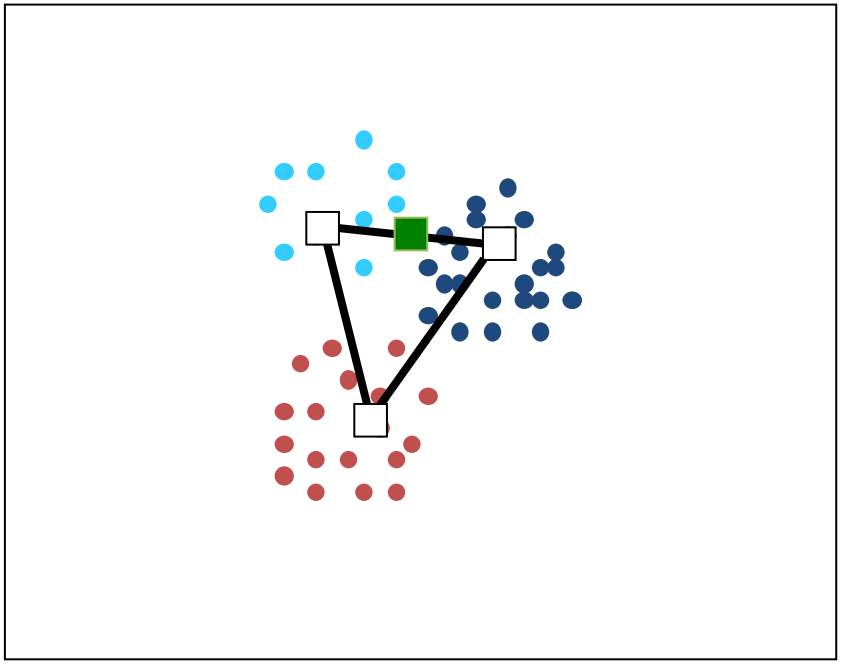
Average of all pairwise distances

Centroid Condensation (mean)



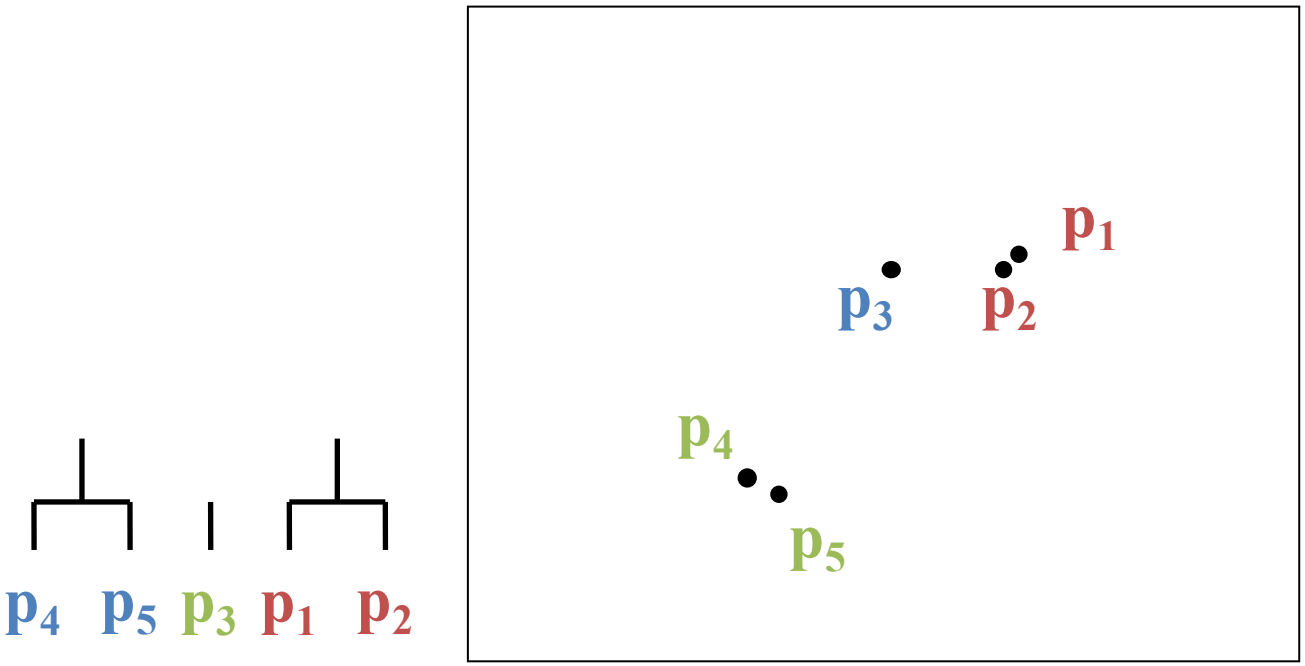
Distance between centroids (means) of two clusters

Median Condensation



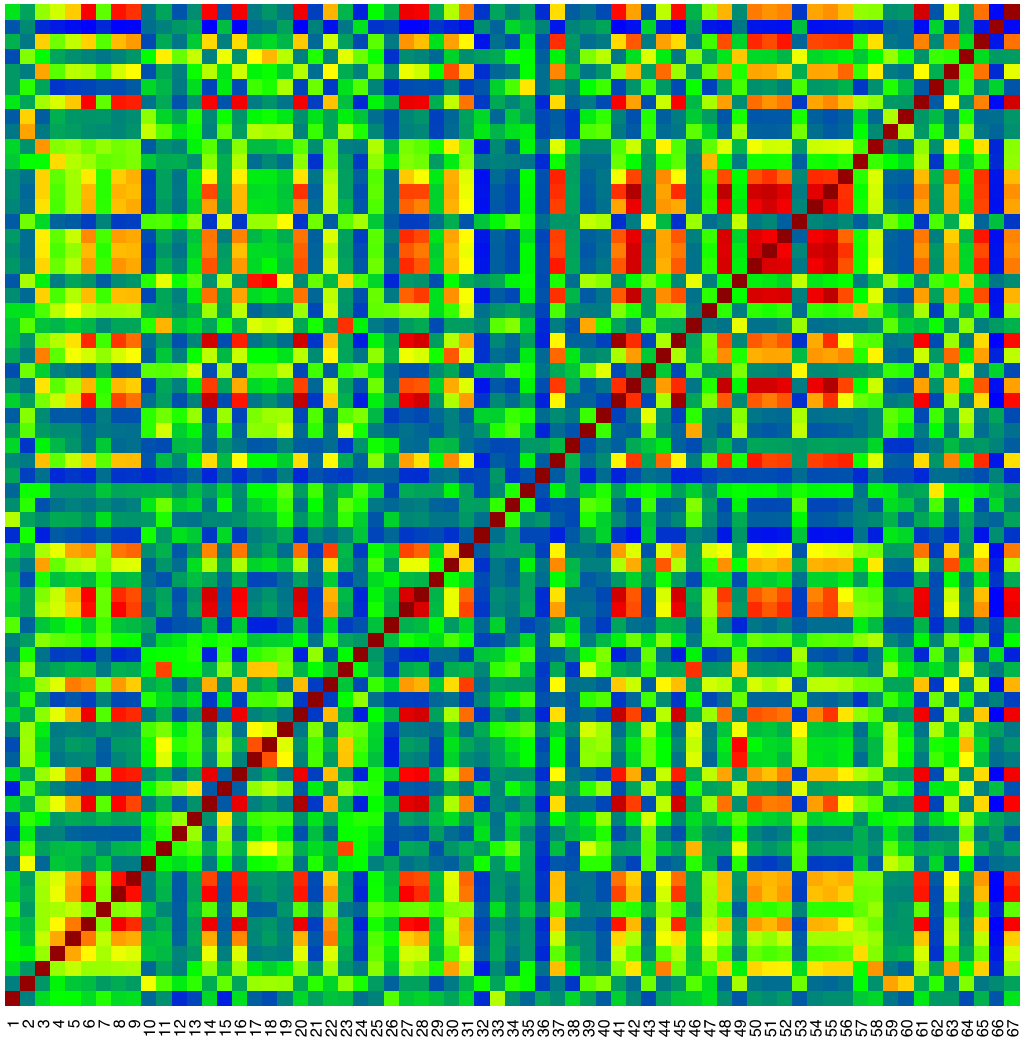
Distance between median distances of two clusters

