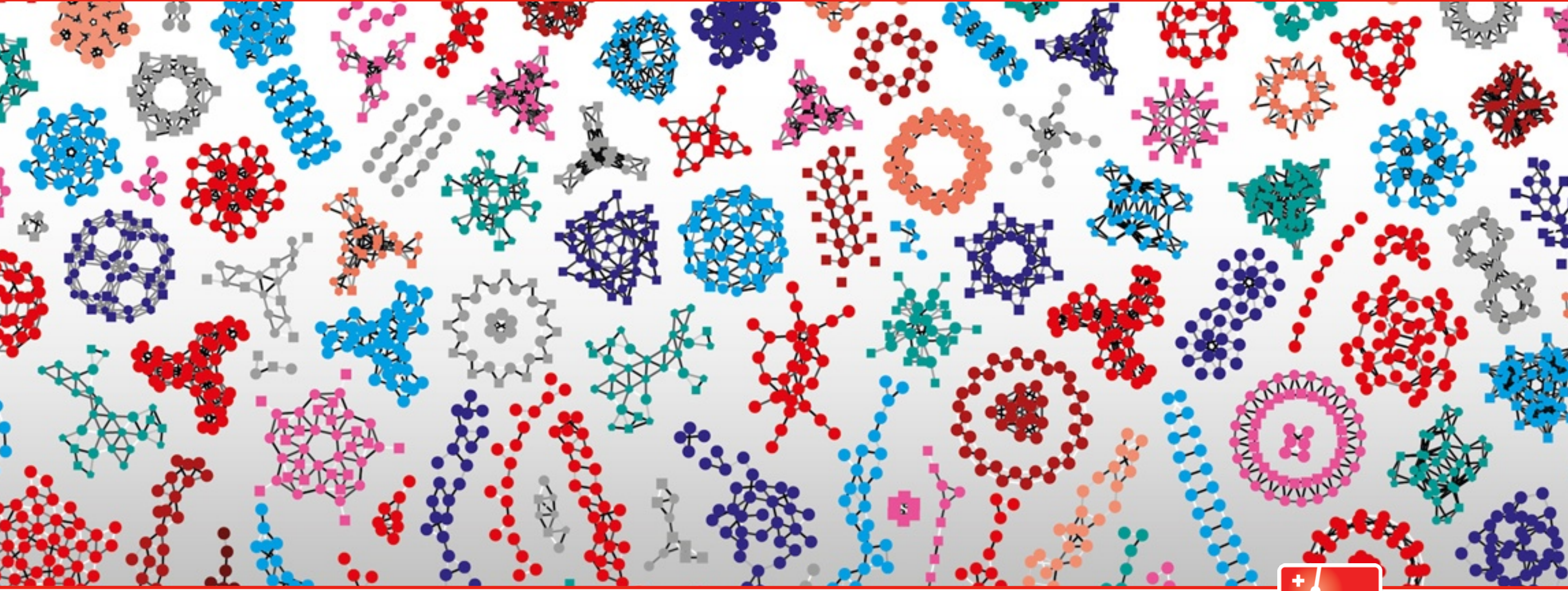


Statistical methods for big data in life sciences and health with R

Linda Dib, Frédéric Schütz

4th of June 2018

Swiss Institute of Bioinformatics

BCF

www.sib.swiss

IMPORTANT

Course room

Monday, Tuesday, Wednesday:

– Génopode Building 2020

Thursday:

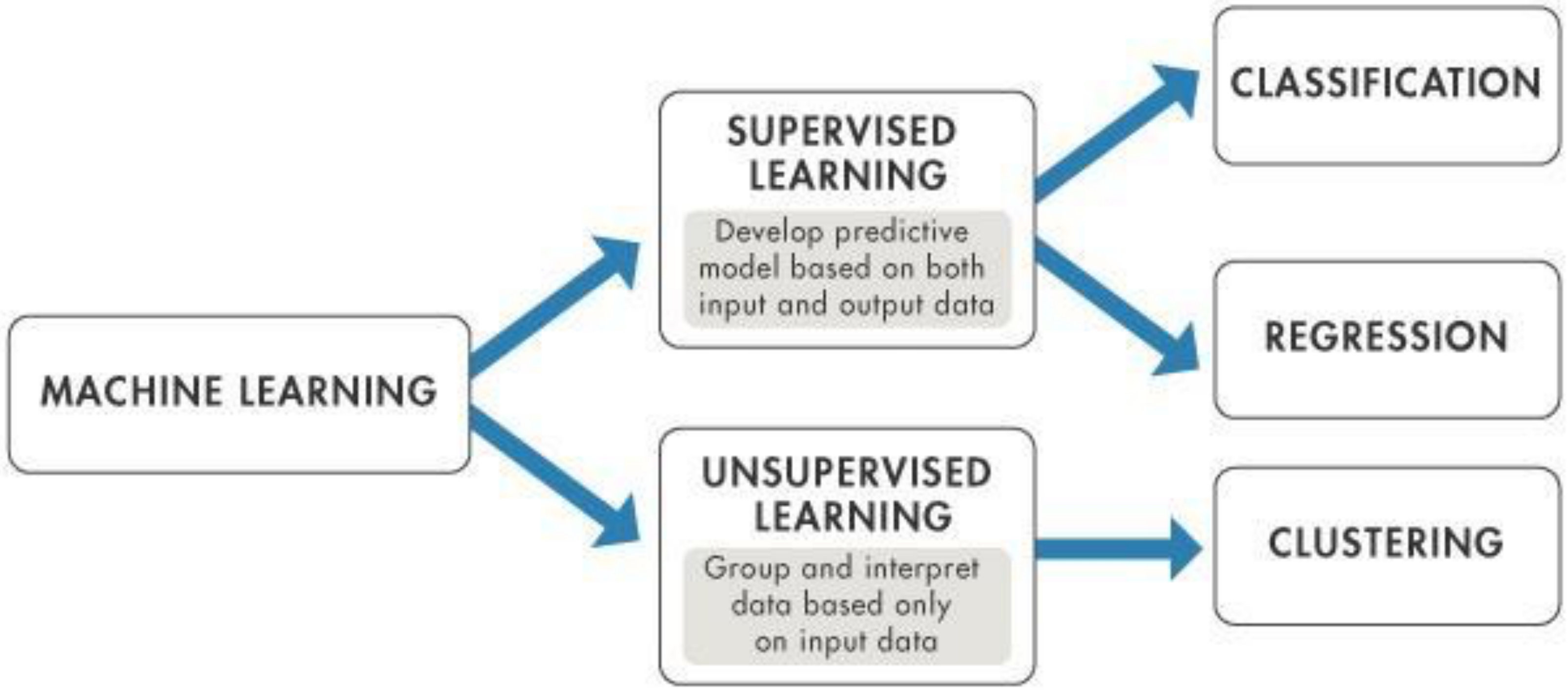
– Amphipôle Building 321

Course web-page

- Course page:
- https://edu.sib.swiss/course/view.php?id=344

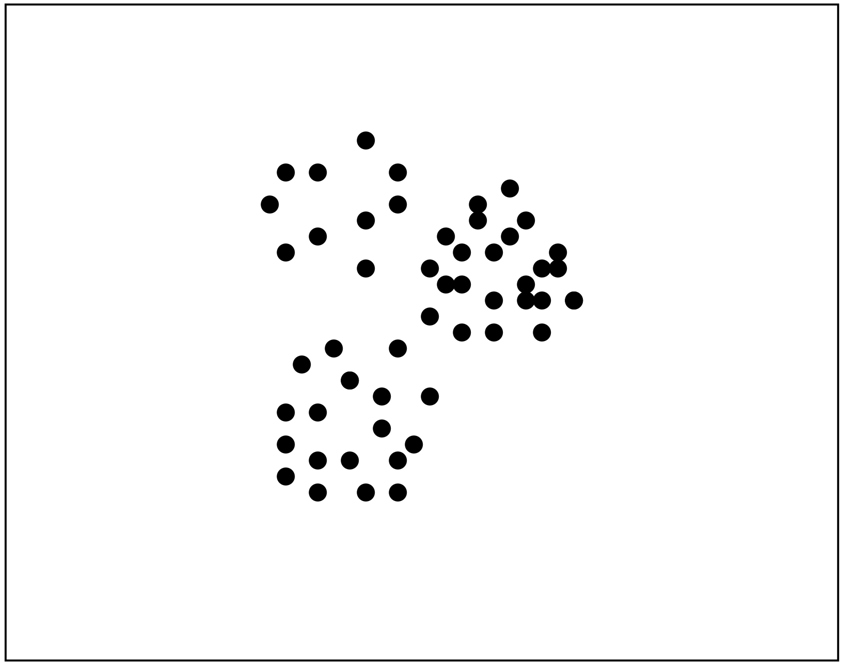
- Login: smbd18
- Password: SIB-smbd18

Machine Learning

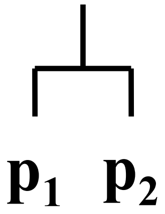

MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

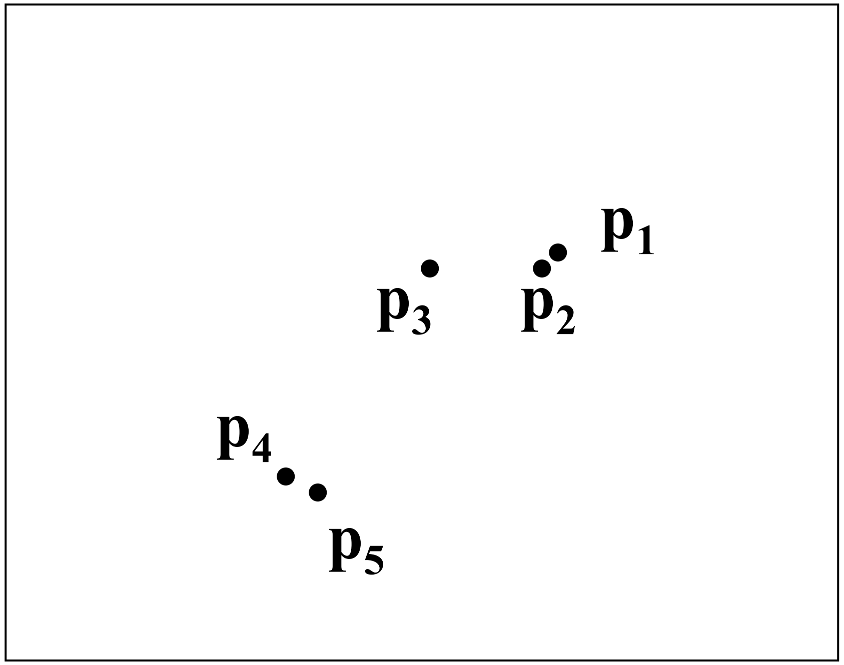
Credits: Rory Bunker, Fadi Thabtah

Clustering

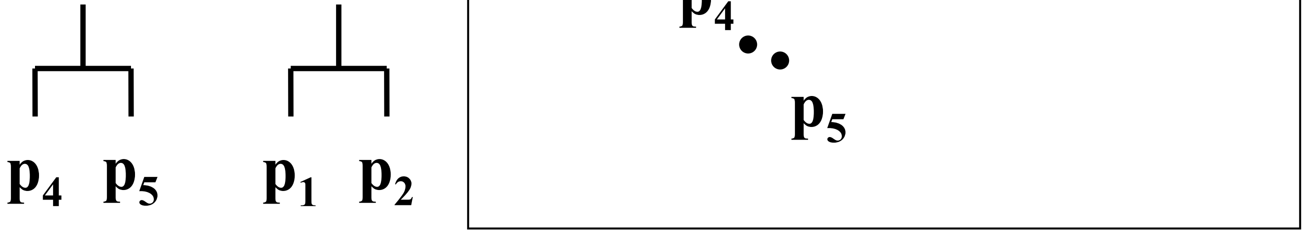


Hierarchical Clustering

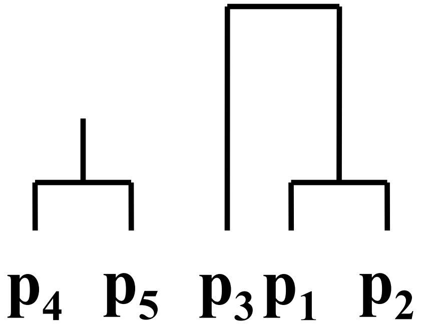


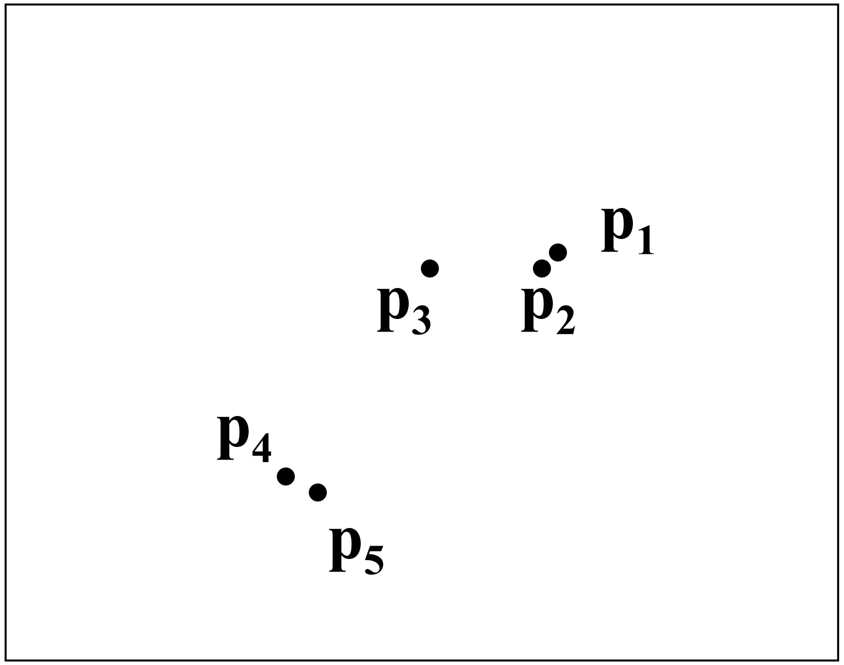


Hierarchical Clustering

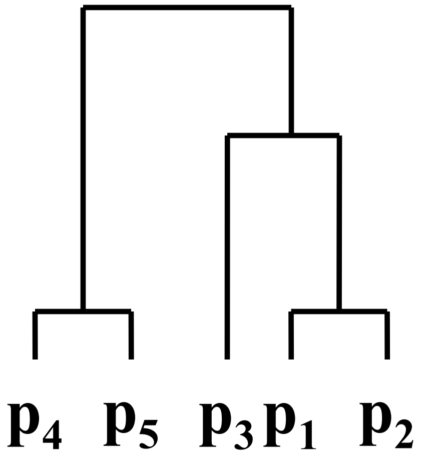


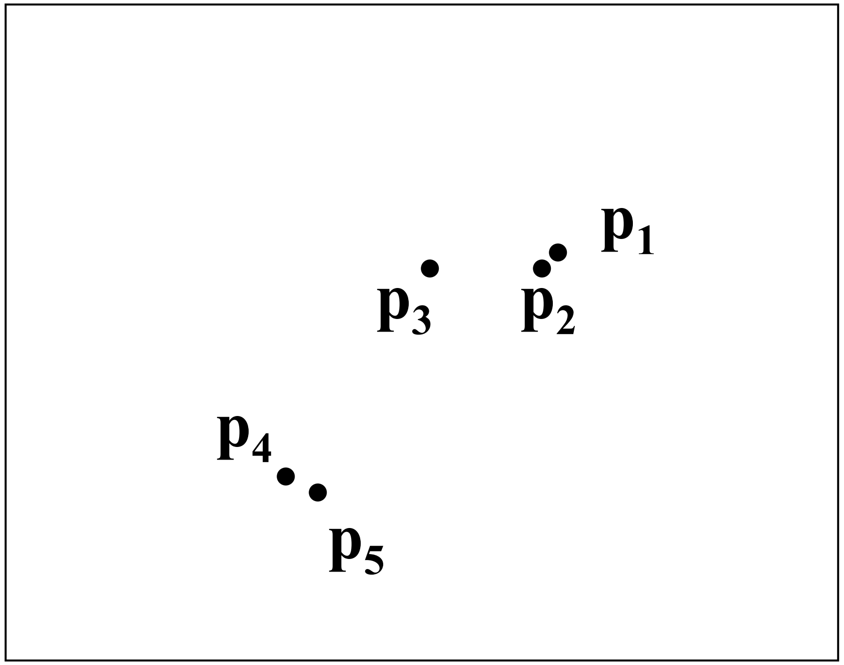
Hierarchical Clustering





Hierarchical Clustering





Distance

Euclidean

X = 2, 0

Y = -2, -2

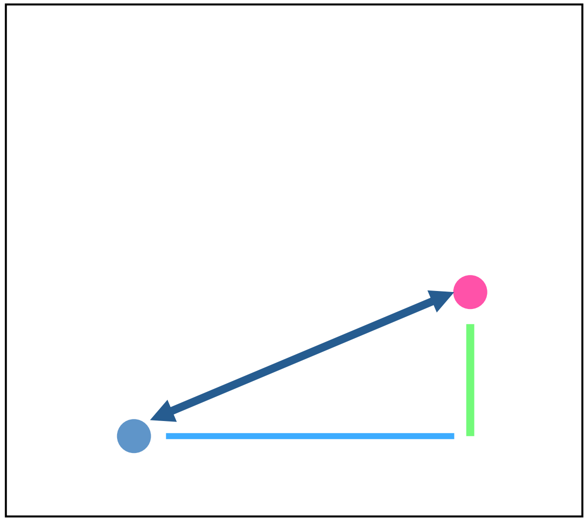
$\sqrt{[\Sigma (y - x)^2]}$

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

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

$= \sqrt{20}$

$= 4.47$



It represents the “multivariate dissimilarity” of X & Y

Squared Euclidean

X = 2, 0

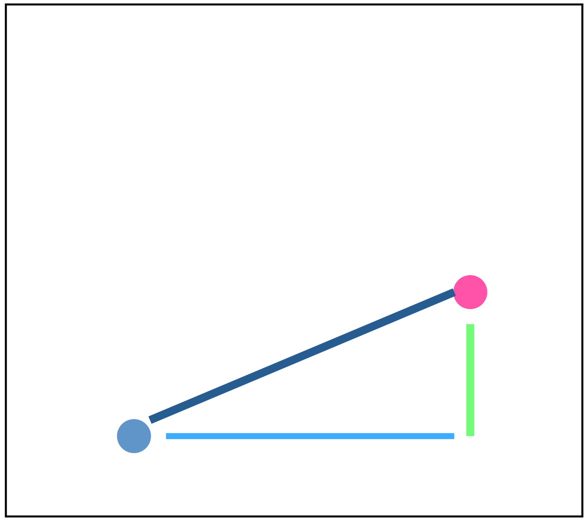
Y = -2, -2

$\Sigma (y - x)^2$

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

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

$= 20$



It represents the "multivariate dissimilarity" of X & Y

City Block

X = 2, 0

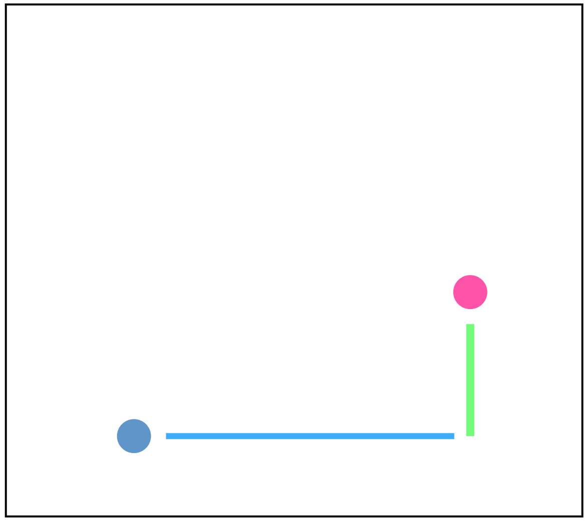
Y = -2, -2

Σ ly – xl

= ([-2 – 2] + [-2 – 2])

= (4 + 2)

