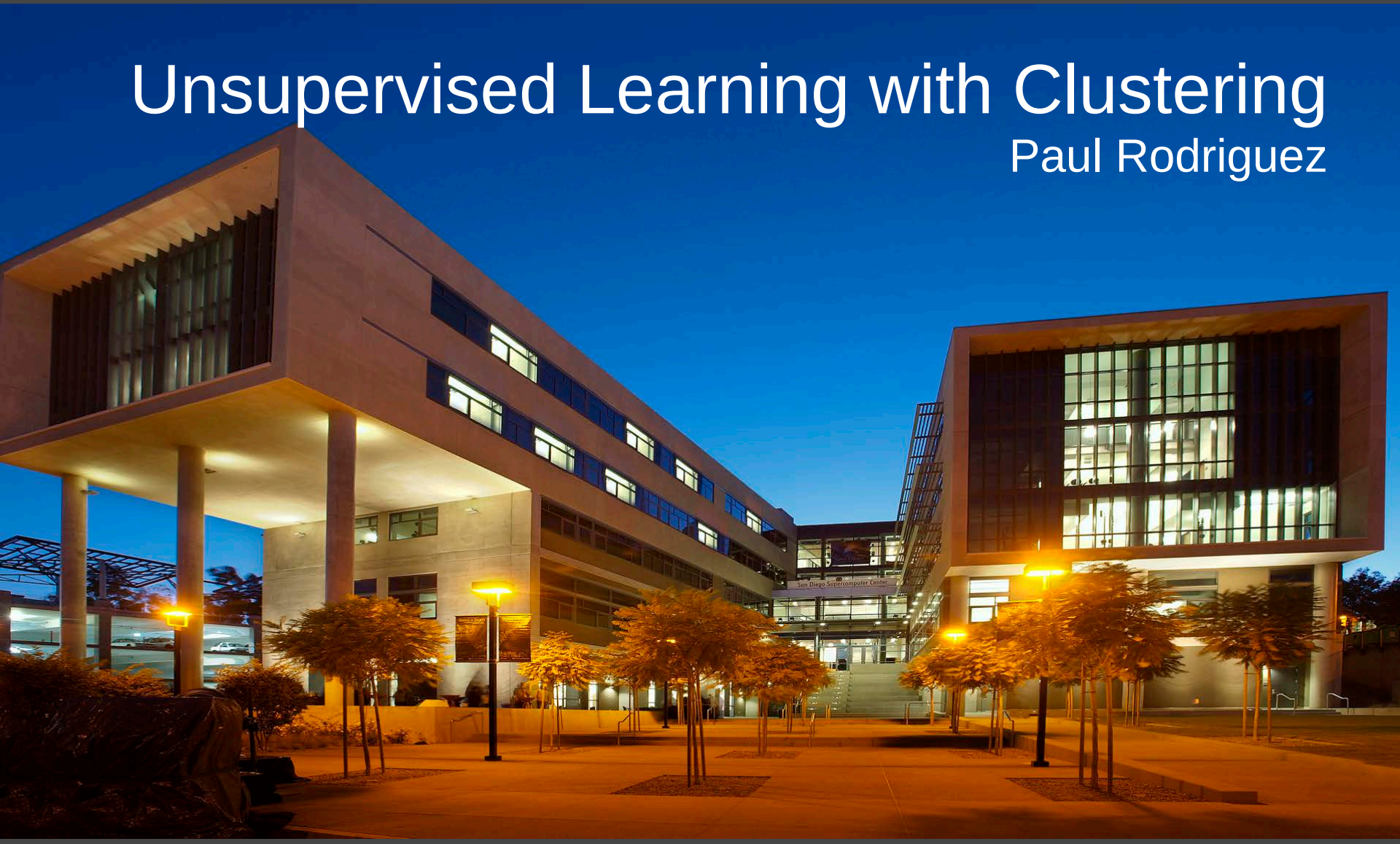
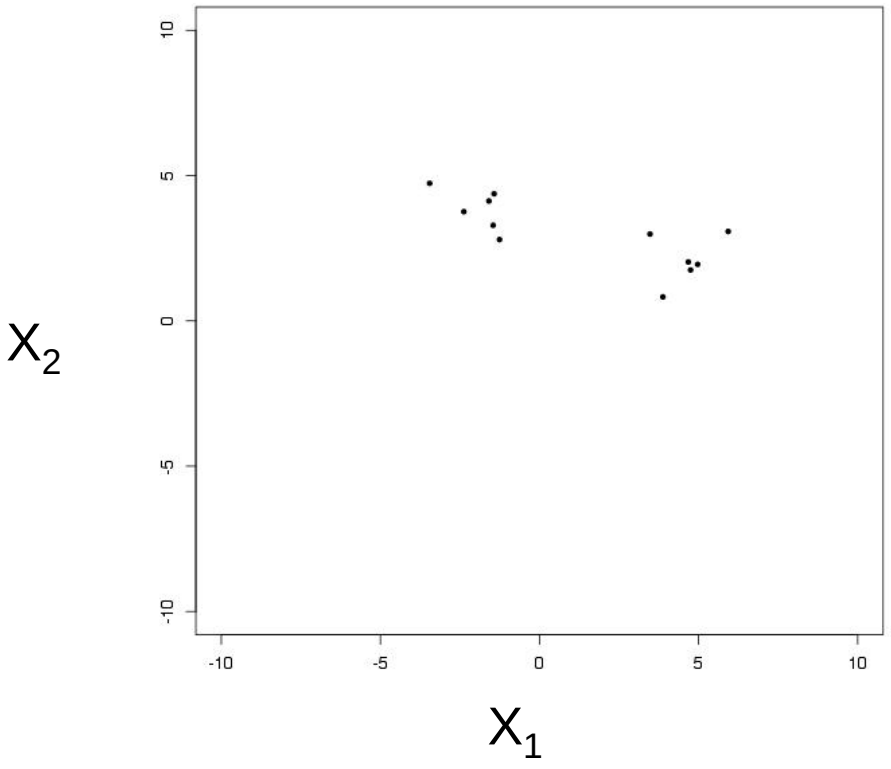
Unsupervised Learning with Clustering