Hierarchical Clustering



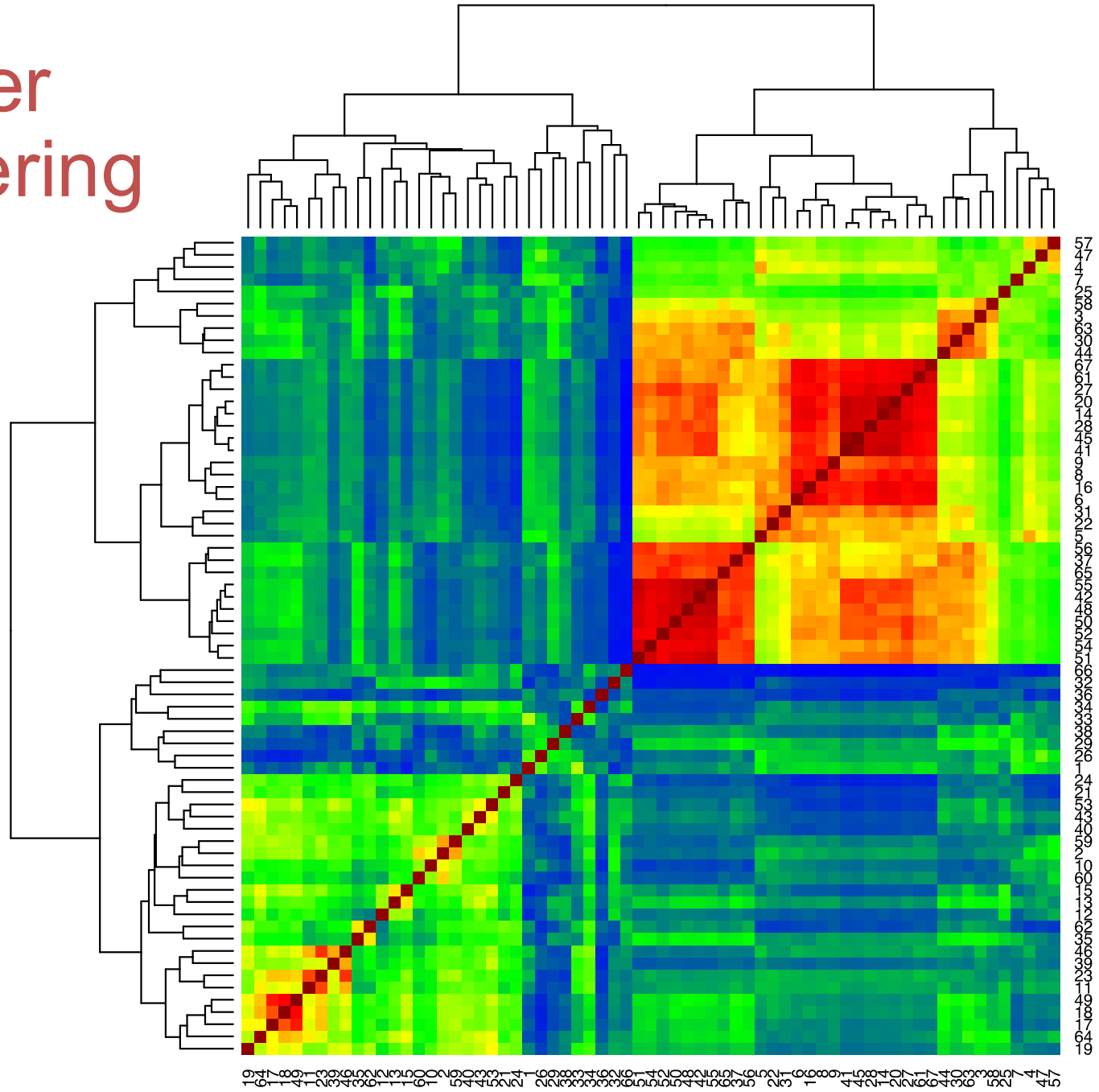
At the beginning every point is a cluster in it self, then we agglomerate …