= 6



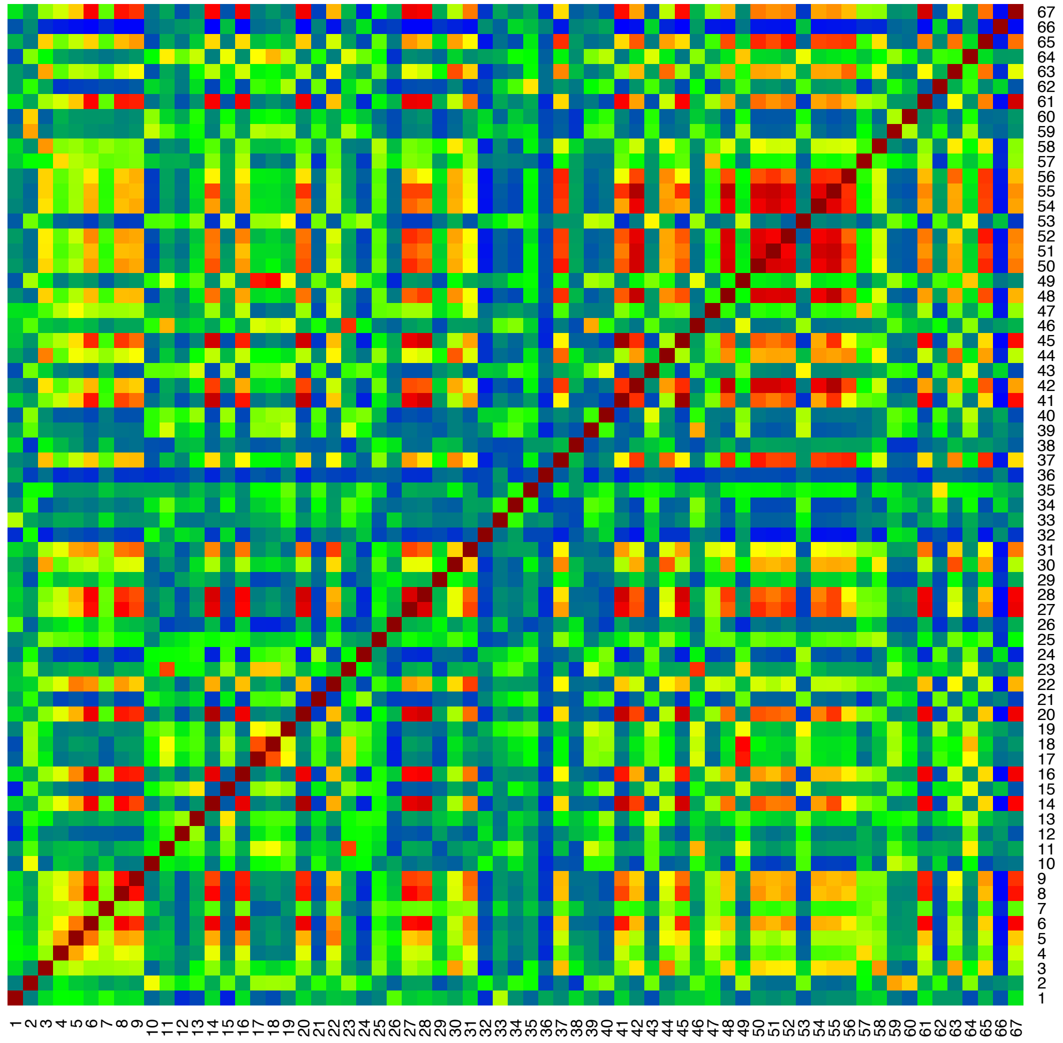
Distance Measures in 2D

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

In R

```
>?dist
```

Distance matrix



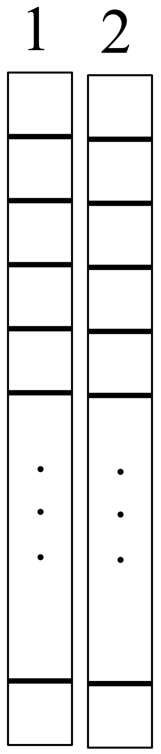
In R

```
>?heatmap
>heatmap(distanceMatrix,Colv=NA, Rowv=NA,
scale="none")
```

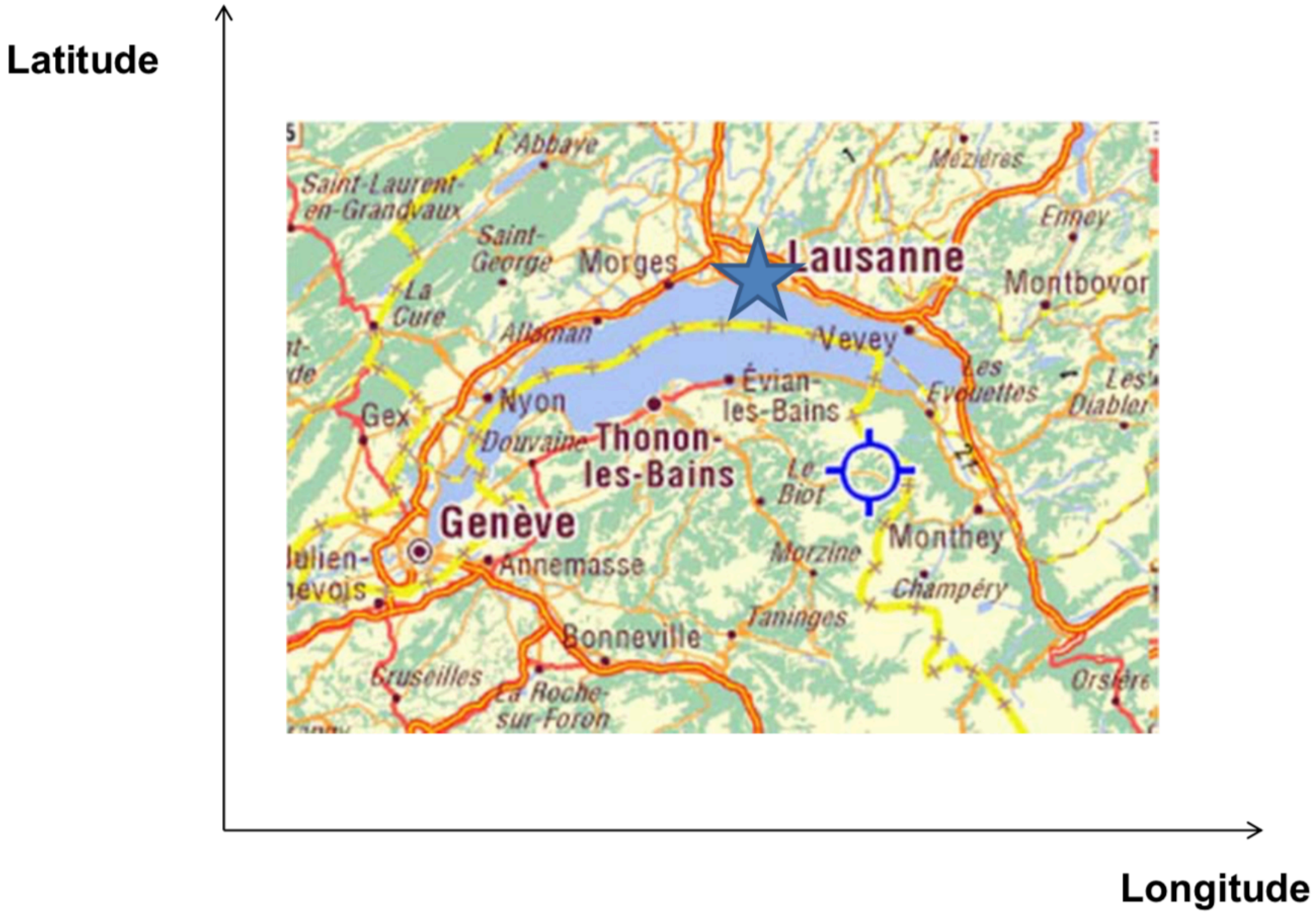
Dimension

Dimension:
the number of coordinates we need to locate a point in a given space.

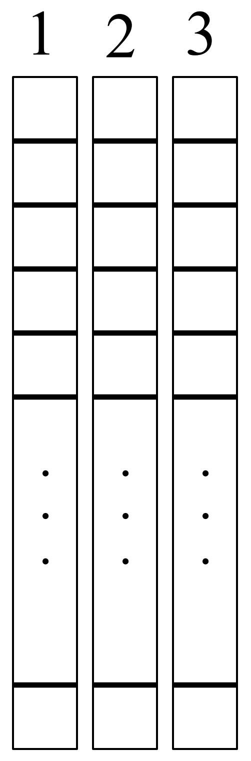
2-dimension



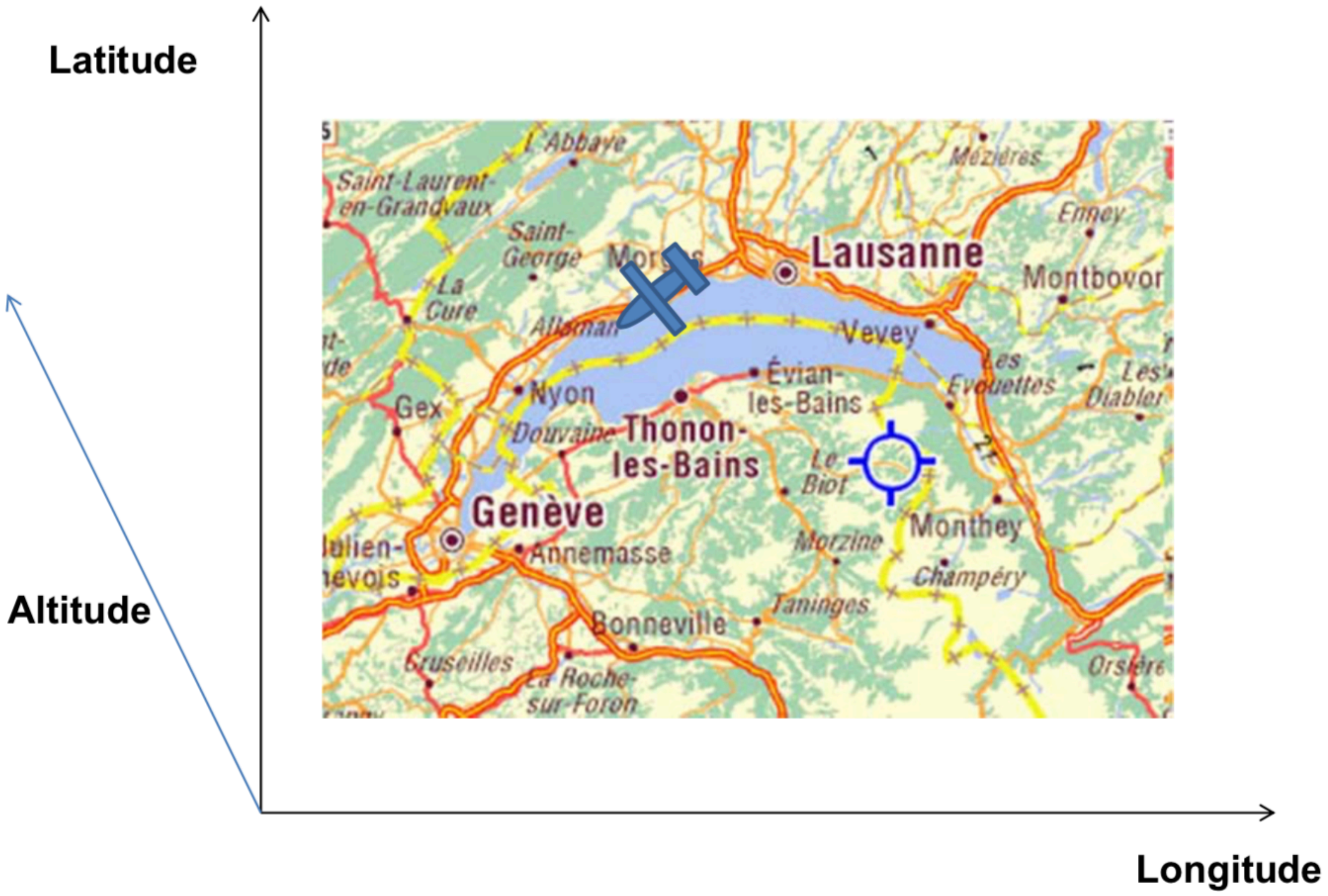
Two dimensions: latitude and longitude



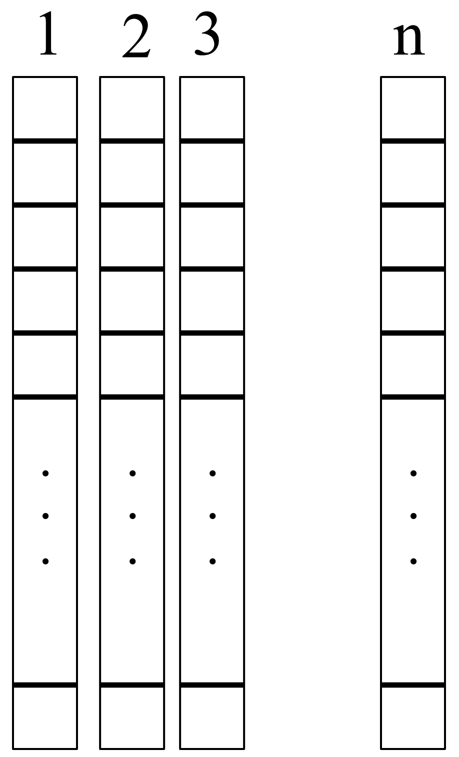
3-dimension



Three dimensions: latitude, longitude and altitude



n-dimension



Dimension in biology?

Example: Peptide

- peptide length
- peptide molecular weight
- peptide extinction coefficient
- peptide net charge at neutral pH
- peptide iso-electric point
- peptide water solubility

Dimension in biology?

Genes

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

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

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

#show heatmap
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none")

#change heatmap’s color
> colorScale <- colorRampPalette(c("blue",
"green","yellow","red","darkred"))(1000)
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none”,
col=colorScale)
```

In R

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

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

#show heatmap
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none")

#change heatmap’s color
> colorScale <- colorRampPalette(c("blue",
"green","yellow","red","darkred"))(1000)
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none”,
col=colorScale)
```

In R

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

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

#show heatmap
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none")

#change heatmap’s color
> colorScale <- colorRampPalette(c("blue",
"green","yellow","red","darkred"))(1000)
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none”,
col=colorScale)
```

In R

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

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

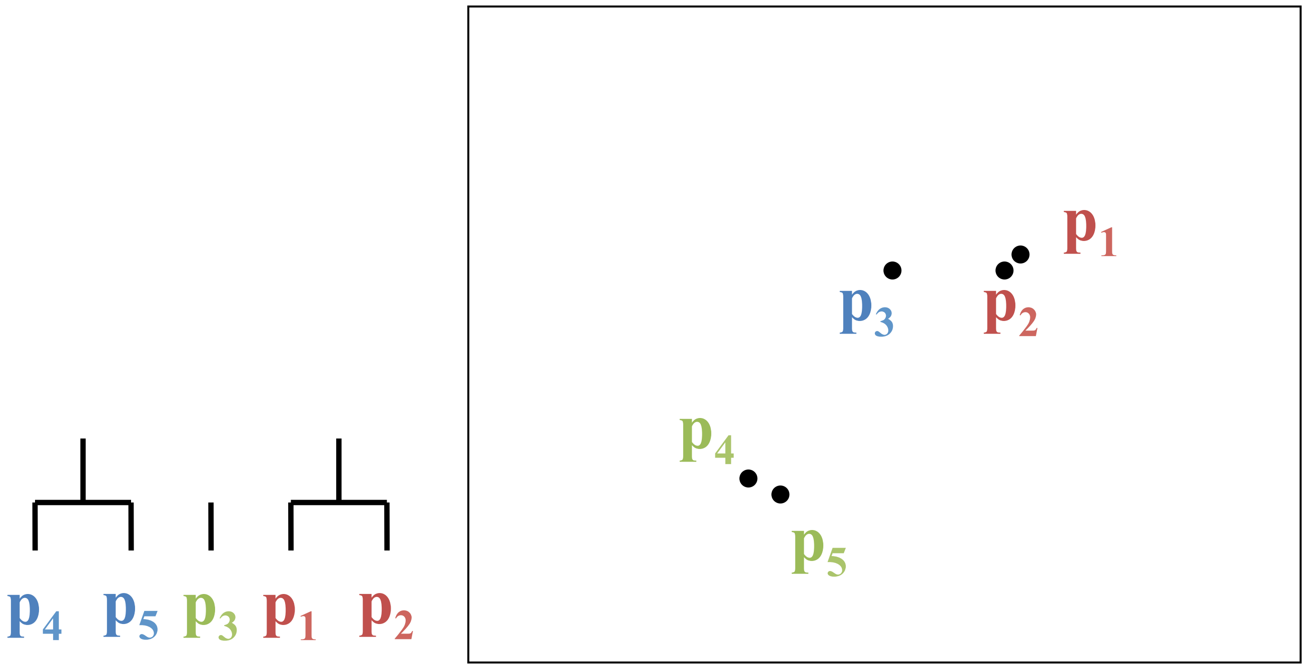
```
#show heatmap
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none")
```

```
#change heatmap’s color
> colorScale <- colorRampPalette(c("blue",
"green","yellow","red","darkred"))(1000)
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none”,
col=colorScale)
```

In R

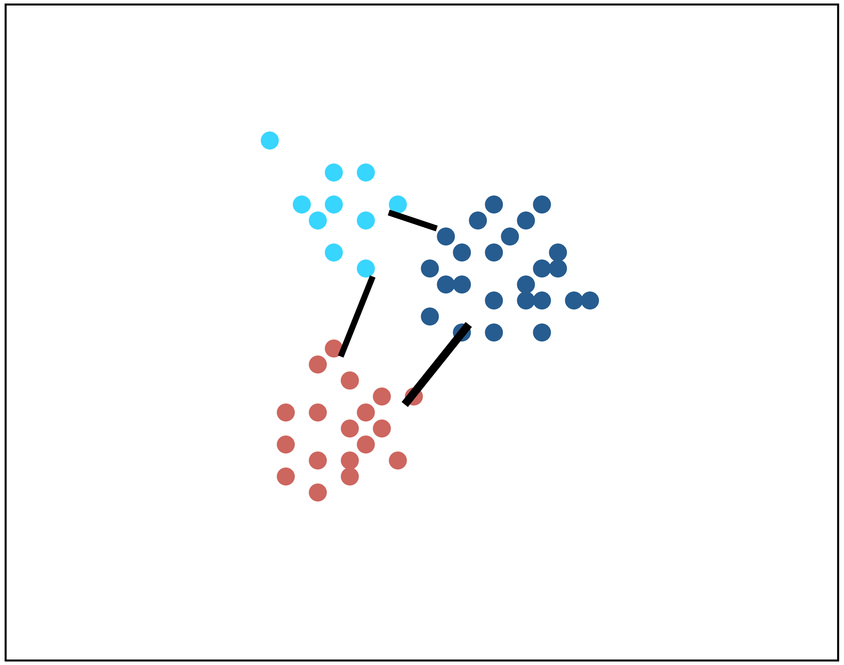
```
#create a random matrix
>mat <- matrix(data = rnorm(300, mean= 100,
sd=10), nrow = 150, ncol = 2)

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

#show heatmap
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none")