Before Clustering

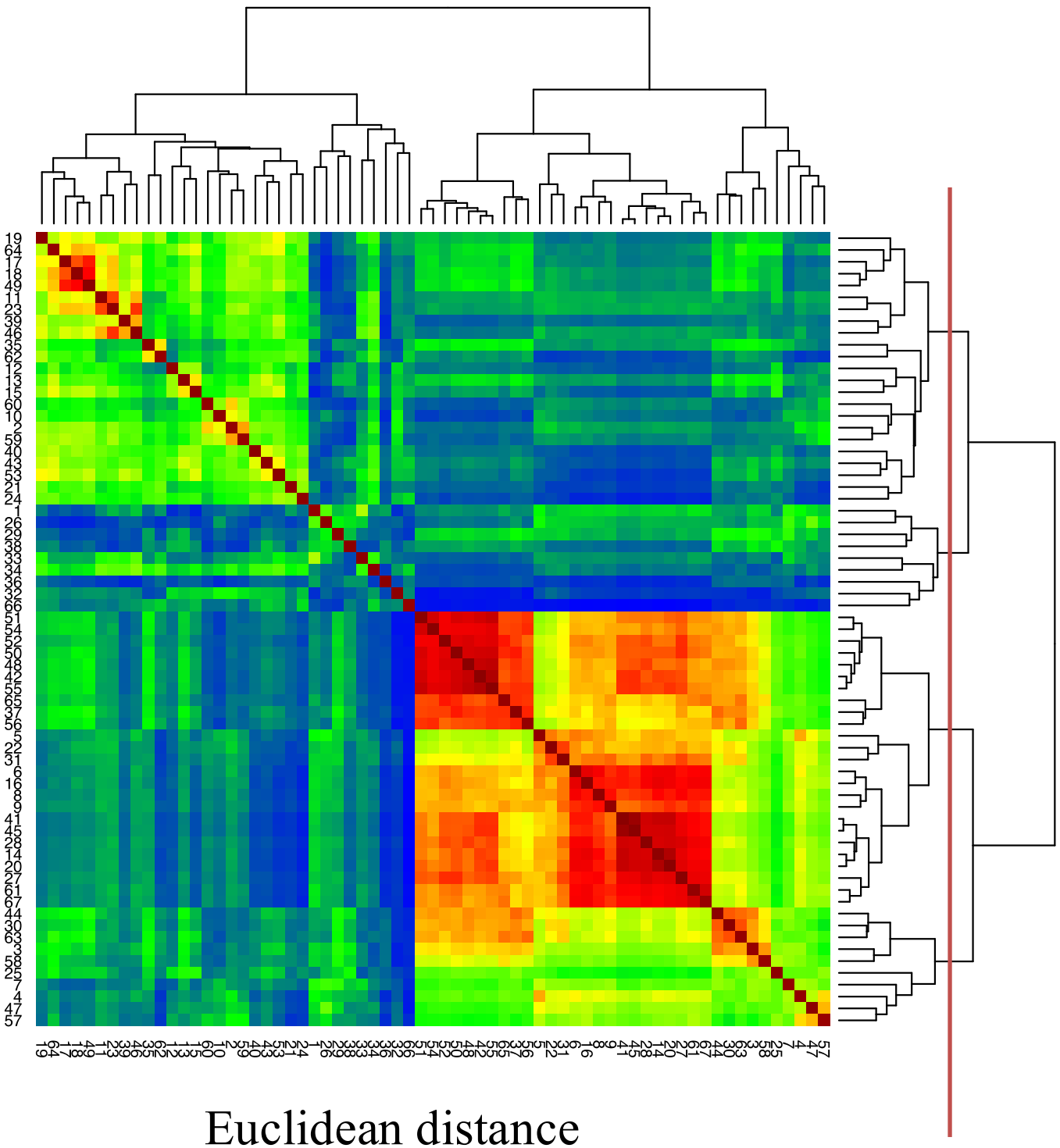


Euclidean distance

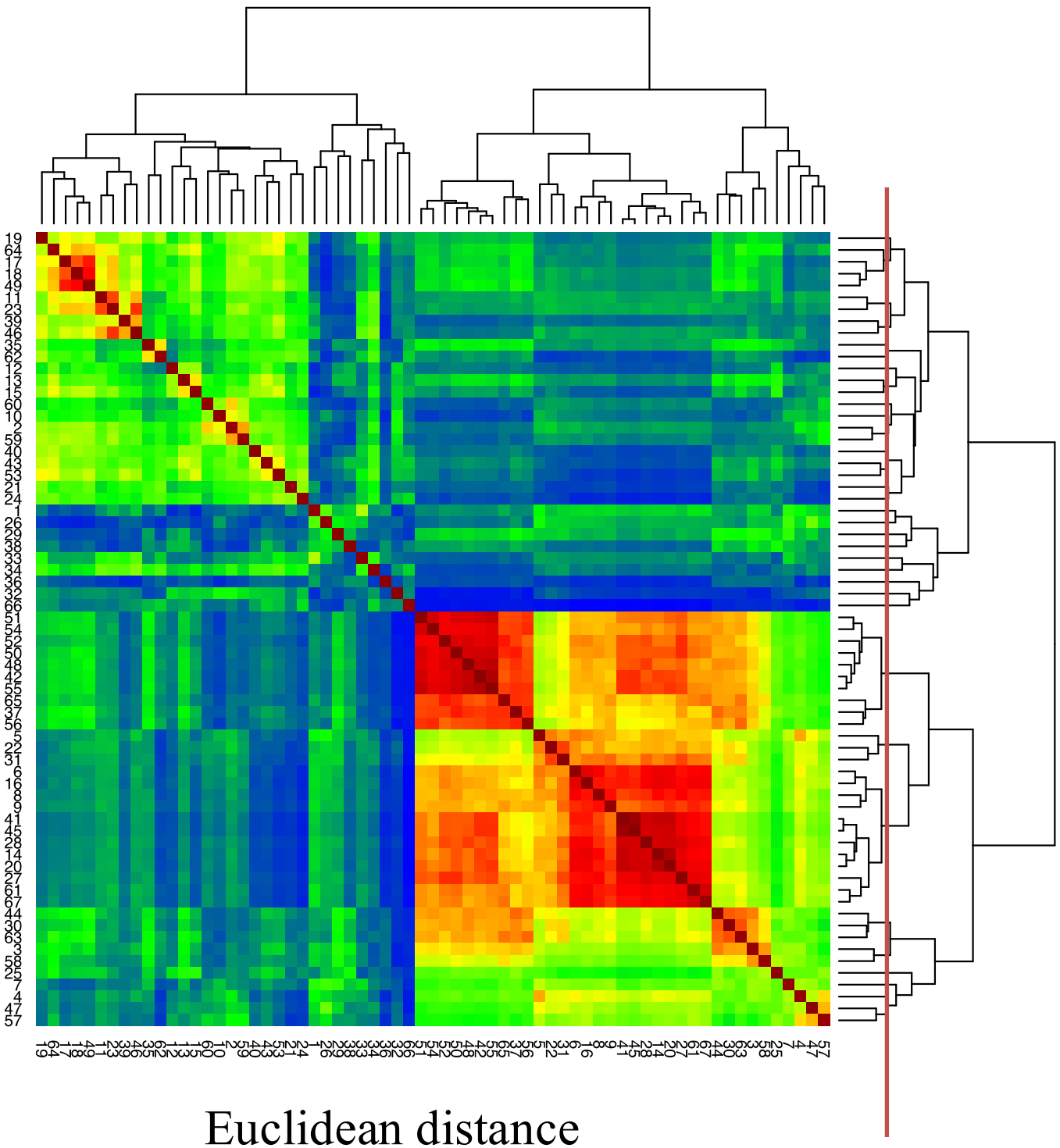
After Clustering



Determine the Termination Condition (TC)



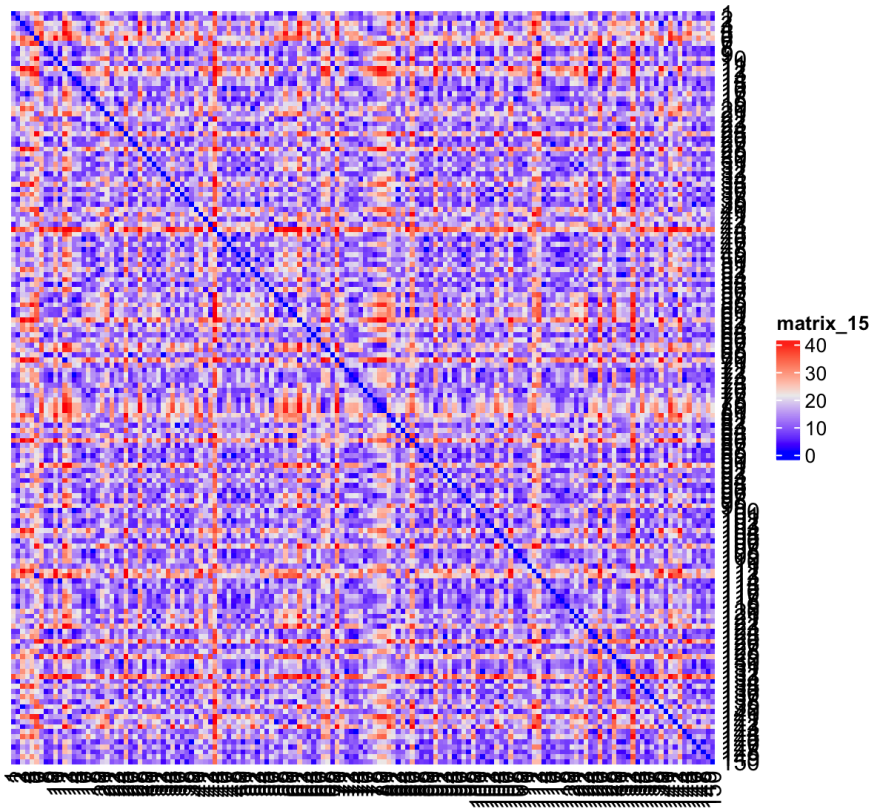
Euclidean distance
complete Linkage



Euclidean distance
complete Linkage

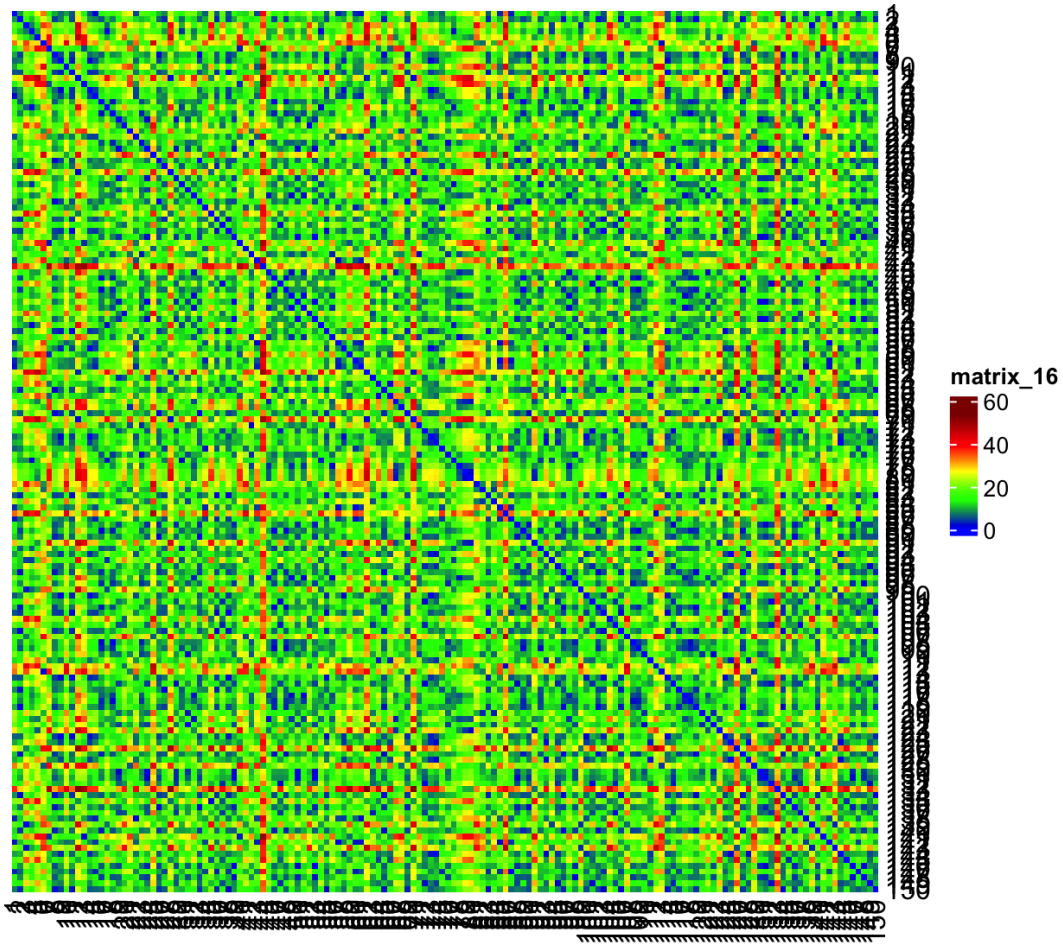
Let's practice In R

```r
#create a random matrix
mat <- matrix(data = rnorm(300, mean= 100, sd=10), nrow = 150, ncol = 2)

#Euclidian distance
mat.dist<-as.matrix(dist(mat))

#show heatmap
Heatmap(mat.dist,cluster_rows = F, cluster_columns = F)
```