#change heatmap’s color
> colorScale <- colorRampPalette(c("blue",
"green","yellow","red","darkred"))(1000)
>heatmap(mat.dist,Colv=NA, Rowv=NA, scale="none”,
col=colorScale)
```

How to aggregate clusters?


p4
p5
p3
p1
p2
p1
p3
p2
p4
p5

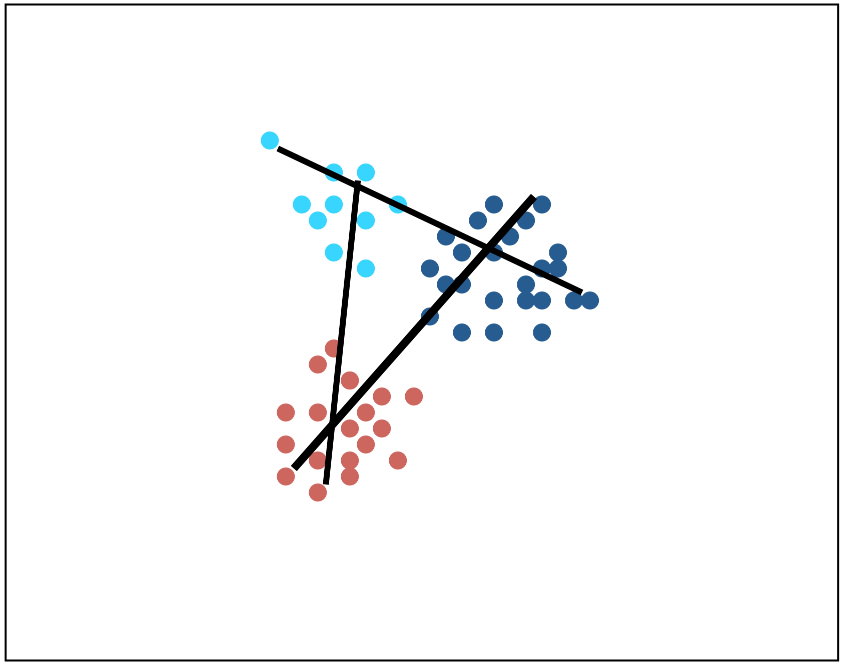
Which clusters to combine?

Single linkage



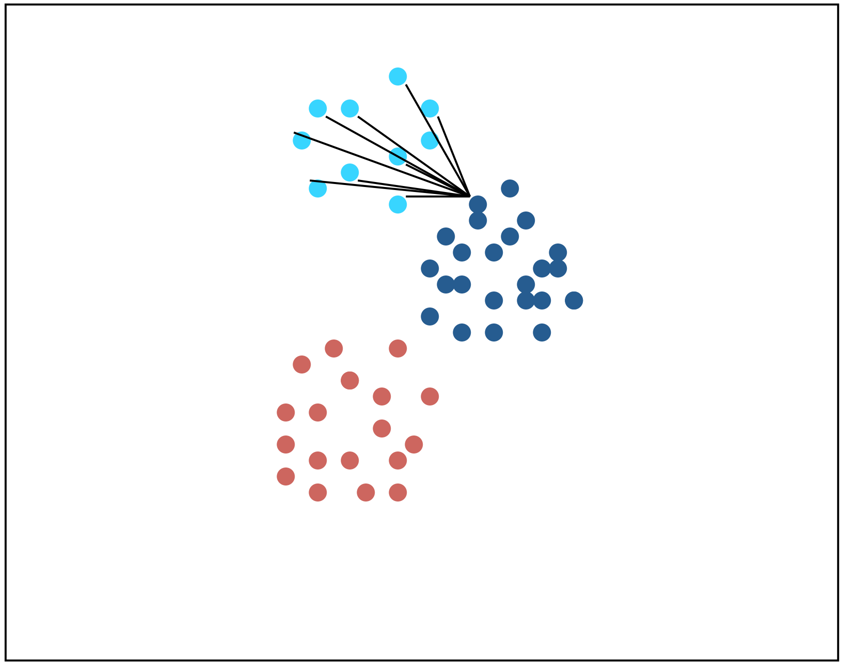
Distance between closest elements in clusters

Complete Linkage



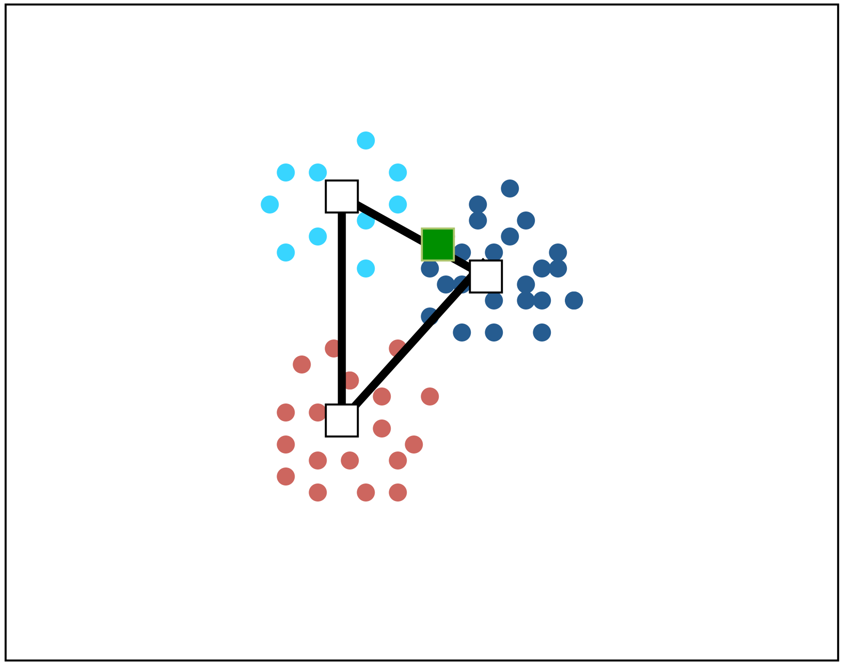
Distance between farthest elements in clusters

Average Linkage



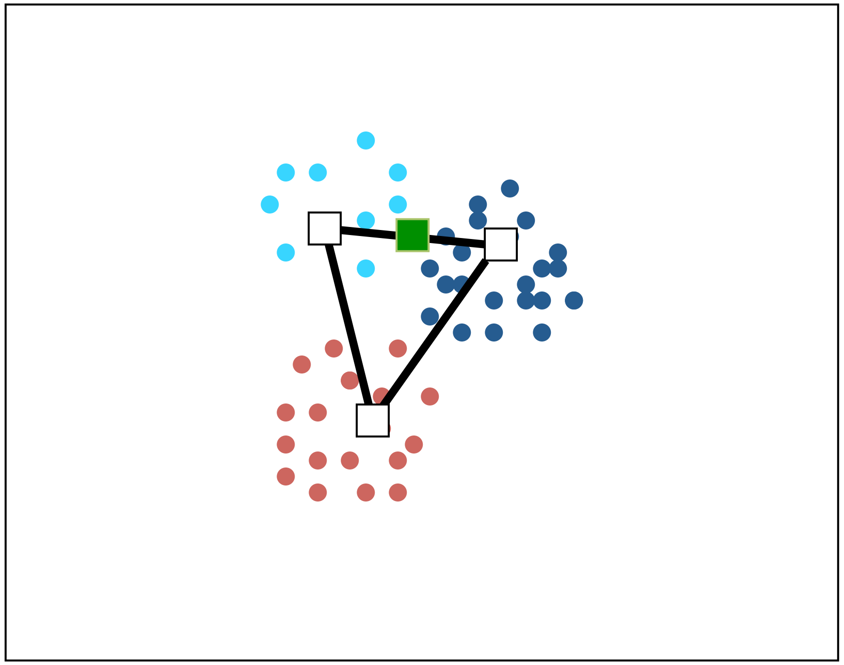
Average of all pairwise distances

Centroid Condensation (mean)



Distance between centroids (means) of two clusters

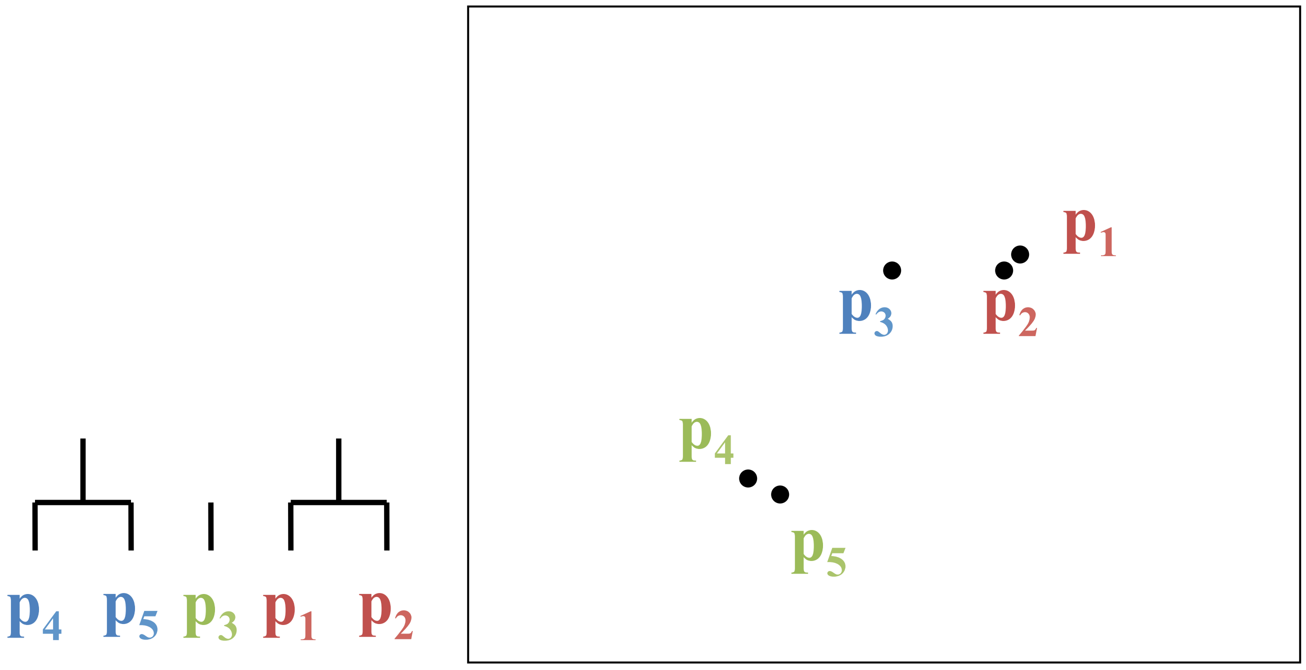
Median Condensation



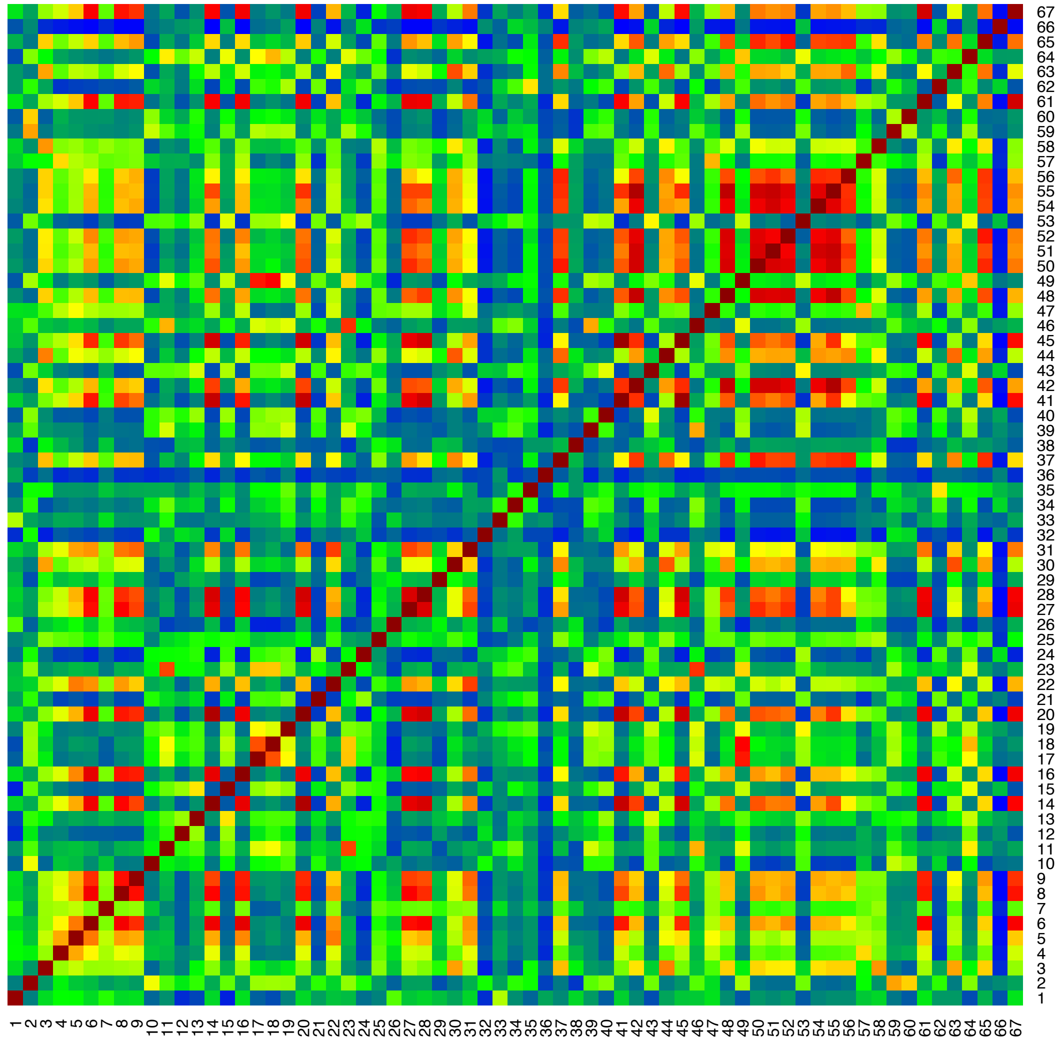
Distance between median distances of two clusters

Clustering methods

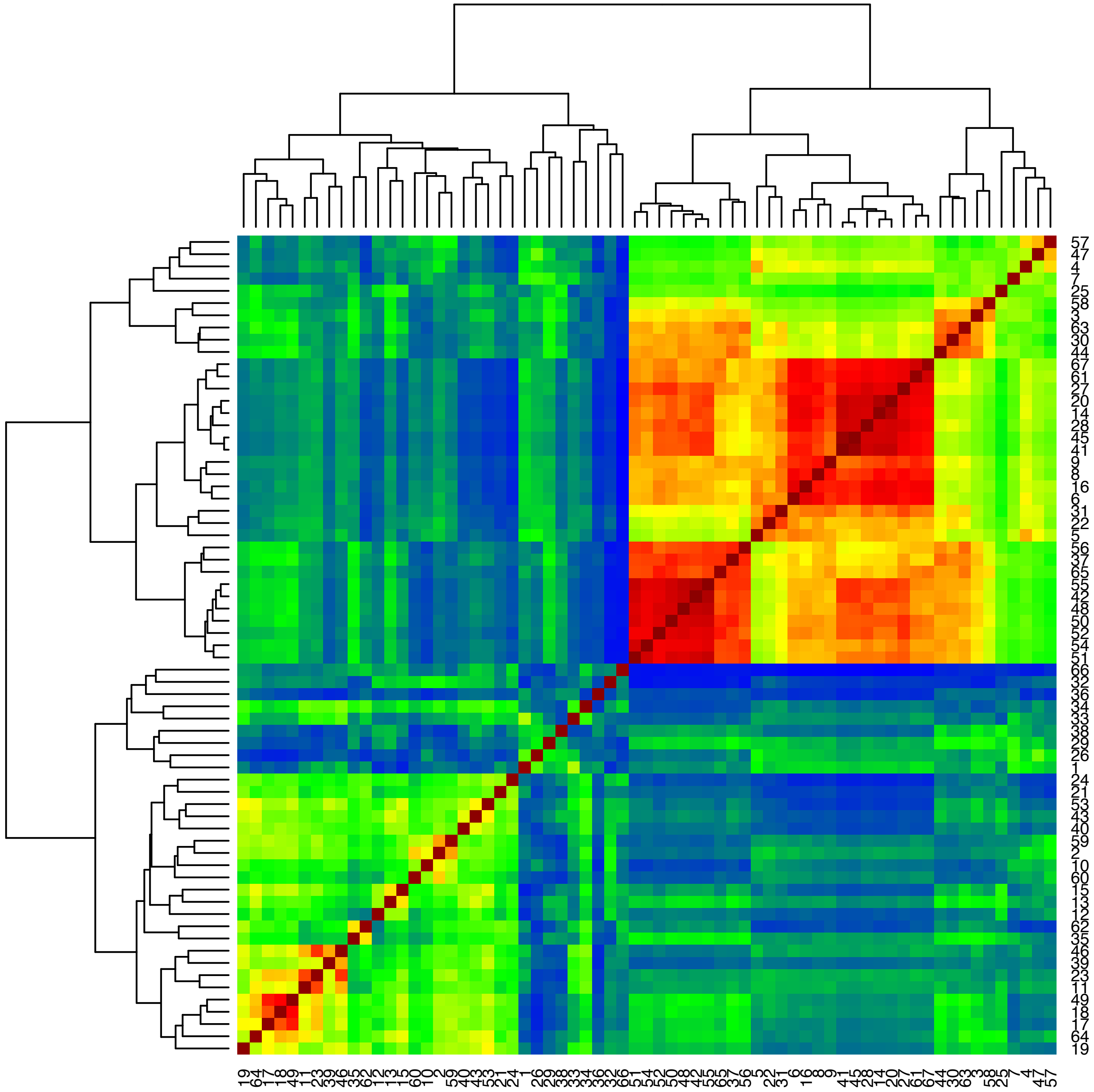
Hierarchical Clustering



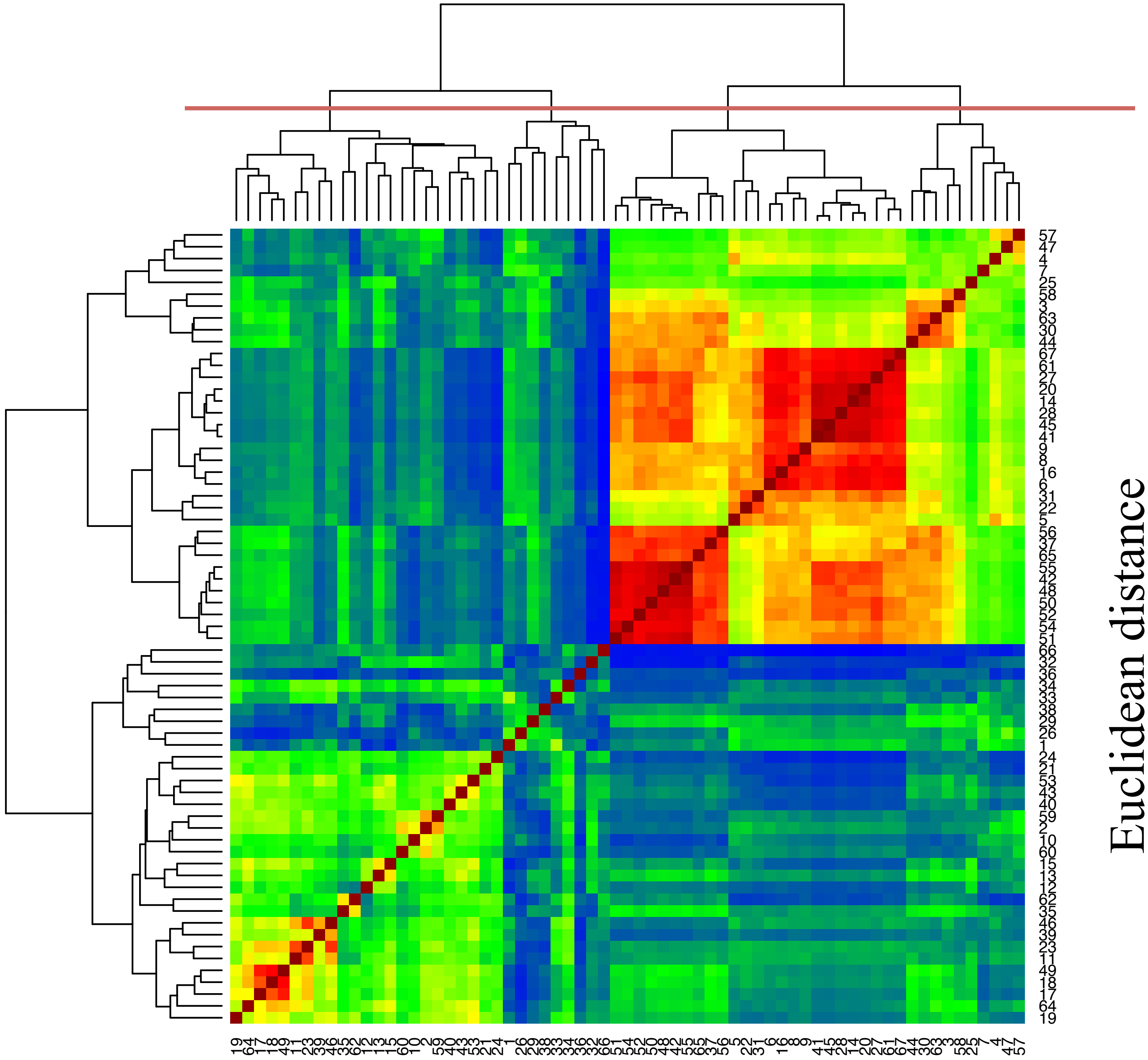
At the beginning every point is a cluster in it self, then we agglomerate …

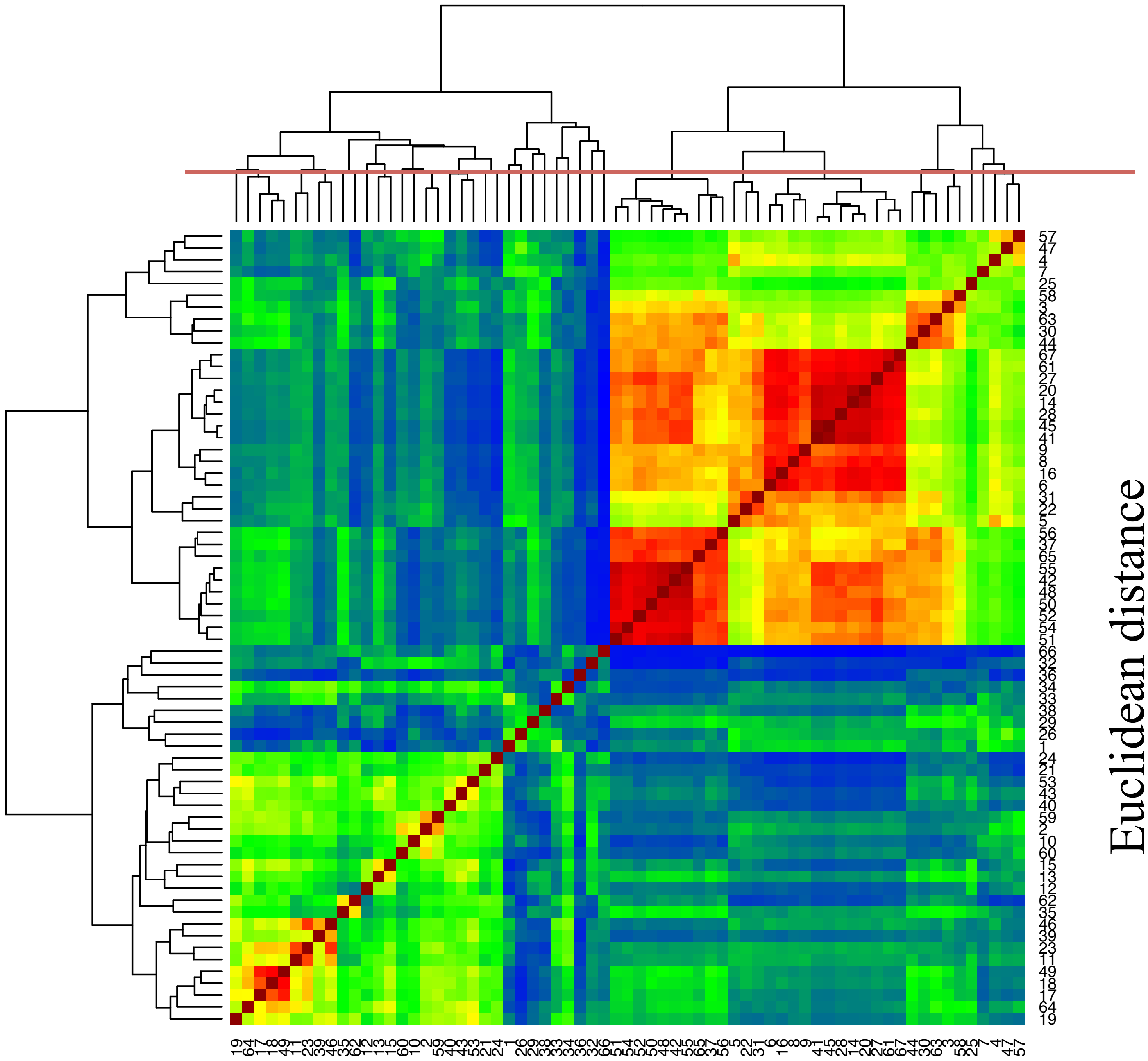


Euclidean distance



Euclidean distance
complete Linkage


Euclidean distance
complete Linkage



Euclidean distance
complete Linkage

How to do it in R