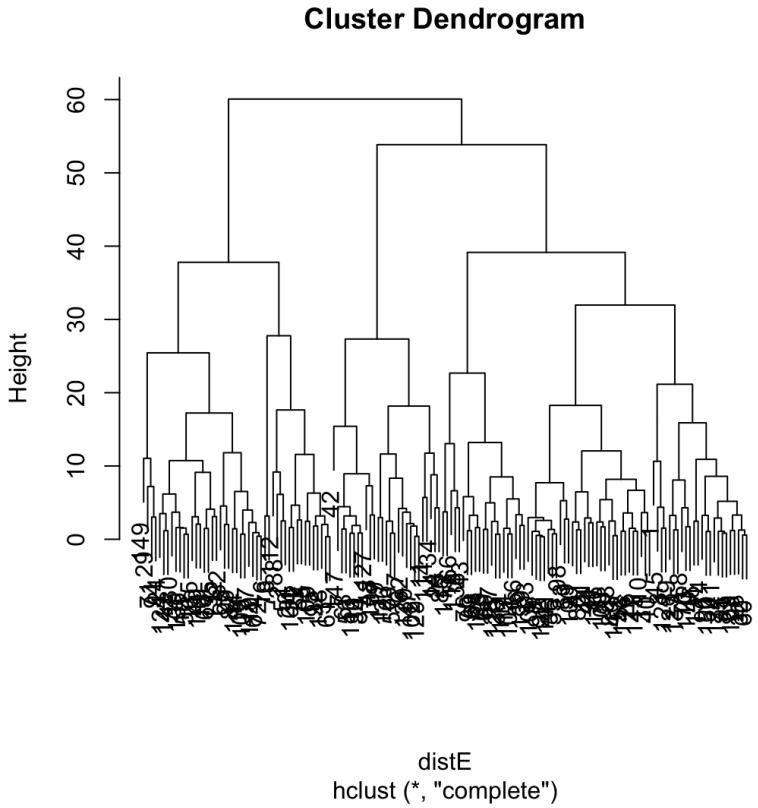
Let’s practice In R

```
#change heatmap’s color
colorScale <- colorRampPalette(c("blue", "green","yellow","red","darkred"))(1000)
Heatmap(mat.dist,cluster_rows = F, cluster_columns = F, col = colorScale)
```



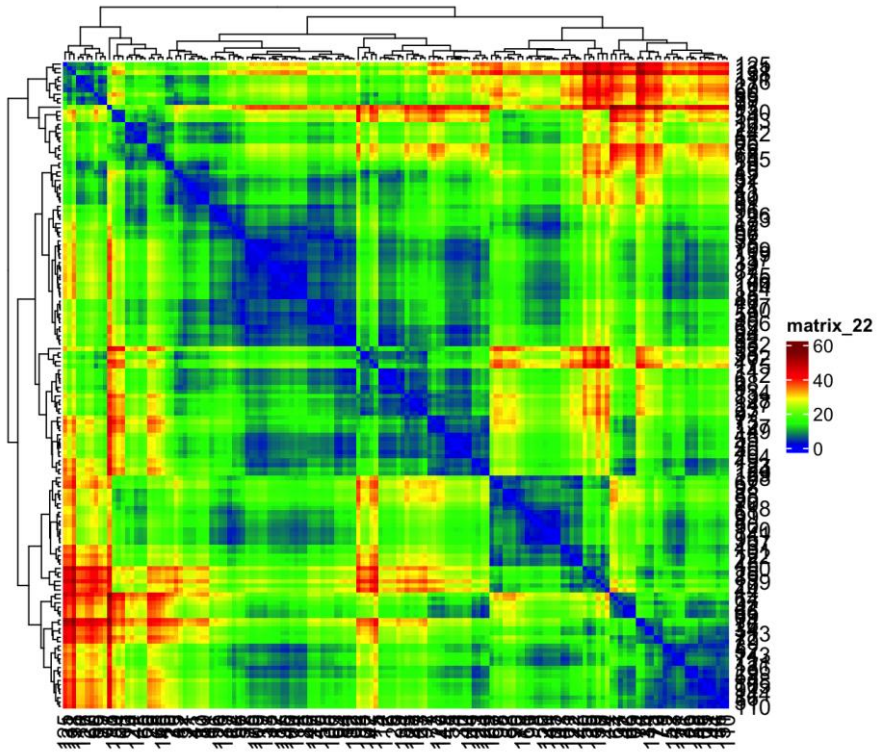
How to do hierarchical clustering in R?

```
?hclust

#Euclidian distance
distE <- dist(mat)

hE <- hclust(distE,method = "complete")
plot(hE)
```



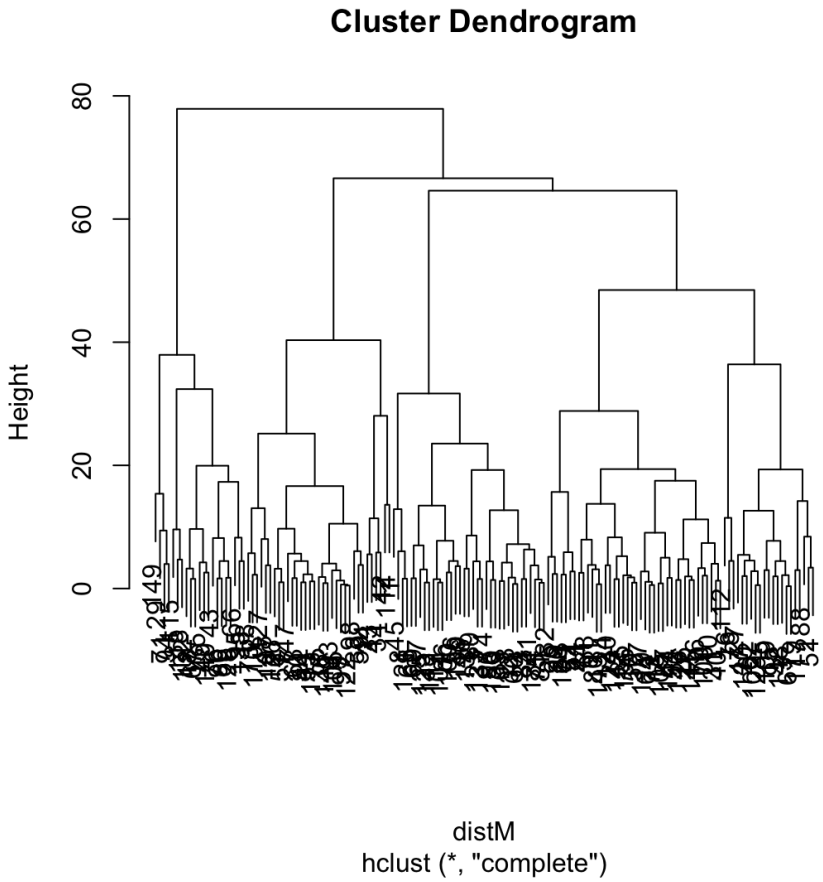
How to do hierarchical clustering in R?

```
?hclust

#Euclidian distance
distE <- dist(mat)

hE <- hclust(distE,method = "complete")
plot(hE)


Heatmap(
  mat.dist,
  cluster_rows = hE,
  cluster_columns = hE,
  col = colorScale
)
```