```
>?hclust

>mat <- matrix(data = rnorm(300, mean= 100, sd=10), nrow =
150, ncol = 2)
>distE<- dist(mat)
>distC<- dist(mat,method="manhattan")

>mat.distE<-as.matrix(dist(mat))
>mat.distC<-as.matrix(dist(mat,method="manhattan")

>heatmap(mat.distE, Colv=NA, Rowv=NA, scale="none")
>heatmap(mat.distC, Colv=NA, Rowv=NA, scale="none")

>hE<-hclust(distE,"complete")
>hC<-hclust(distC,"complete")

>plot(hE)
>plot(hC)
```

How to do it in R

```
>?hclust

>mat <- matrix(data = rnorm(300, mean= 100, sd=10), nrow =
150, ncol = 2)
>distE<- dist(mat)
>distC<- dist(mat,method="manhattan")

>mat.distE<-as.matrix(dist(mat))
>mat.distC<-as.matrix(dist(mat,method="manhattan"))

>heatmap(mat.distE, Colv=NA, Rowv=NA, scale="none")
>heatmap(mat.distC, Colv=NA, Rowv=NA, scale="none")

>hE<-hclust(distE,"complete")
>hC<-hclust(distC,"complete")

>plot(hE)
>plot(hC)
```

How to do it in R

```
>?hclust

>mat <- matrix(data = rnorm(300, mean= 100, sd=10), nrow =
150, ncol = 2)
>distE<- dist(mat)
>distC<- dist(mat,method=“manhattan”)

>mat.distE<-as.matrix(dist(mat))
>mat.distC<-as.matrix(dist(mat,method=“manhattan”)

>heatmap(mat.distE, Colv=NA, Rowv=NA, scale="none")
>heatmap(mat.distC, Colv=NA, Rowv=NA, scale="none”)

>hE<-hclust(distE,”complete")
>hC<-hclust(distC,”complete")

>plot(hE)
>plot(hC)
```

Hierarchical cluster on big data sets

Hierarchical cluster on big data sets

hclust doesn’t work on large dataset

Hierarchical cluster on big data sets

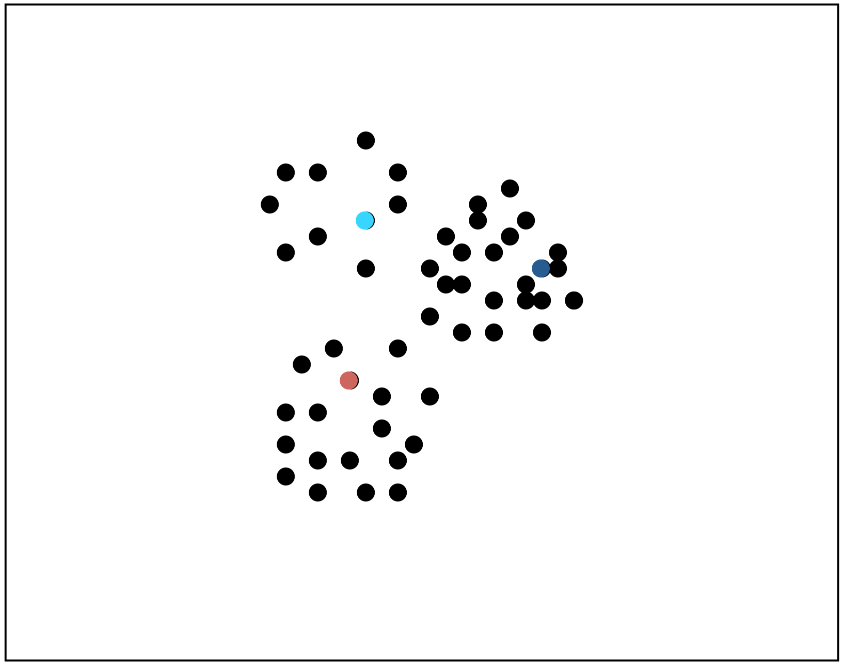
hclust doesn’t work on large dataset

Solution:

You can use **kmeans**, which normally suitable for this amount of data, to calculate an important number of centers (1000, 2000, ...) and perform a hierarchical clustering approach on the coordinates of these centers.

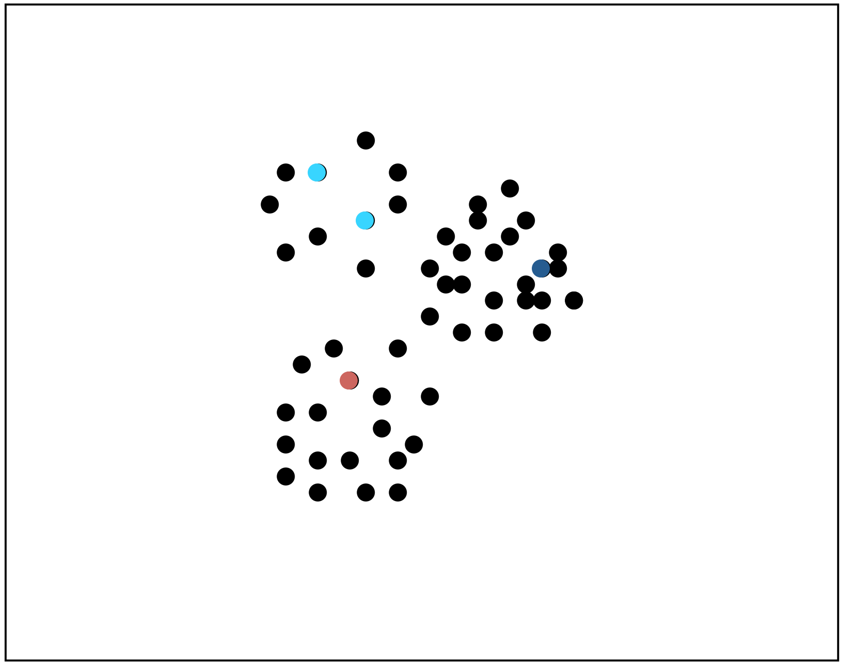
K-means Clustering

Number of clusters = 3



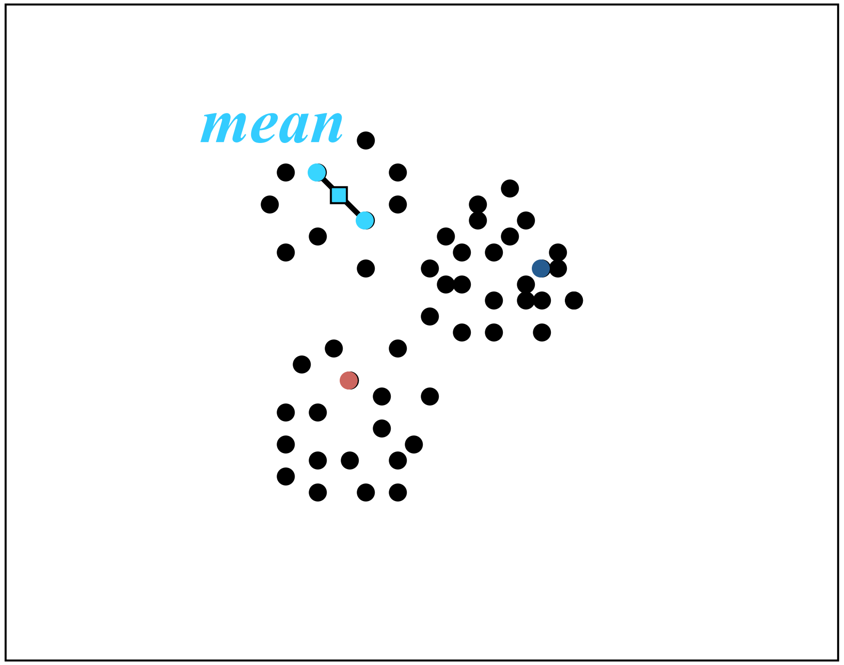
K-means Clustering

Number of clusters = 3



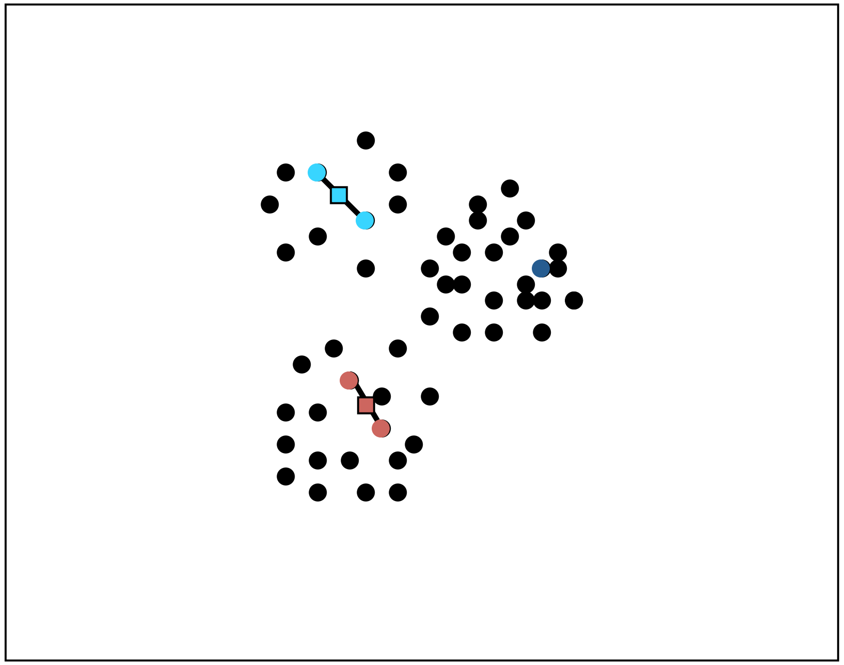
K-means Clustering

Number of clusters = 3



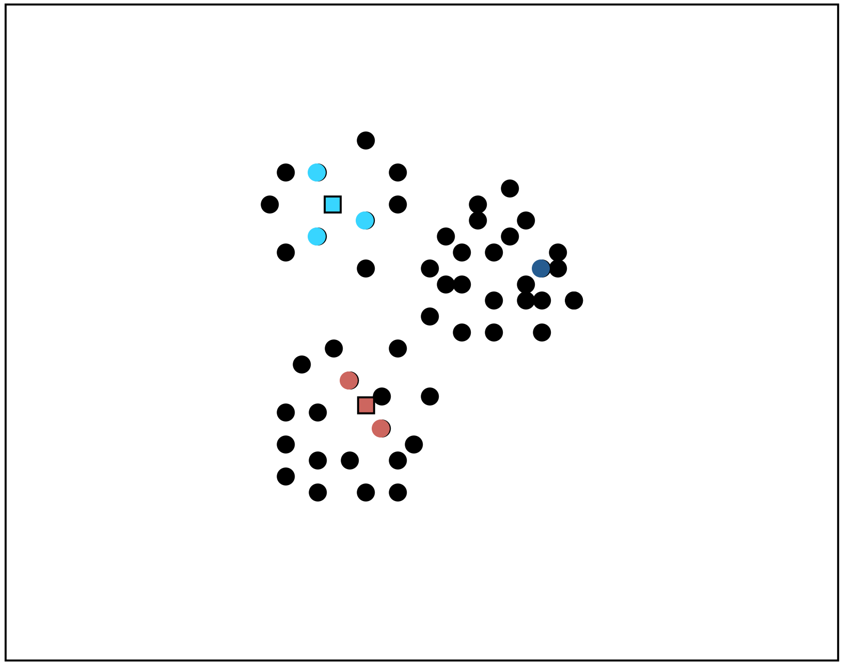
K-means Clustering

Number of clusters = 3



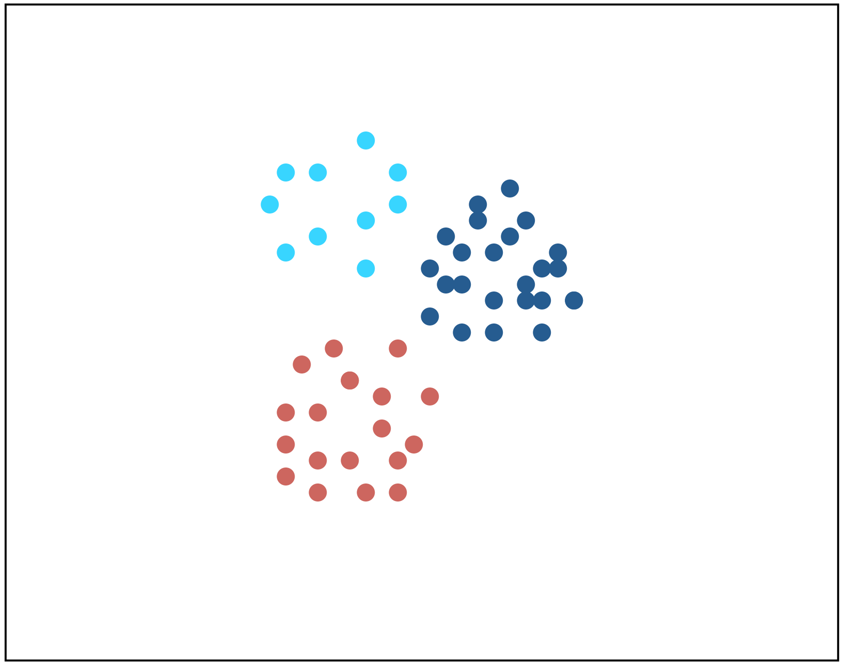
K-means Clustering

Number of clusters = 3



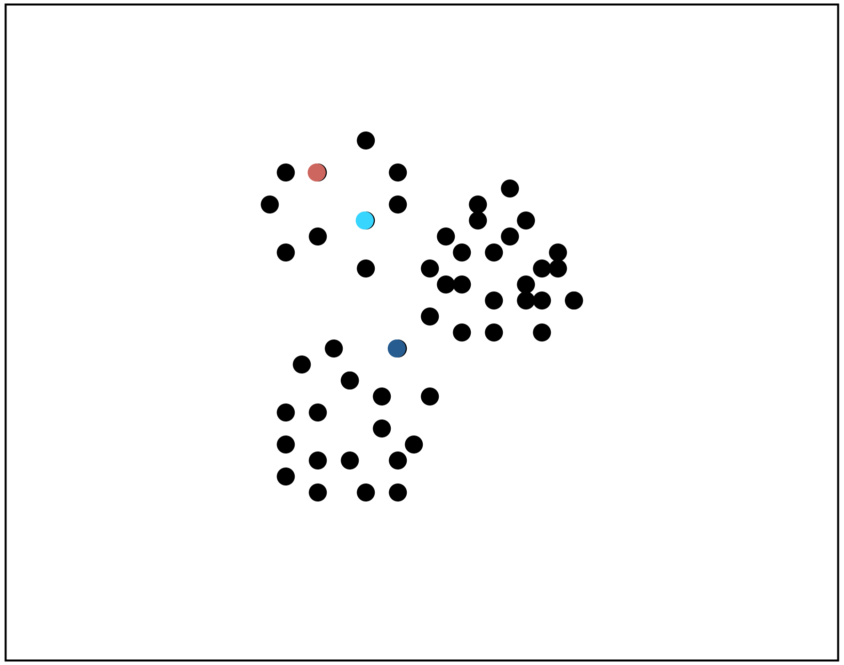
K-means Clustering

Number of clusters = 3



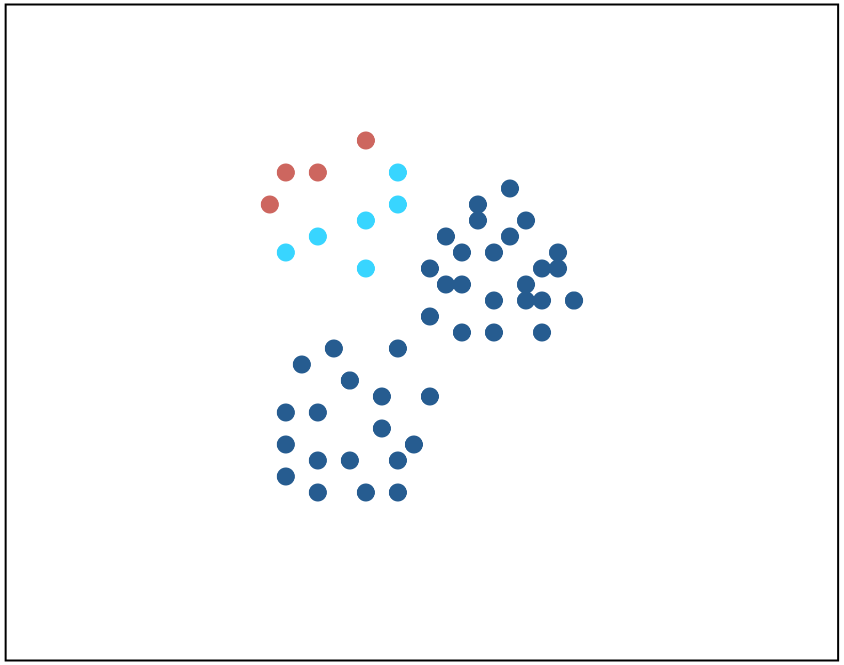
K-means Clustering

Number of clusters = 3



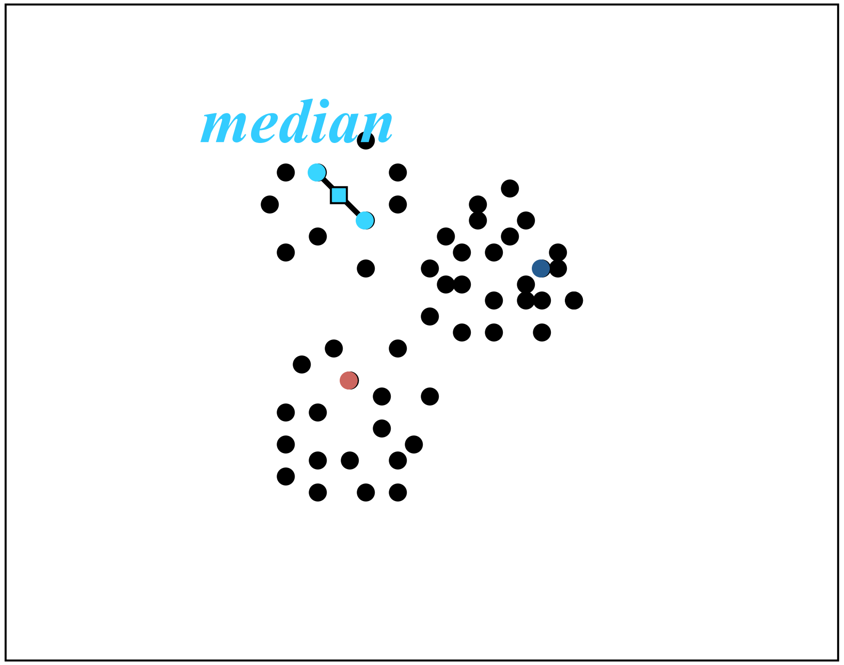
K-means Clustering

Number of clusters = 3



Fuzzy C-means Clustering

Number of clusters = 3



In R

```
>mat <- matrix(data = rnorm(300, mean= 100, sd=10),
               nrow = 150,
               ncol = 2
>df<-data.frame(x)

>kmeans(df,3)
```

In R

```
>mat <- matrix(data = rnorm(300, mean= 100, sd=10),
               nrow = 150,
               ncol = 2
>df<-data.frame(x)

>kmeans(df,3)

>cl.1 <- kmeans(df, 3, iter.max = 1)
>plot(df, col = cl.1$cluster)
>points(cl.1$centers, col = 1:5, pch = 8)
```

In R

```
>mat <- matrix(data = rnorm(300, mean= 100, sd=10),
               nrow = 150,
               ncol = 2
>df<-data.frame(x)

>kmeans(df,3)

>cl.1 <- kmeans(df, 3, iter.max = 1)
>plot(df, col = cl.1$cluster)
>points(cl.1$centers, col = 1:5, pch = 8)

>cl.10 <- kmeans(df, 3, iter.max = 10)
>plot(df, col = cl.10$cluster)
>points(cl.10$centers, col = 1:5, pch = 8)

>cl.100 <- kmeans(df, 3, iter.max = 100)
>plot(df, col = cl.100$cluster)
>points(cl.100$centers, col = 1:5, pch = 8)
```

Hierarchical cluster on big data sets

Use kmeans as an intermediate step

```
>x<- rbind(matrix(rnorm(70000, sd = 0.3), ncol = 2),
           matrix(rnorm(70000, mean = 1, sd = 0.3),
           ncol = 2))
>colnames(x) <- c("x", "y")
>cl     <- kmeans(x, 1000, iter.max=20)
>cah    <- hclust(cl$centers, graph=FALSE, nb.clust=-1)
```

Hierarchical cluster on big data sets

Use kmeans as an intermediate step

```
>x<- rbind(matrix(rnorm(70000, sd = 0.3), ncol = 2),
           matrix(rnorm(70000, mean = 1, sd = 0.3),
           ncol = 2))
>colnames(x) <- c("x", "y")
>cl     <- kmeans(x, 1000, iter.max=20)
>cah    <- hclust(cl$centers, graph=FALSE, nb.clust=-1)
```

Hierarchical cluster on big data sets

Use kmeans as an intermediate step

```
>x<- rbind(matrix(rnorm(70000, sd = 0.3), ncol = 2),
           matrix(rnorm(70000, mean = 1, sd = 0.3),
           ncol = 2))
>colnames(x) <- c("x", "y")
>cl     <- kmeans(x, 1000, iter.max=20)
>cah    <- hclust(dist(cl$centers), graph=FALSE,
nb.clust=-1)
```

K-means cluster on big data

But what if you have a data set that won’t fit into memory?

RevoScaleR solution

RevoScaleR package has new k-means function implementation: rxKmeans

It is implemented as an external memory algorithm that works on a chunk of data at a time.

Once all of the chunks have been processed, the means are updated one last time to produce the final result.

rxKmeans

```
#Step 1: Prep and Import Data
#Initialize some variables to specify the data sets.
inputFileData <- paste0("/media/sf_docVM/",
"dataClustering.csv")

#Import the  data.
clustering_data<- rxImport(inData = inputFileData)

#run kmeans
z<-rxKmeans(~ Coord_X + Coord_Y, data =
clustering_data, numClusters = 3, maxIterations=100)

#plot outcome
DF <-
data.frame(clustering_data$Coord_X,clustering_data$Coor
d_Y)
plot(DF, col = z$cluster)
points(z$centers, col = 1:5, pch = 8)
```

rxKmeans

```
#Step 1: Prep and Import Data
#Initialize some variables to specify the data sets.
inputFileData <- paste0("/media/sf_docVM/",
"dataClustering.csv")

#Import the  data.
clustering_data<- rxImport(inData = inputFileData)

#run kmeans
z<-rxKmeans(~ Coord_X + Coord_Y, data =
clustering_data, numClusters = 3, maxIterations=100)

#plot outcome
DF <-
data.frame(clustering_data$Coord_X,clustering_data$Coor
d_Y)
plot(DF, col = z$cluster)
points(z$centers, col = 1:5, pch = 8)
```

rxKmeans

```
#Step 1: Prep and Import Data
#Initialize some variables to specify the data sets.
inputFileData <- paste0("/media/sf_docVM/",
"dataClustering.csv")

#Import the  data.
clustering_data<- rxImport(inData = inputFileData)

#run kmeans
z<-rxKmeans(~ Coord_X + Coord_Y, data =
clustering_data, numClusters = 3, maxIterations=100)

#plot outcome
DF <-
data.frame(clustering_data$Coord_X,clustering_data$Coor
d_Y)
plot(DF, col = z$cluster)
points(z$centers, col = 1:5, pch = 8)
```

rxKmeans

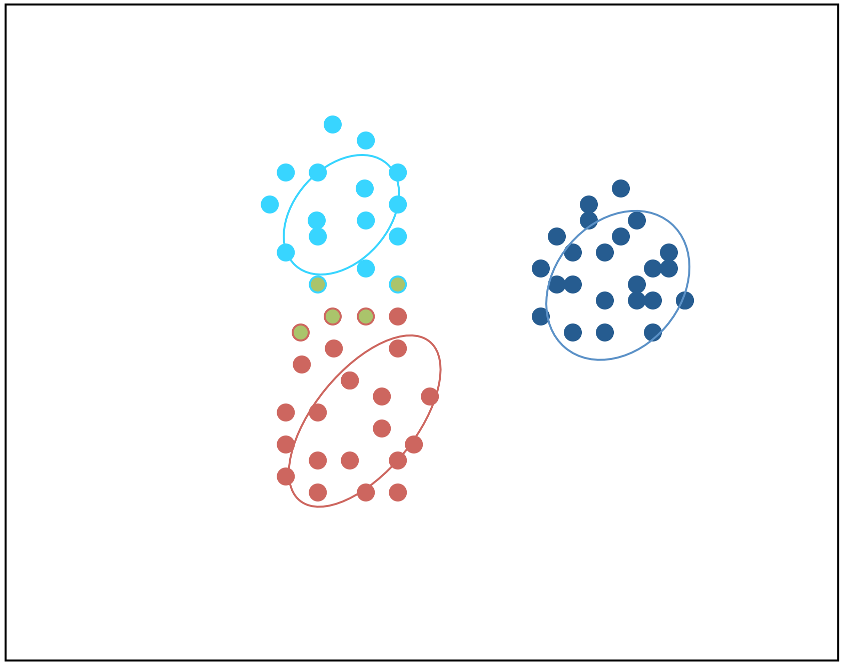
```
#Step 1: Prep and Import Data
#Initialize some variables to specify the data sets.
inputFileData <- paste0("/media/sf_docVM/",
"dataClustering.csv")

#Import the  data.
clustering_data<- rxImport(inData = inputFileData)

#run kmeans
z<-rxKmeans(~ Coord_X + Coord_Y, data =
clustering_data, numClusters = 3, maxIterations=100)

#plot outcome
DF <-
data.frame(clustering_data$Coord_X,clustering_data$Coor
d_Y)
plot(DF, col = z$cluster)
points(z$centers, col = 1:5, pch = 8)
```

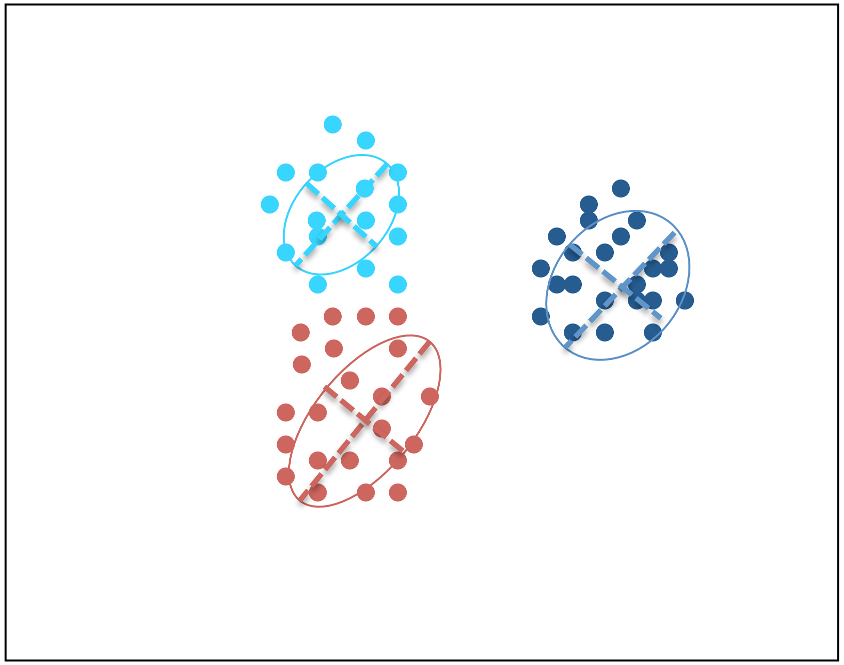
K-means & C-means

Drawbacks:

1. Specify number of clusters
2. Non probabilistic methods

Model-based Clustering



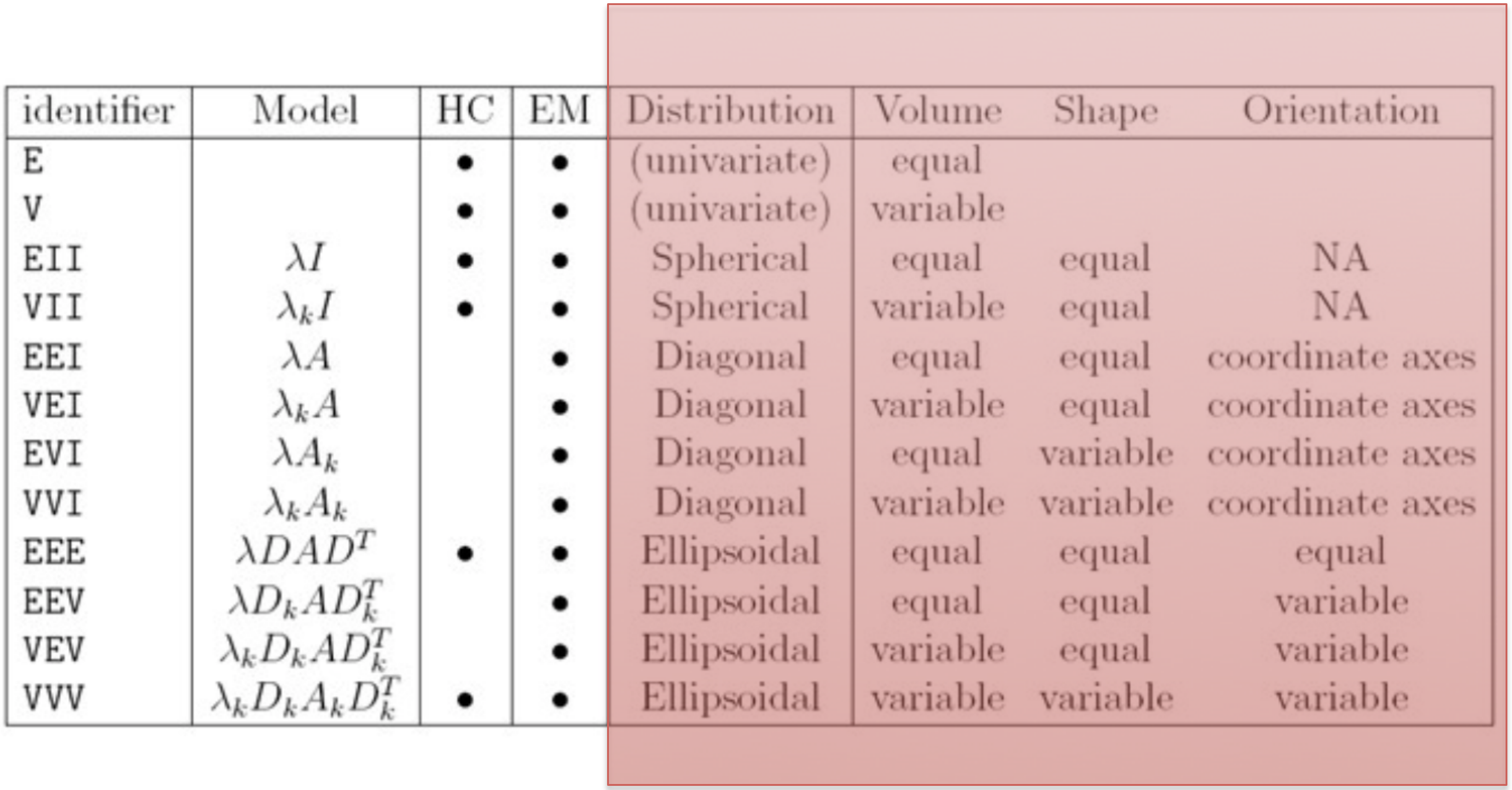


distribution
(univariate, spherical, diagonal,
elipsoidal)

data volume
(equal, variable)

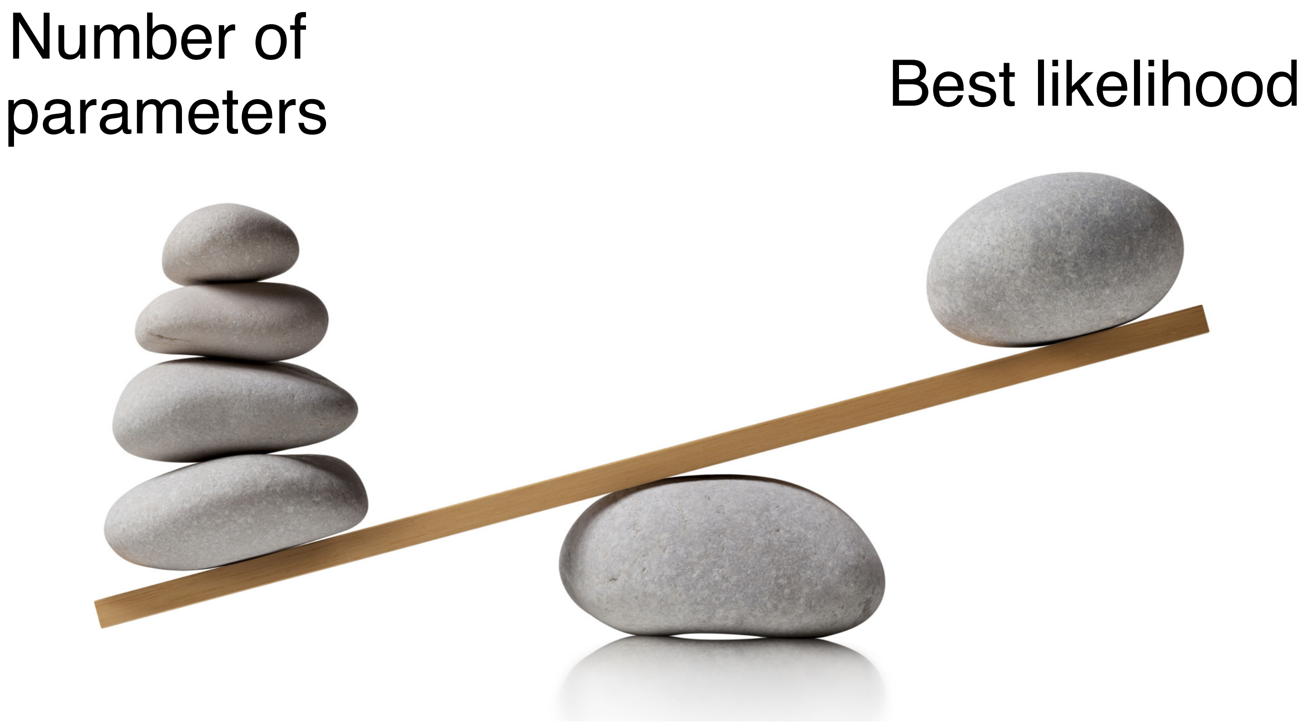
shape
(equal, variable)

Model selection



identifier	Model	HC	EM	Distribution	Volume	Shape	Orientation
E		•	•	(univariate)	equal		
V		•	•	(univariate)	variable		
EII	λI	•	•	Spherical	equal	equal	NA
VII	$\lambda_k I$	•	•	Spherical	variable	equal	NA
EEI	λA		•	Diagonal	equal	equal	coordinate axes
VEI	$\lambda_k A$		•	Diagonal	variable	equal	coordinate axes
EVI	λA_k		•	Diagonal	equal	variable	coordinate axes
VVI	$\lambda_k A_k$		•	Diagonal	variable	variable	coordinate axes
EEE	$\lambda D A D^T$	•	•	Ellipsoidal	equal	equal	equal
EEV	$\lambda D_k A D_k^T$		•	Ellipsoidal	equal	equal	variable
VEV	$\lambda_k D_k A D_k^T$		•	Ellipsoidal	variable	equal	variable
VVV	$\lambda_k D_k A_k D_k^T$	•	•	Ellipsoidal	variable	variable	variable

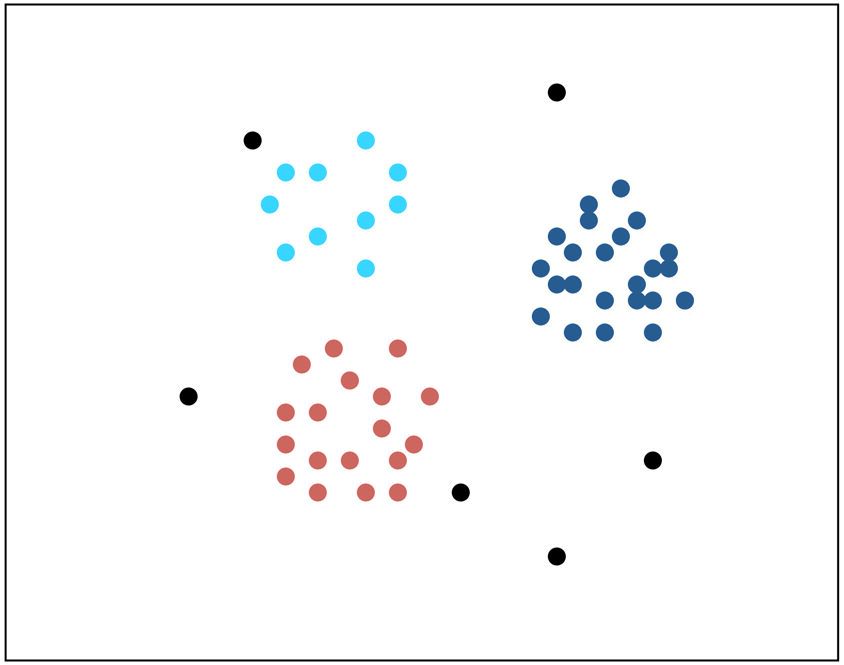
BIC


Number of parameters
Best likelihood

In R

```
>?mclustBIC
>?Mclust


>BIC <- mclustBIC(df)
>plot(BIC)
>summary(BIC)
>mod1 <- Mclust(df, x = BIC)
>summary(mod1, parameters = TRUE)
>plot(mod1, what = "classification")
```

CLAG clustering



In R