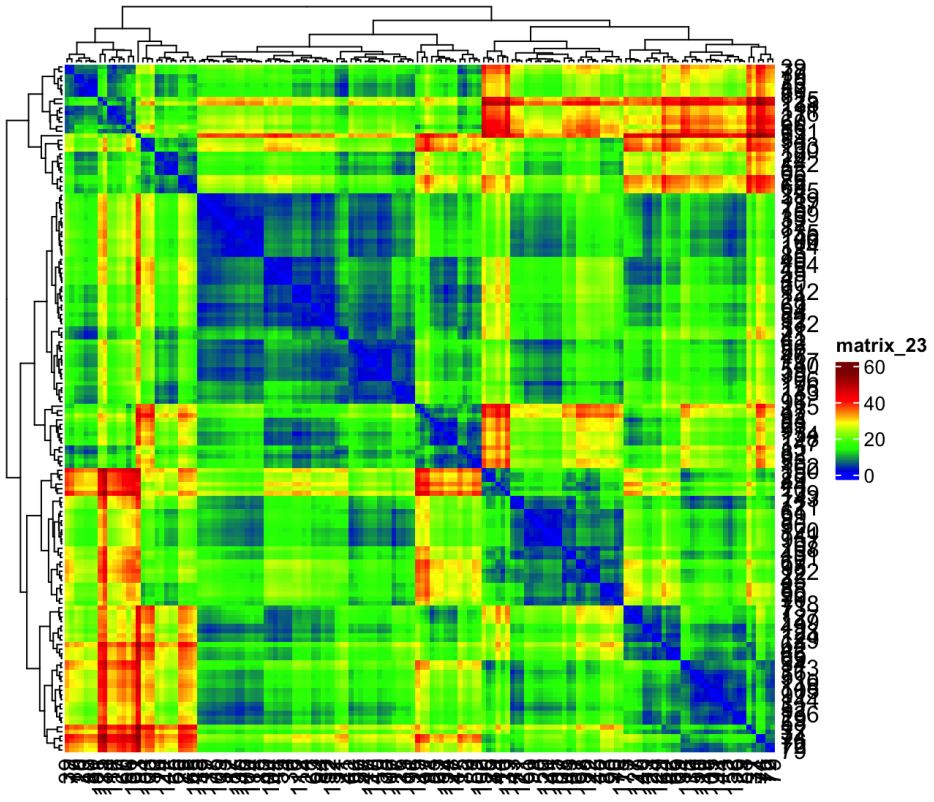
How to do hierarchical clustering in R?

```
#manhattan distance
distM<-dist(mat,method="manhattan")

hM <- hclust(distM,"complete")
plot(hM)
```



How to do hierarchical clustering in R?

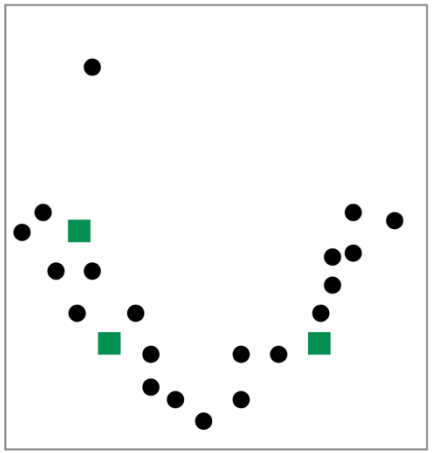
```
#manhattan distance
distM<-dist(mat,method="manhattan")

hM <- hclust(distM,"complete")
plot(hM)


Heatmap(
  mat.dist,
  cluster_rows = hM,
  cluster_columns = hM,
  col = colorScale
)
```



K-means Clustering

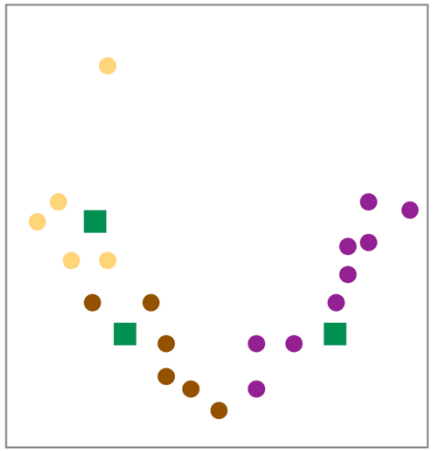
Number of clusters = 3

a



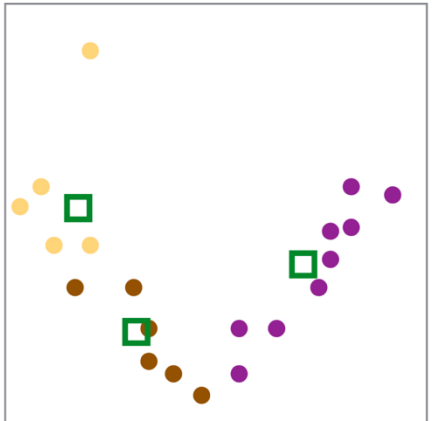
Start with 3 initial points

b



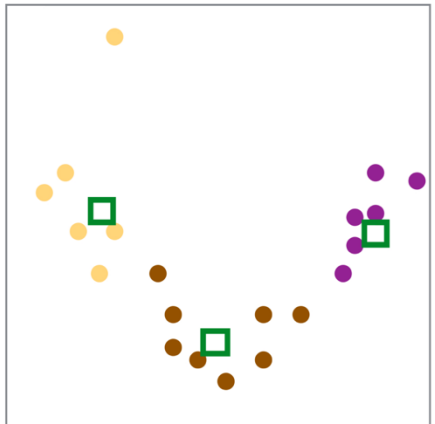
For each point determine to which initial point it is the closest

c



Move initial points to the centroids of the clusters

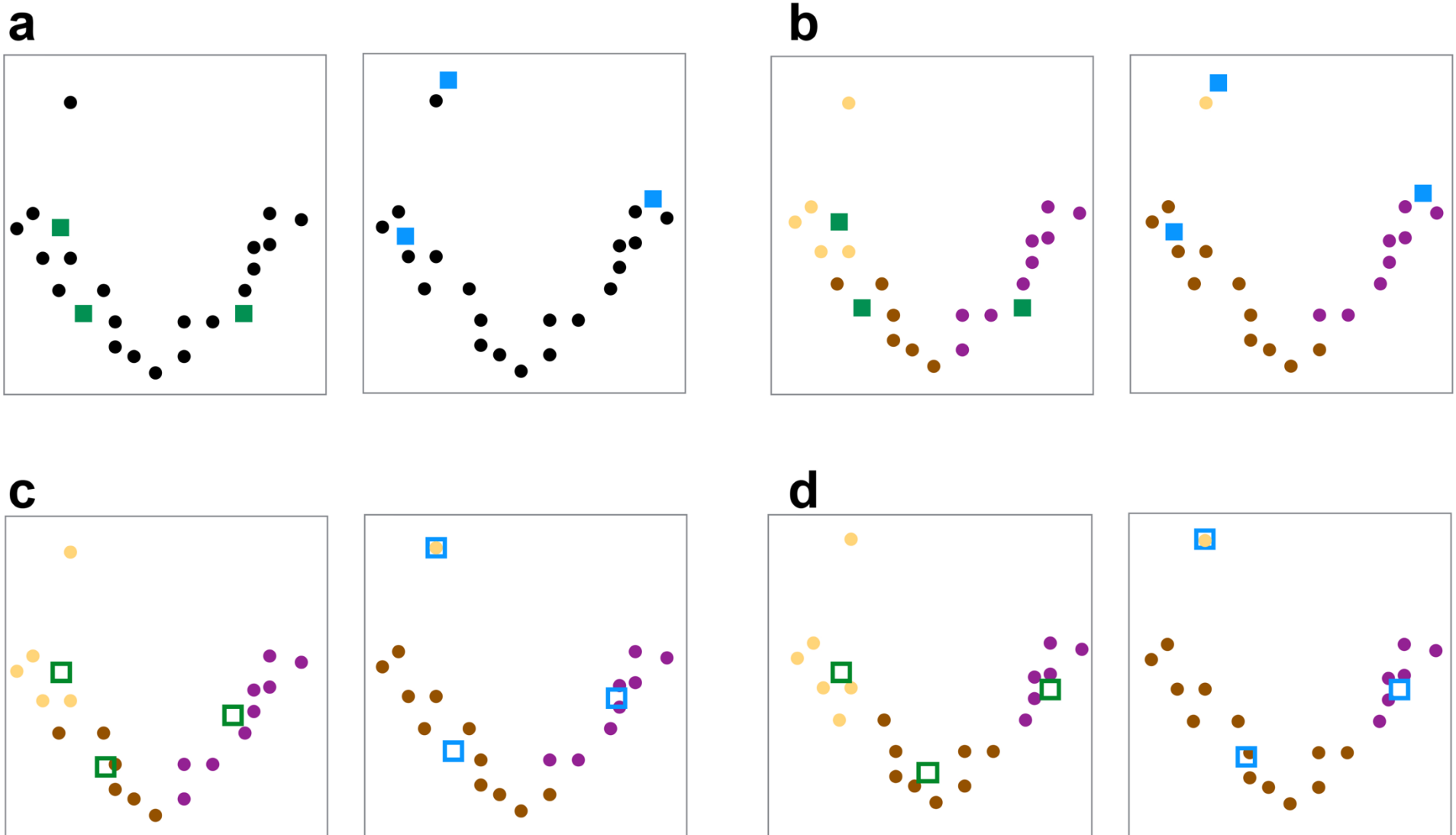
d



Color again each point! Repeat b and c until obtaining stabilisation

K-means Clustering

Number of clusters = 3



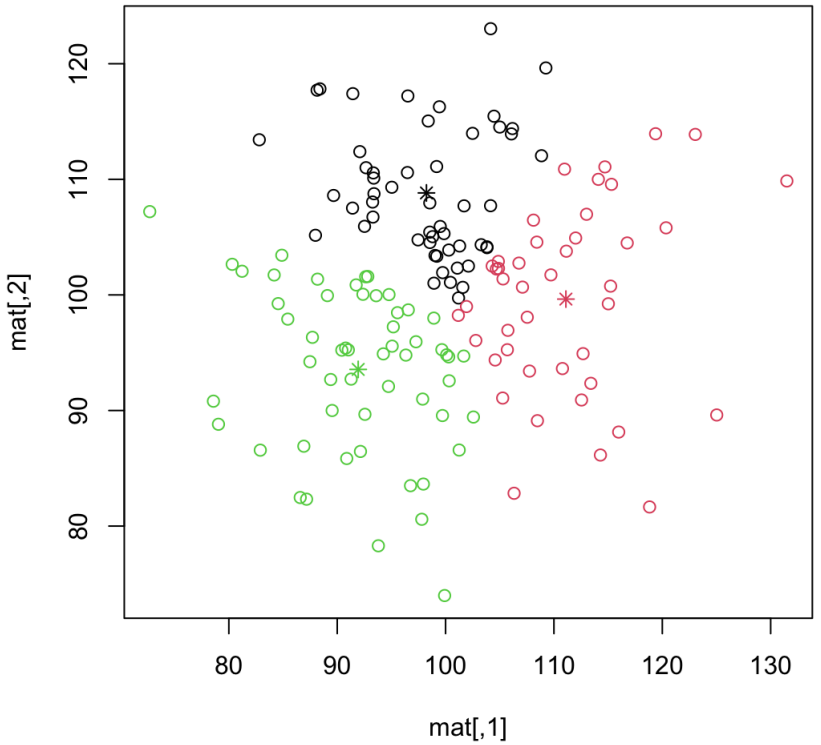
K-means

Drawbacks:

1. Specify number of clusters
2. Non probabilistic methods
3. Not stable

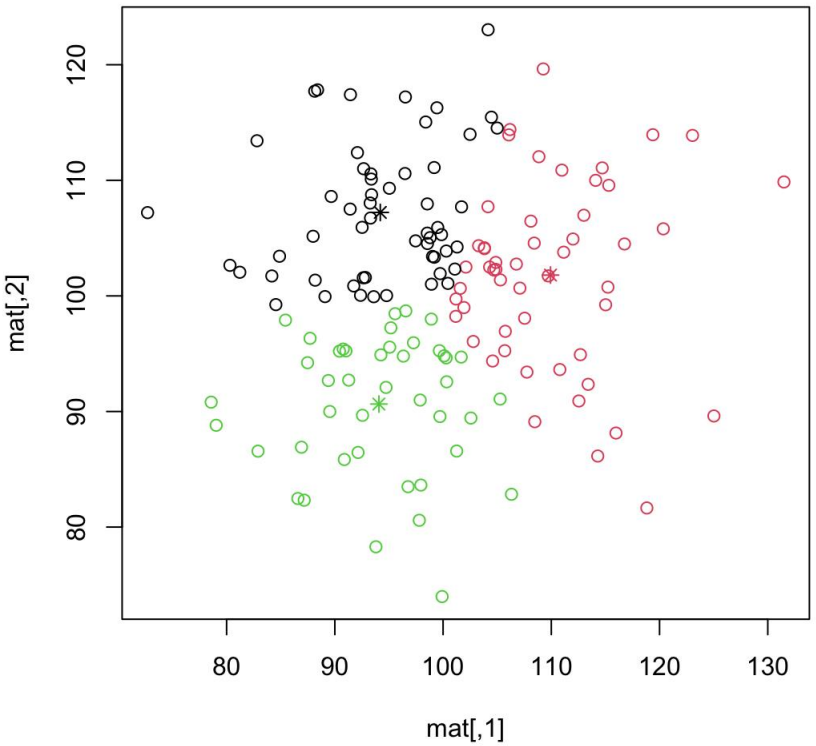
Kmeans in R

```
?kmeans

# k=3, 1 iteration
cl.1 <- kmeans(mat, 3, iter.max = 1)

# plot the samples and color by cluster
plot(mat, col = cl.1$cluster)

# add centroids (stars)
points(cl.1$centers, col = 1:5, pch = 8)
```