```
>library(CLAG)
>CLAG.clust?
```

Challenge

Distance between points of 150 dimensions

1. Create randomly 150 points of 300 dimensions out of a normal distribution with a mean value of 100 and a sd of 10
2. Calculate the Euclidian and City blocks distance between these points
3. Plot the heatmaps for Euclidian and City blocks distance

Challenge

Distance between points of 150 dimensions-continuous

4. Cluster each of the distance matrices using hierarchical clustering (hclust) model using complete agglomeration method and plot the dendorgam

5. Plot the clusters dendogram

6. Repeat 4 and 5 by changing the agglomeration method. Use centroid and median methods.

7. Use the heatmap function by allowing it to automatically classify the data points

8. Can we change the agglomeration method in heatmap call?

Challenge: solution

```
>mat <- matrix(data = rnorm(45000, mean= 100,
sd=10), nrow = 150, ncol = 300)
```

```
>mat.distE<-as.matrix(dist(mat))
>mat.distC<-
as.matrix(dist(mat),method=“manhattan”)
```

```
>heatmap(mat.distE,Colv=NA, Rowv=NA, scale="none")
>heatmap(mat.distC,Colv=NA, Rowv=NA, scale="none”)
```

Challenge: solution

```
>hE<-hclust(distE,”complete")
>hC<-hclust(distC,”complete")

>plot(hE)
>plot(hC)

>hE<-hclust(distE,”centroid")
>hC<-hclust(distC,”centroid")

>plot(hE)
>plot(hC)


>hE<-hclust(distE,”median")
>hC<-hclust(distC,”median")

>plot(hE)
>plot(hC)
```

Challenge: solution

```
>hE<-hclust(distE,”centroid")
>hC<-hclust(distC,”centroid")

>plot(hE)
>plot(hC)


>hE<-hclust(distE,”median")
>hC<-hclust(distC,”median")

>plot(hE)
>plot(hC)

>heatmap(distE)
>heatmap(distC)
```

Challenge: solution-RevoScaleR

```
df<-matrix(rnorm( 300*150, 150, 10), nrow = 150)
head(df)
dist_eu<-as.matrix(dist(df,method = "euclidean"))
head(dist_eu)

dist_man<-as.matrix(dist(df,method= "manhattan"))
heatmap(dist_eu,scale = "none",Rowv = NA,Cowv=NA)
heatmap(dist_man,scale = "none",Rowv = NA,Cowv=NA)

DF <- data.frame(dist_eu)
head(DF)
XDF <- paste(tempfile(), "xdf", sep=".")
if (file.exists(XDF)) file.remove(XDF)
rxDataStep(inData = DF, outFile = XDF)

# Example using an XDF file as a data source
rxKmeans(as.formula(paste("~",paste(names(DF),collapse="+"))),
         data = XDF, numClusters = 3, maxIterations=100)
```

Challenge

Points in plates

1. Import the data from dataClustering.csv
2. What is the dimension of this dataset?
3. How many data point do we have?
4. Evaluate Euclidian distance of points in a plates
5. Classify point to find clusters using hierarchical clustering and the average agglomeration method

Points in plates-continuous

6. We expect to have 3 clusters. When you apply k-means algorithm using 1 iteration, does it differ from applying it using 10 or 100 iterations?

7. Repeat question 6 Using k-means implemented in RevolScaleR

8. What is the outcome of the C-means clustering?

```
install.packages("e1071")
library(e1071)
?cmeans
```

Challenge: solution

```
>library("cluster")
>mydata1<-read.csv(”dataClustering.csv")
>df<-data.frame(mydata1$Coord_X ,mydata1$Coord_Y )
>colnames(df) <- c("X", "Y")
>plot(df$X, df$Y)
```

Challenge: solution

```
>library("cluster")
>mydata1<-read.csv(”dataClustering.csv")
>df<-data.frame(mydata1$Coord_X ,mydata1$Coord_Y )
>colnames(df) <- c("X", "Y")
>plot(df$X, df$Y)

#evaluate Euclidian distance
>df.dist<-dist(df)
# classify
>df.h<-hclust(df.dist,"ave")
>plot(df.h)

>colorScale <- colorRampPalette(c("blue",
"green","yellow","red","darkred"))(1000)
>heatmap(as.matrix(df.dist),Colv=NA, Rowv=NA,
scale="none”, col=colorScale)
```

Challenge: solution

```
>kmeans(df,3)

>cl.1 <- kmeans(df, 3, iter.max = 1)
>plot(df, col = cl.1$cluster)
>points(cl.1$centers, col = 1:5, pch = 8)
```

Challenge: solution

```
>kmeans(df,3)

>cl.1 <- kmeans(df, 3, iter.max = 1)
>plot(df, col = cl.1$cluster)
>points(cl.1$centers, col = 1:5, pch = 8)

>cl.10 <- kmeans(df, 3, iter.max = 10)
>plot(df, col = cl.10$cluster)
>points(cl.10$centers, col = 1:5, pch = 8)

>cl.100 <- kmeans(df, 3, iter.max = 100)
>plot(df, col = cl.100$cluster)
>points(cl.100$centers, col = 1:5, pch = 8)
```

Challenge: rxKmeans

```
#Step 1: Prep and Import Data
#Initialize some variables to specify the data sets.
   inputFileData <- paste0("/media/sf_docVM/",
"dataClustering.csv")

#Import the  data.
clustering_data<- rxImport(inData = inputFileData)

#run kmeans
z<-rxKmeans(~ Coord_X + Coord_Y, data =
clustering_data, numClusters = 3, maxIterations=100)

#plot outcome
DF <-
data.frame(clustering_data$Coord_X,clustering_data$Coor
d_Y)
plot(DF, col = z$cluster)
points(z$centers, col = 1:5, pch = 8)
```

Challenge: solution

```
>library(e1071)
>cmeans(df,3)

>cl.1 <- cmeans(df, 3, iter.max = 1)
>plot(df, col = cl.1$cluster)
>points(cl.1$centers, col = 1:5, pch = 8)

>cl.10 <- cmeans(df, 3, iter.max = 10)
>plot(df, col = cl.10$cluster)
>points(cl.10$centers, col = 1:5, pch = 8)

>cl.100 <- cmeans(df, 3, iter.max = 100)
>plot(df, col = cl.100$cluster)
>points(cl.100$centers, col = 1:5, pch = 8)
```

Challenge

Points in plates-continuous

Library(mclust)

7. What are the top 3 models *mclustBIC* function suggests based on the BIC criterion?

8. How many clusters did it find using the top model?

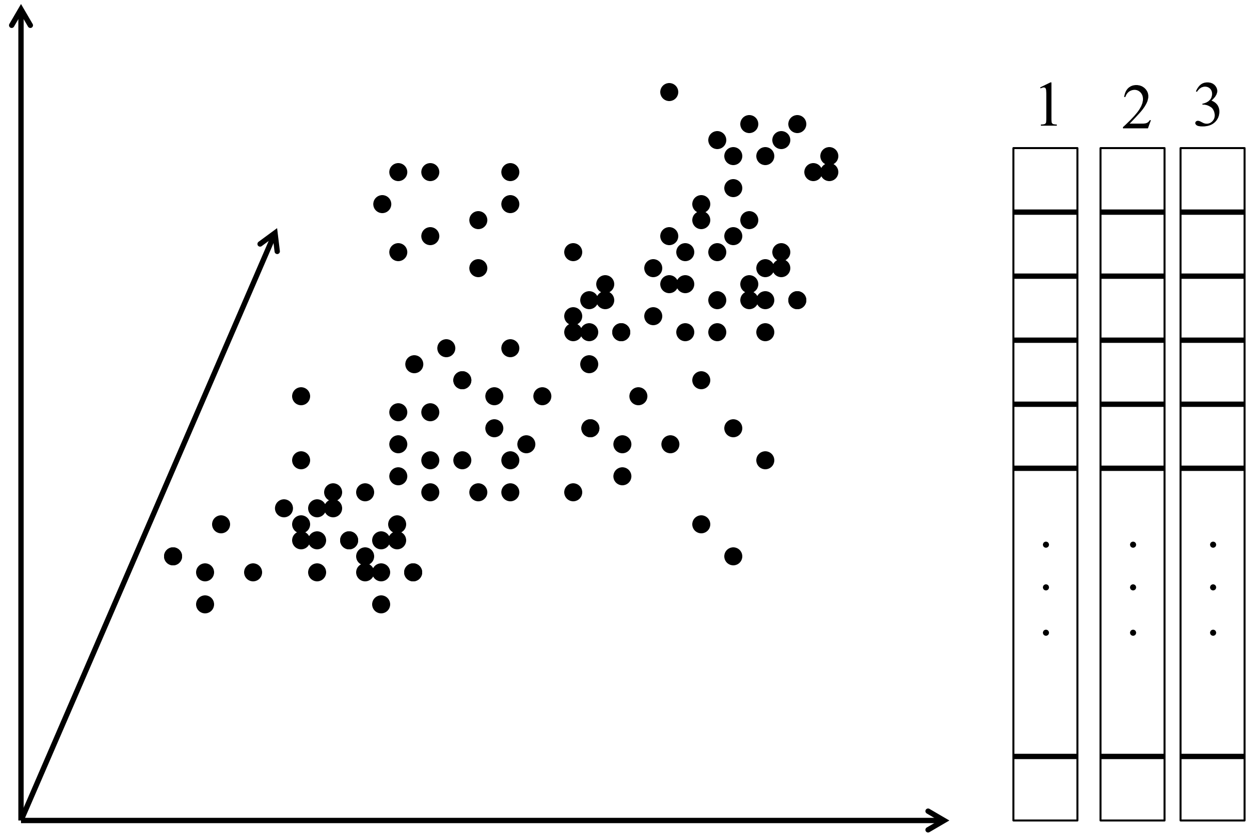
10. Plot the outcome

Challenge: solution