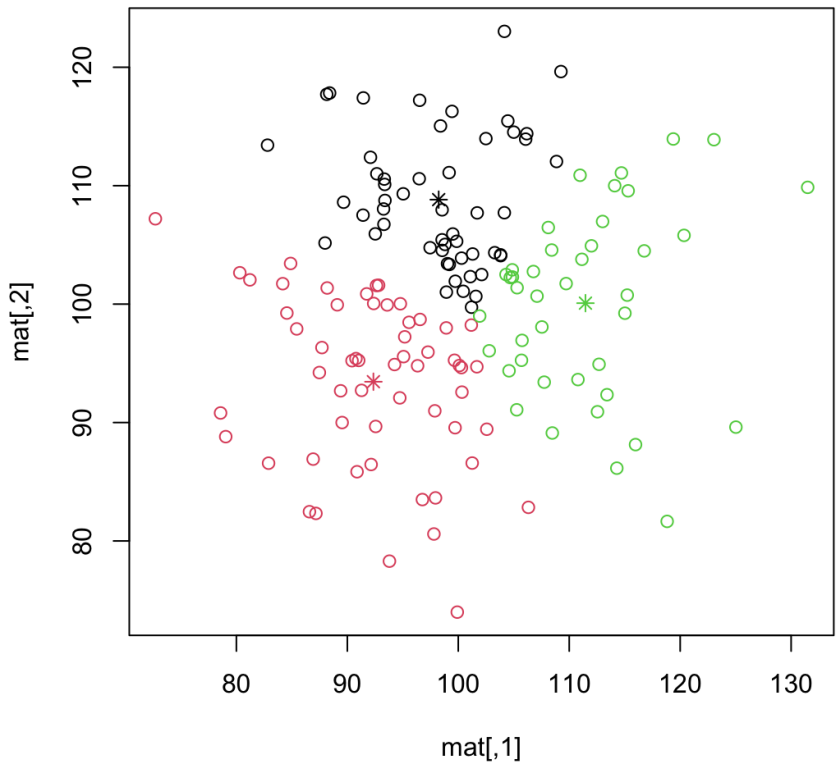
Kmeans in R

```
# k=3, 10 iterations
cl.10 <- kmeans(mat, 3, iter.max = 10)
plot(mat, col = cl.10$cluster)
points(cl.10$centers, col = 1:5, pch = 8)
```

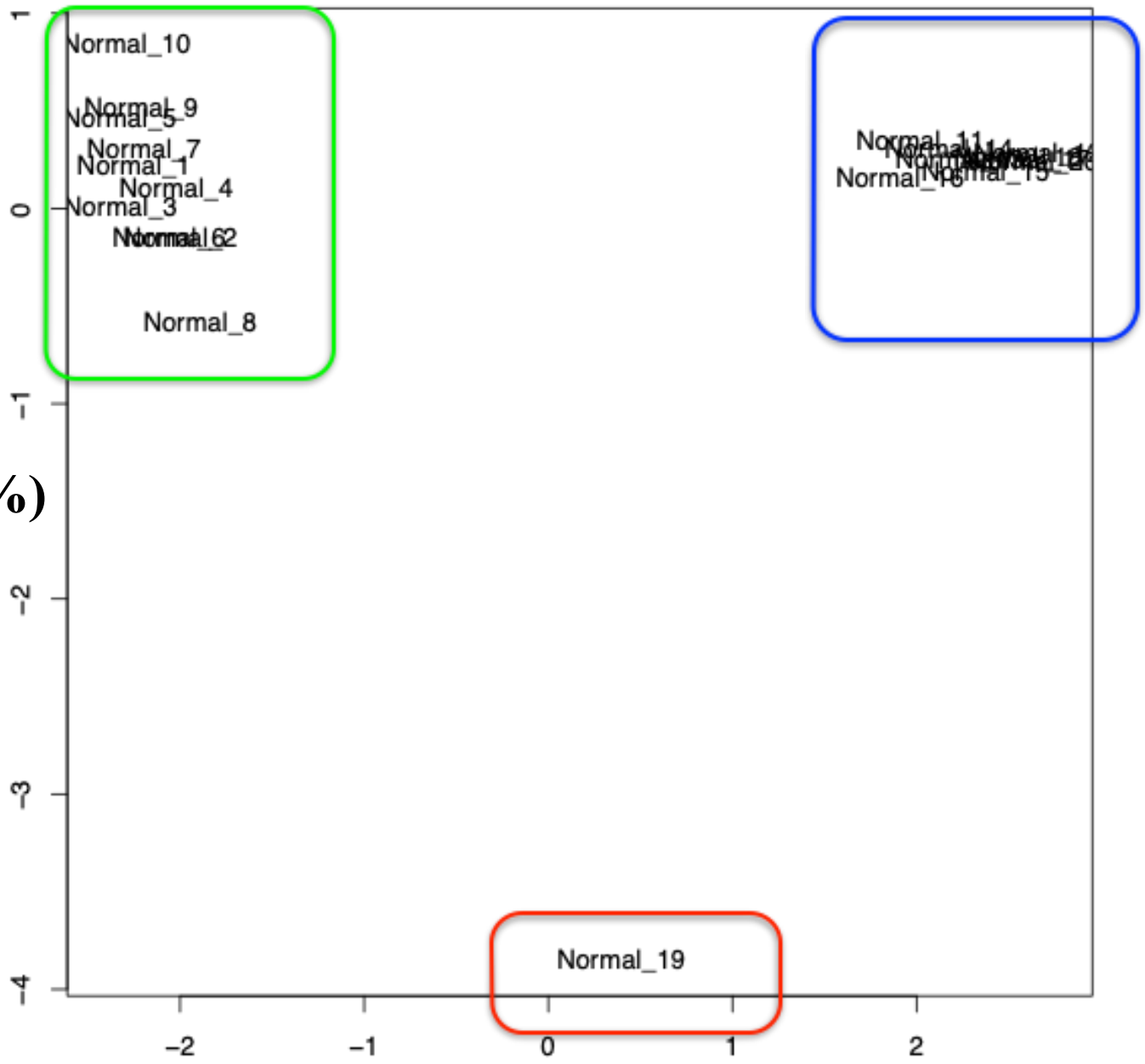


Kmeans in R

```
# k=3, 100 iterations
cl.10 <- kmeans(mat, 3, iter.max = 100)
plot(mat, col = cl.10$cluster)
points(cl.10$centers, col = 1:5, pch = 8)
```



Once you have the clusters, what do you do with them ?



PC2 (12.8%)

PC1 (20.5%)

Exercise:

Cholera Dataset
(include a PCA with clusters)

Thank you for your attention