```
>library("mclust")
>BIC <- mclustBIC(df)
>plot(BIC)
>summary(BIC)
>mod1 <- Mclust(df, x = BIC)
>summary(mod1, parameters = TRUE)
>plot(mod1, what = "classification")
```

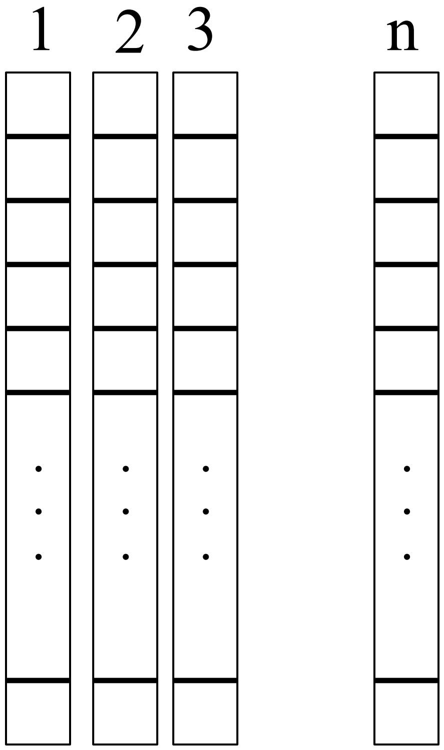
Dimension representation

3-dimension



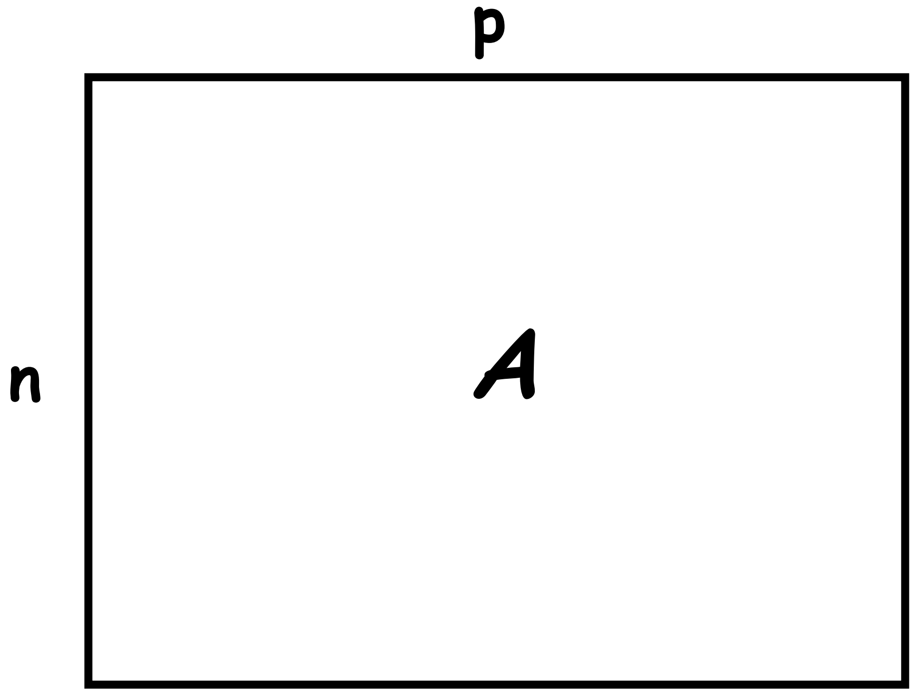
n-dimension

?

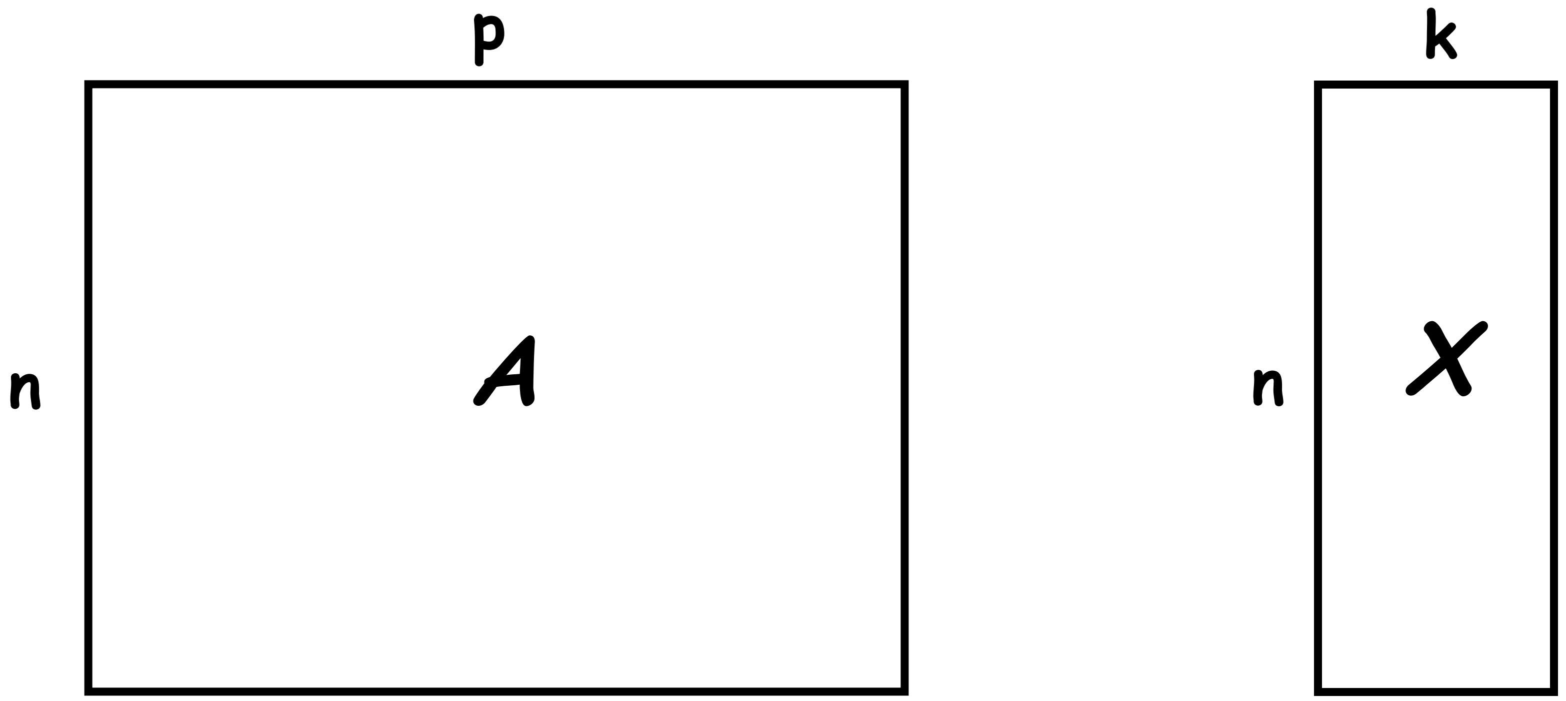


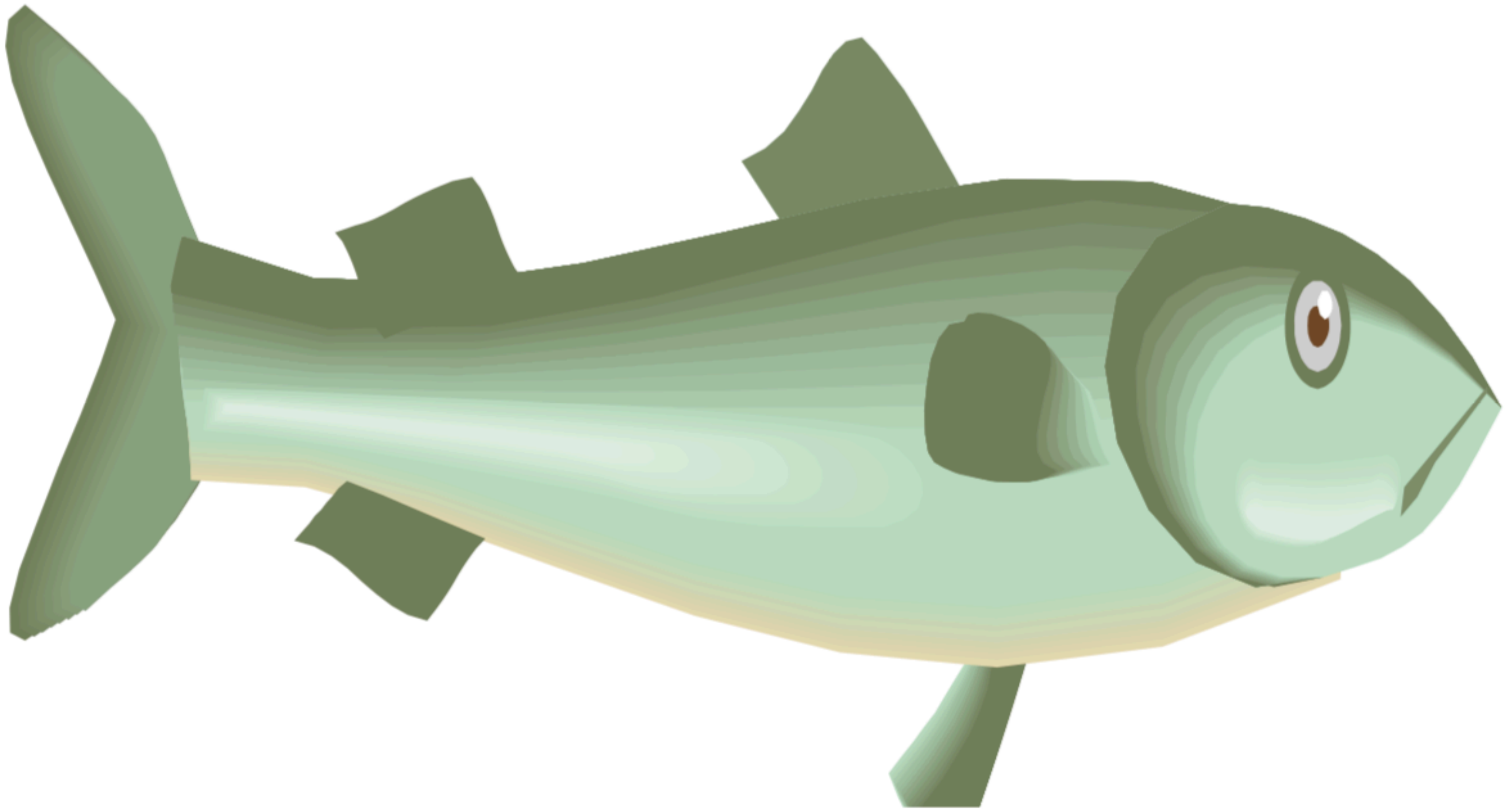
Principal Component Analysis (PCA)

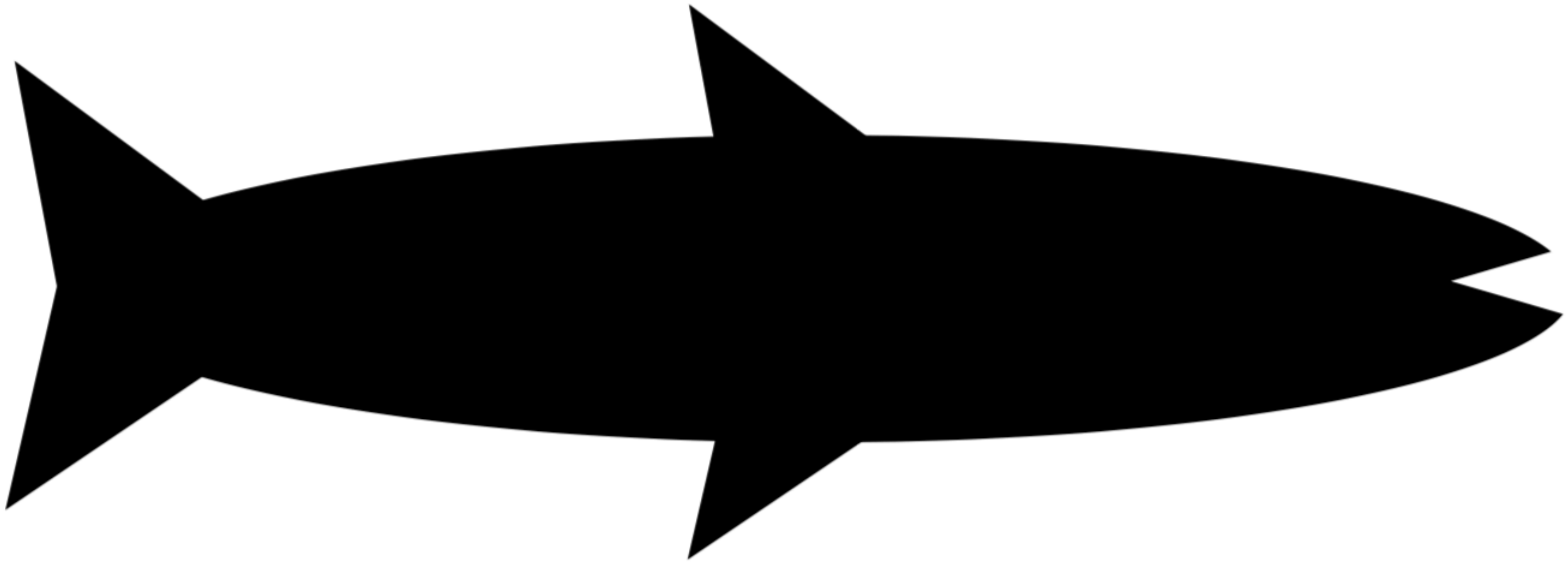
Pearson (1901) and Hotelling (1933)

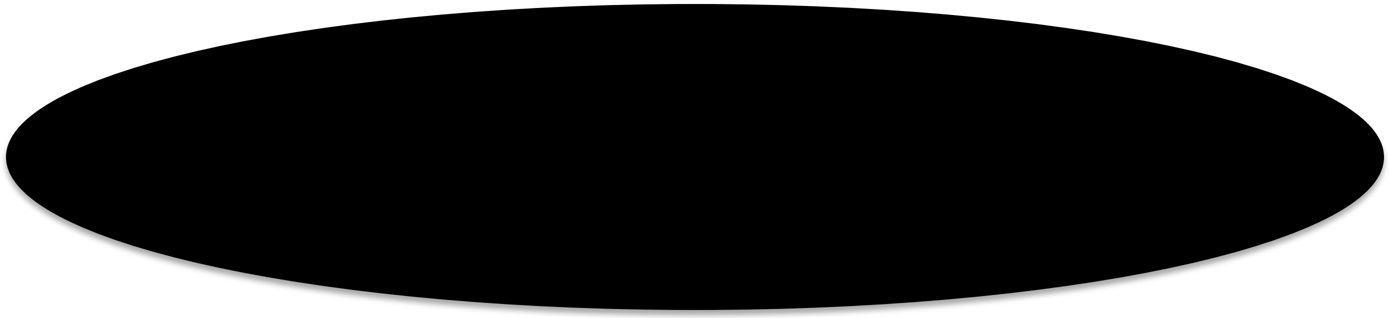

p
n
A

Data Reduction


p
n
A
k
n
X







clarity of
representation

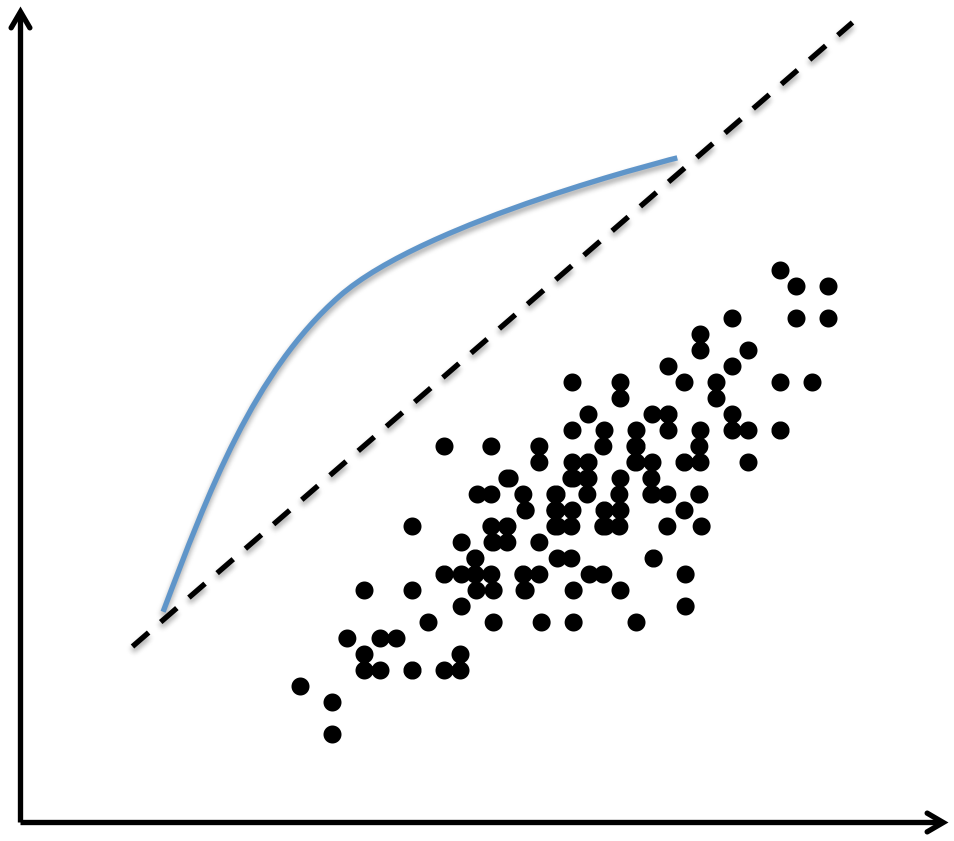
Over-simplification

PCA is based on variance

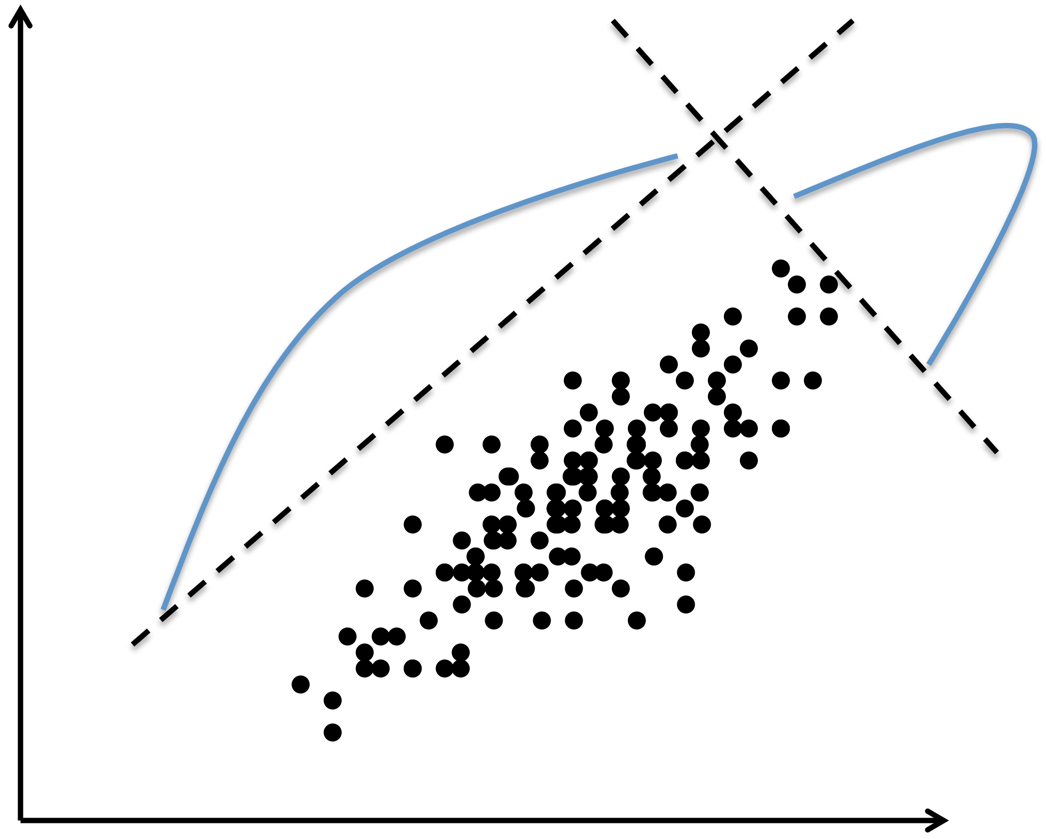
PCA = Rotate axis

Which and how?

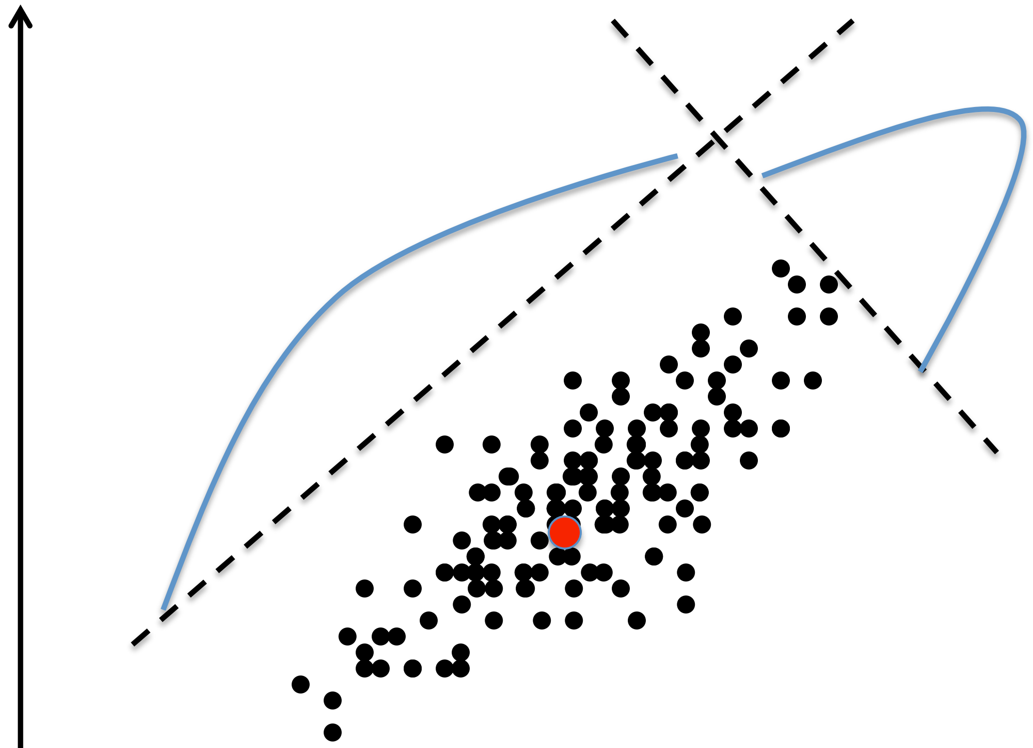
1. Largest variance first

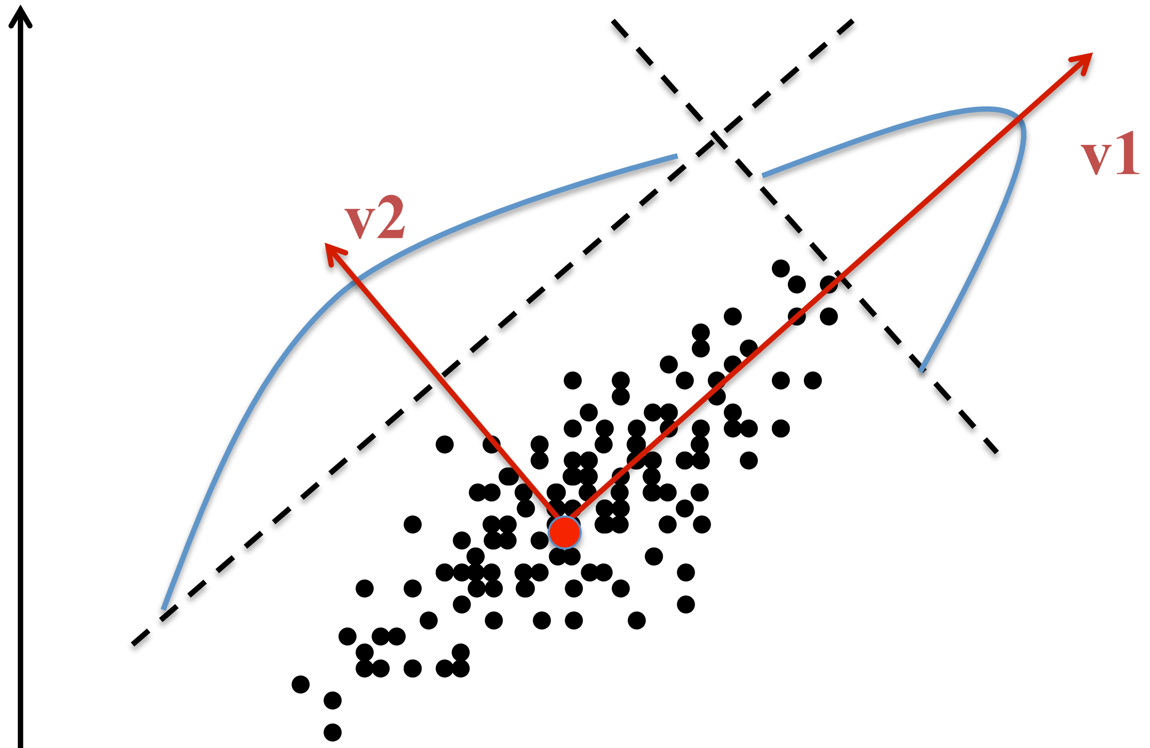


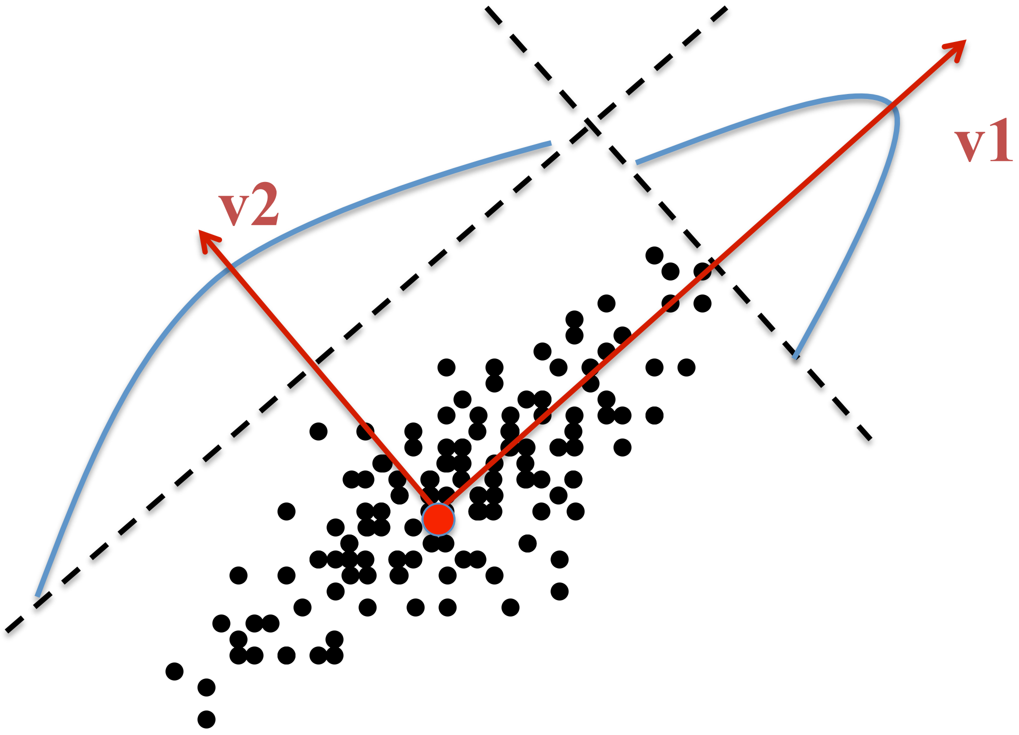
2. Select uncorrelated principal axis (orthogonal)

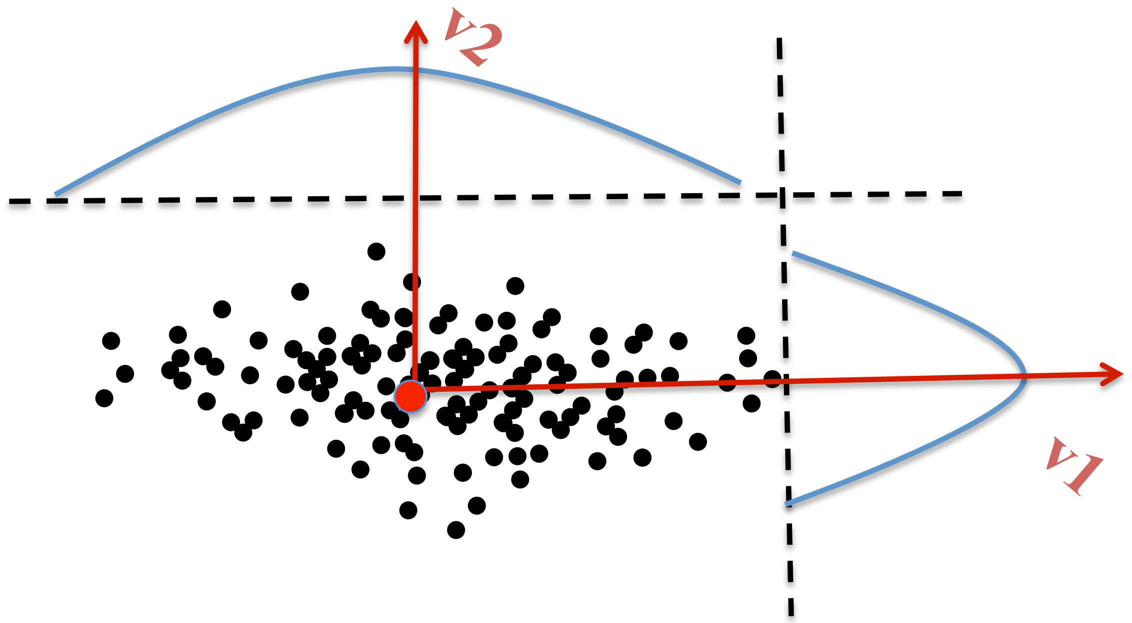


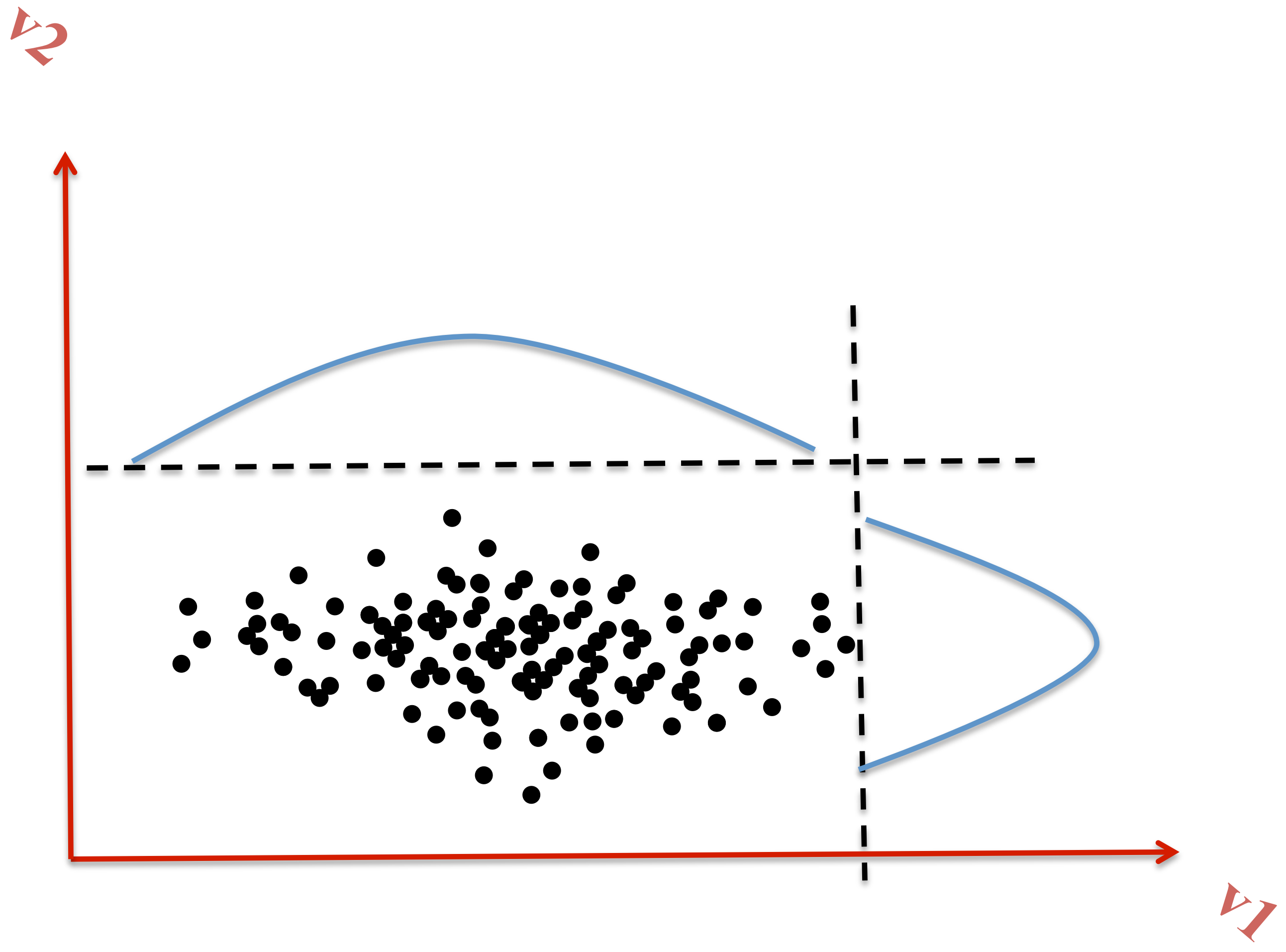
centroid




v2
v1


v2
v1


v2
v1



Without centroid

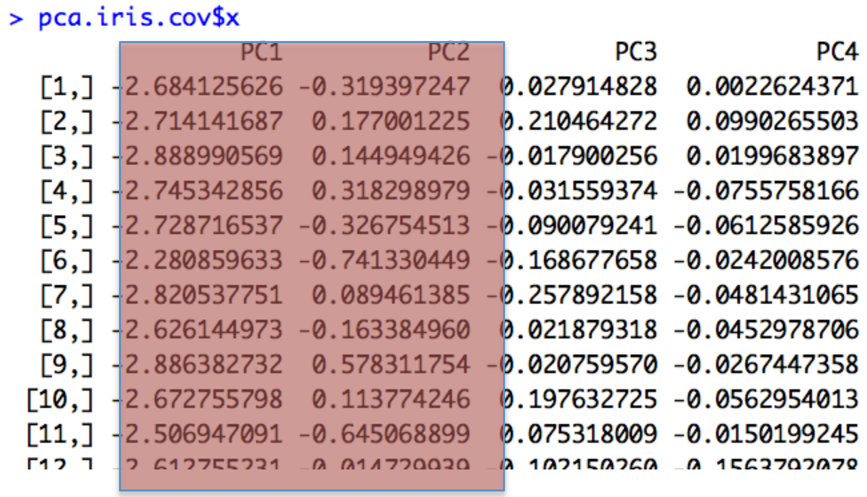
In R

```
>pca<-prcomp(data, center = TRUE, scale. = FALSE)
#coordinate of sample on components were identified

#Importance of components
>summary(pca)
```

In R

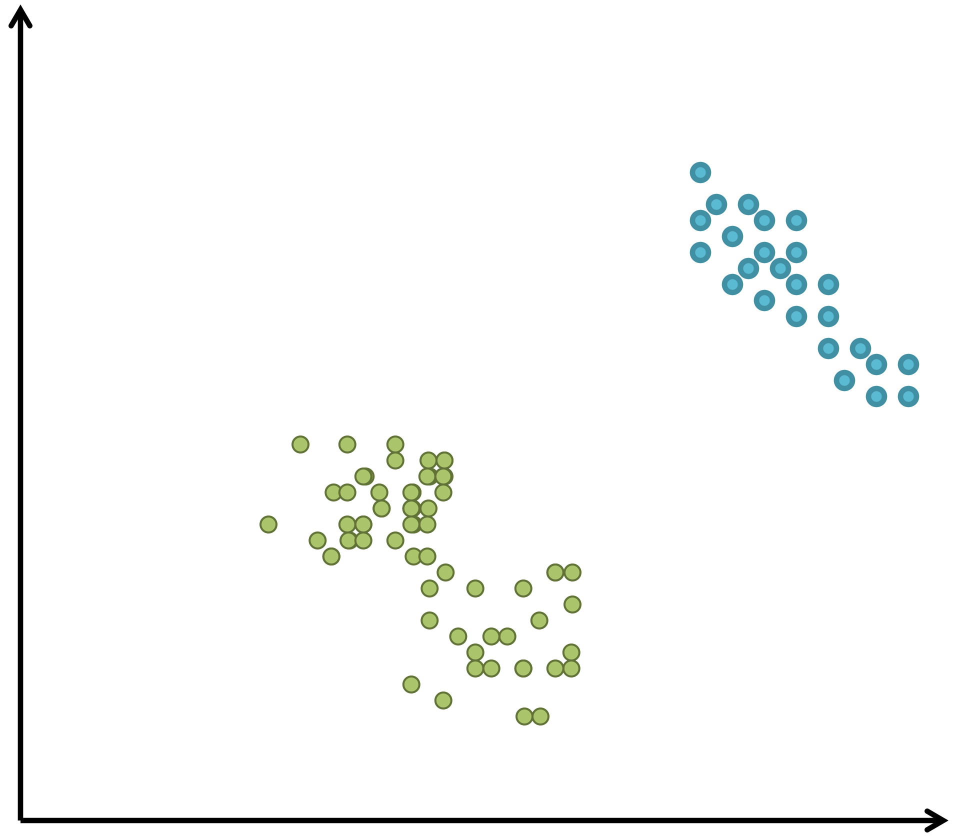
```
>pca<-prcomp(data, center = TRUE, scale. = FALSE)
#coordinate of sample on components were identified
```

```
#Importance of components
>summary(pca)

>pca$x
>plot(pca$x)
```

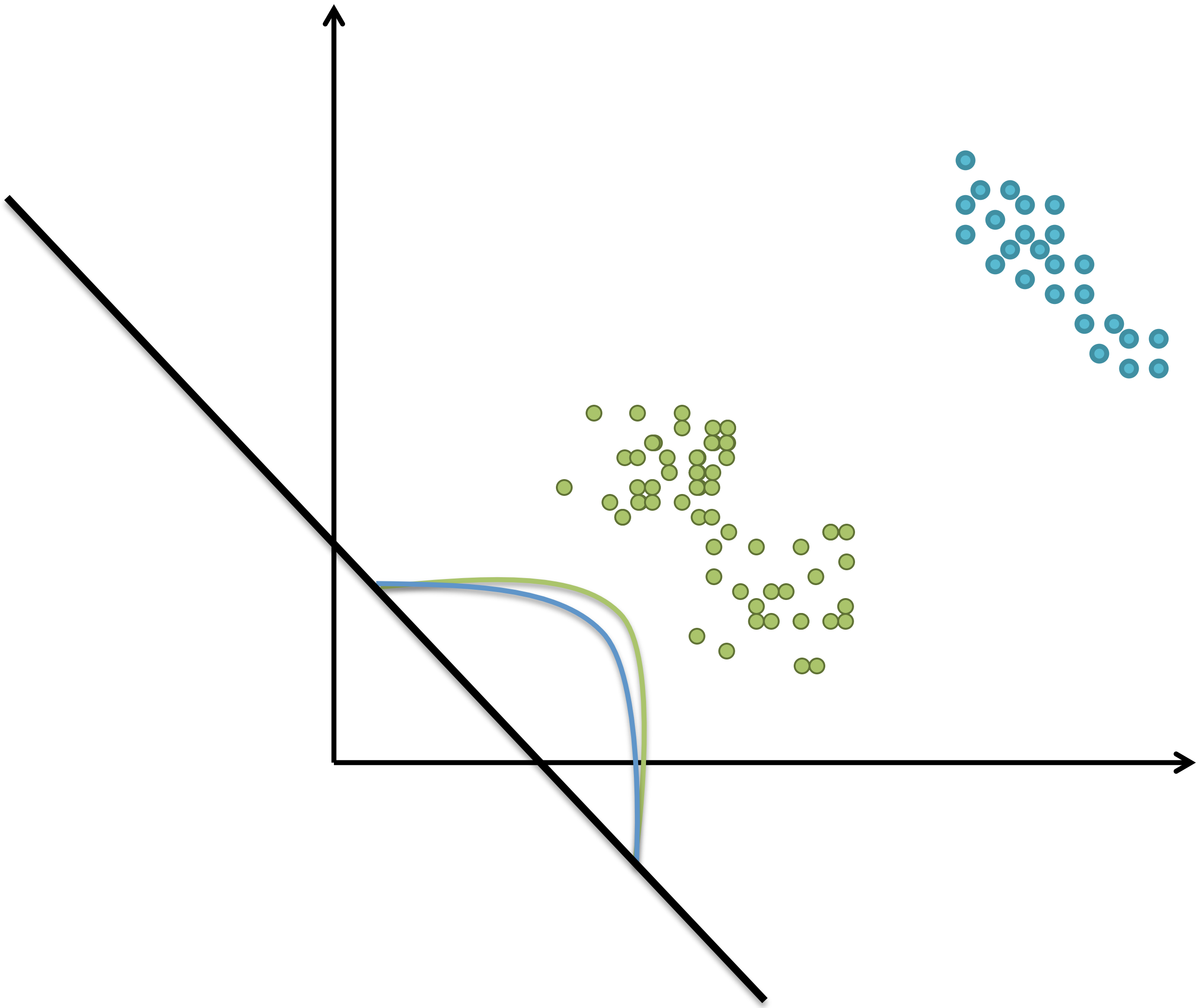


```
> pca.iris.cov$x
              PC1          PC2          PC3           PC4
 [1,] -2.684125626 -0.319397247  0.027914828  0.0022624371
 [2,] -2.714141687  0.177001225  0.210464272  0.0990265503
 [3,] -2.888990569  0.144949426 -0.017900256  0.0199683897
 [4,] -2.745342856  0.318298979 -0.031559374 -0.0755758166
 [5,] -2.728716537 -0.326754513 -0.090079241 -0.0612585926
 [6,] -2.280859633 -0.741330449 -0.168677658 -0.0242008576
 [7,] -2.820537751  0.089461385 -0.257892158 -0.0481431065
 [8,] -2.626144973 -0.163384960  0.021879318 -0.0452978706
 [9,] -2.886382732  0.578311754 -0.020759570 -0.0267447358
[10,] -2.672755798  0.113774246  0.197632725 -0.0562954013
[11,] -2.506947091 -0.645068899  0.075318009 -0.0150199245
[12,] -2.612755231 -0.014729939 -0.102150260 -0.1563792078
```

LDA

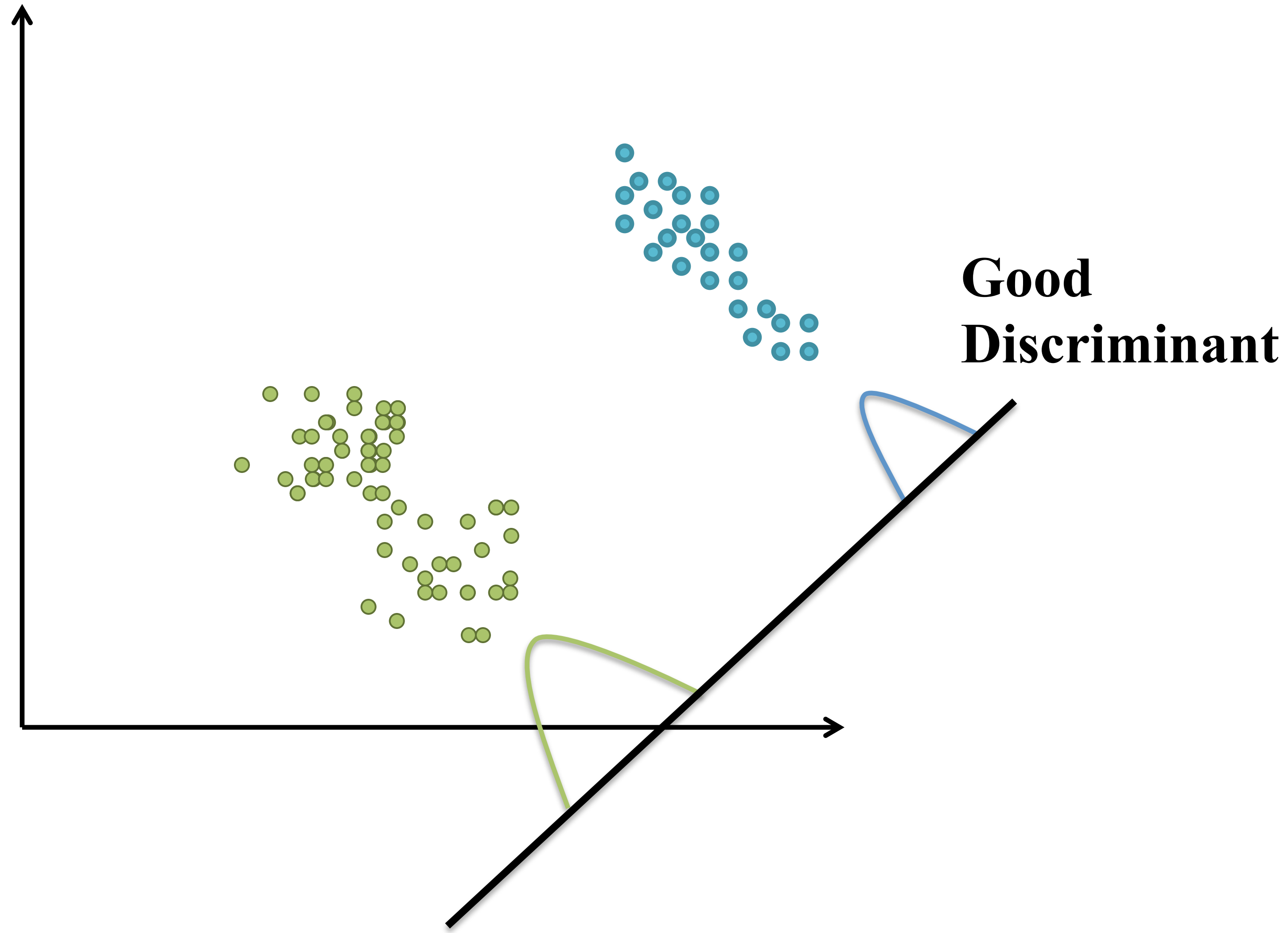


LDA



Bad Discriminant

LDA



```
library(MASS)
data(iris)
head(iris, 3)
train <- sample(1:150, 75)
r <- lda(formula = Species ~ .,
         data = iris,
         prior = c(1,1,1)/3,
         subset = train)
r$prior
r$counts
#means for each covariate
r$means
#with 3 classes we have at most two linear discriminants
r$scaling
#the singular values (svd) that gives the ratio of the between-
 and within-group standard deviations on the linear discriminan
 variables.
r$svd
# amount of the between-group variance that is explained by eac
 linear discriminant
prop = r$svd^2/sum(r$svd^2)
head(r2$class)
head(r2$posterior, 3)
plda = predict(object = r, newdata = iris[-train, ])
```

```
library(MASS)
data(iris)
head(iris, 3)
train <- sample(1:150, 75)
r <- lda(formula = Species ~ .,
         data = iris,
         prior = c(1,1,1)/3,
         subset = train)
r$prior
r$counts
#means for each covariate
r$means
#with 3 classes we have at most two linear discriminants
r$scaling
#the singular values (svd) that gives the ratio of the between-
 and within-group standard deviations on the linear discriminar
 variables.
r$svd
# amount of the between-group variance that is explained by eac
 linear discriminant
prop = r$svd^2/sum(r$svd^2)
head(r2$class)
head(r2$posterior, 3)
plda = predict(object = r, newdata = iris[-train, ])
```

```
library(MASS)
data(iris)
head(iris, 3)
train <- sample(1:150, 75)
r <- lda(formula = Species ~ .,
         data = iris,
         prior = c(1,1,1)/3,
         subset = train)
r$prior
r$counts
#means for each covariate
r$means
#with 3 classes we have at most two linear discriminants
r$scaling
#the singular values (svd) that gives the ratio of the between-
 and within-group standard deviations on the linear discriminar
 variables.
r$svd
# amount of the between-group variance that is explained by eac
 linear discriminant
prop = r$svd^2/sum(r$svd^2)
head(r2$class)
head(r2$posterior, 3)
plda = predict(object = r, newdata = iris[-train, ])
```

```
library(MASS)
data(iris)
head(iris, 3)
train <- sample(1:150, 75)
r <- lda(formula = Species ~ .,
         data = iris,
         prior = c(1,1,1)/3,
         subset = train)
r$prior
r$counts
#means for each covariate
r$means
#with 3 classes we have at most two linear discriminants
r$scaling
#the singular values (svd) that gives the ratio of the between
- and within-group standard deviations on the linear discrimina
-  variables.
r$svd
# amount of the between-group variance that is explained by eac
 linear discriminant
prop = r$svd^2/sum(r$svd^2)
head(r2$class)
head(r2$posterior, 3)
plda = predict(object = r, newdata = iris[-train, ])
```

```
library(MASS)
data(iris)
head(iris, 3)
train <- sample(1:150, 75)
r <- lda(formula = Species ~ .,
         data = iris,
         prior = c(1,1,1)/3,
         subset = train)
r$prior
r$counts
#means for each covariate
r$means
#with 3 classes we have at most two linear discriminants
r$scaling
#the singular values (svd) that gives the ratio of the between-
 and within-group standard deviations on the linear discriminan
 variables.
r$svd
# amount of the between-group variance that is explained by eac
prop = r$svd^2/sum(r$svd^2)
head(r2$class)
head(r2$posterior, 3)
plda = predict(object = r, # predictions
               newdata = iris[-train, ])
```

```
library(MASS)
data(iris)
head(iris, 3)
train <- sample(1:150, 75)
r <- lda(formula = Species ~ .,
         data = iris,
         prior = c(1,1,1)/3,
         subset = train)
r$prior
r$counts
#means for each covariate
r$means
#with 3 classes we have at most two linear discriminants
r$scaling
#the singular values (svd) that gives the ratio of the between-
 and within-group standard deviations on the linear discrimina
 variables.
r$svd
# amount of the between-group variance that is explained by
each linear discriminant
prop = r$svd^2/sum(r$svd^2)
head(r2$class)
head(r2$posterior, 3)
plda = predict(object = r, newdata = iris[-train, ])
```

```
library(MASS)
data(iris)
head(iris, 3)
train <- sample(1:150, 75)
r <- lda(formula = Species ~ .,
         data = iris,
         prior = c(1,1,1)/3,
         subset = train)
r$prior
r$counts
#means for each covariate
r$means
#with 3 classes we have at most two linear discriminants
r$scaling
#the singular values (svd) that gives the ratio of the between-
 and within-group standard deviations on the linear discriminar
 variables.
r$svd
# amount of the between-group variance that is explained by eac
 linear discriminant
prop = r$svd^2/sum(r$svd^2)
head(r2$class)
head(r2$posterior, 3)
plda = predict(object = r, newdata = iris[-train, ])
```

Thank you for your attention