Paul Rodriguez



Clustering Idea

- **Given a set of data can we find a natural grouping?**



Essential R commands:

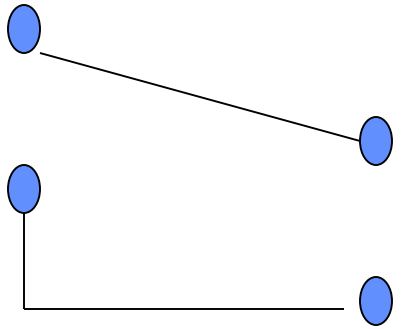
```
D     =rnorm(12,0,1) #generate 12
                     #random normal
X1    =matrix(D,6,2) #put into 6x2 matrix
X1[,1]=X1[,1]+4;     #shift center
X1[,2]=X1[,2]+2;

#repeat for another set of points

#bind data points and plot
plot(rbind(X1,X2),
          xlim=c(-10,10),ylim=c(-10,10));
```

Why Clustering

- **A good grouping implies some structure**
- **In other words, given a good grouping, we can then:**
  - Interpret and label clusters
  - Identify important features
  - Characterize new points by the closest cluster (or nearest neighbors)
  - Use the cluster assignments as a compression or summary of the data

Clustering Objective

- **Objective: find subsets that are similar within cluster and dissimilar between clusters**
- **Similarity defined by distance measures**
  - Euclidean distance
  - Manhattan distance



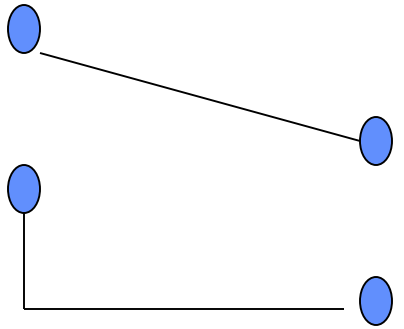
Clustering Objective

- **Objective: find subsets that are similar within cluster and dissimilar between clusters**
- **Similarity defined by distance measures**

  - Euclidean distance =

    $\text{sqrt}[(a1 - b1)^2 + (a2 - b2)^2 + \ldots)]$
  - Manhattan distance

    $[|a1 - b1| + |a2 - b2| + \ldots)]$

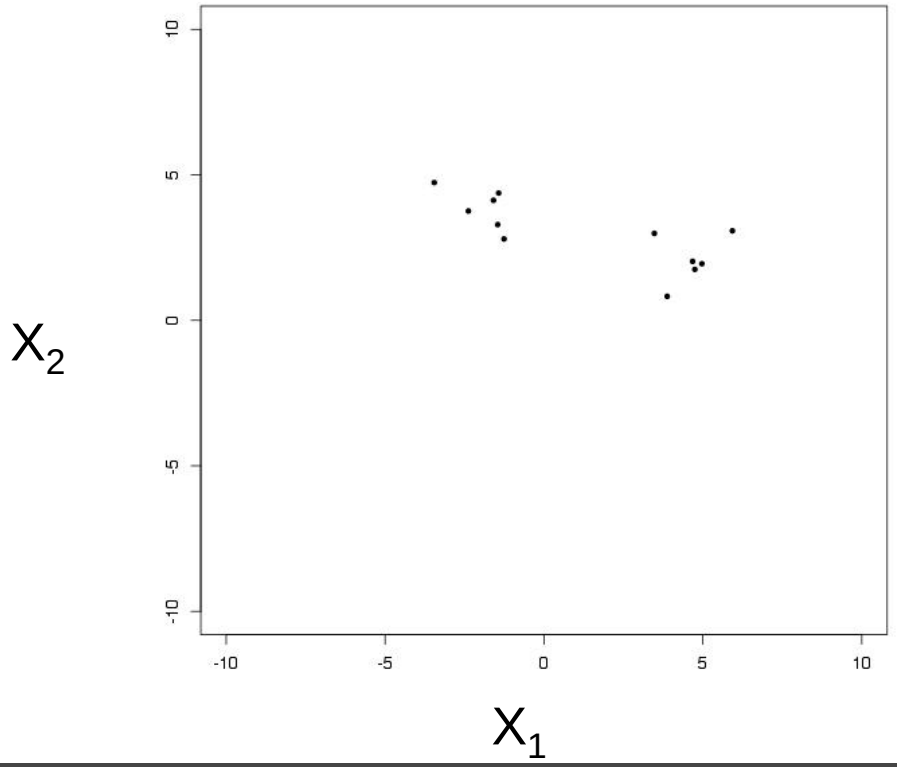


Kmeans Clustering

- **A simple, effective, and standard method**
  - Start with K initial cluster centers
  - Loop:
    - Assign each data point to nearest cluster center
    - Calculate mean of cluster for new center
  - Stop when assignments don't change
- **Issues:**
  - How to choose K?
  - How to choose initial centers?
  - Will it always stop?

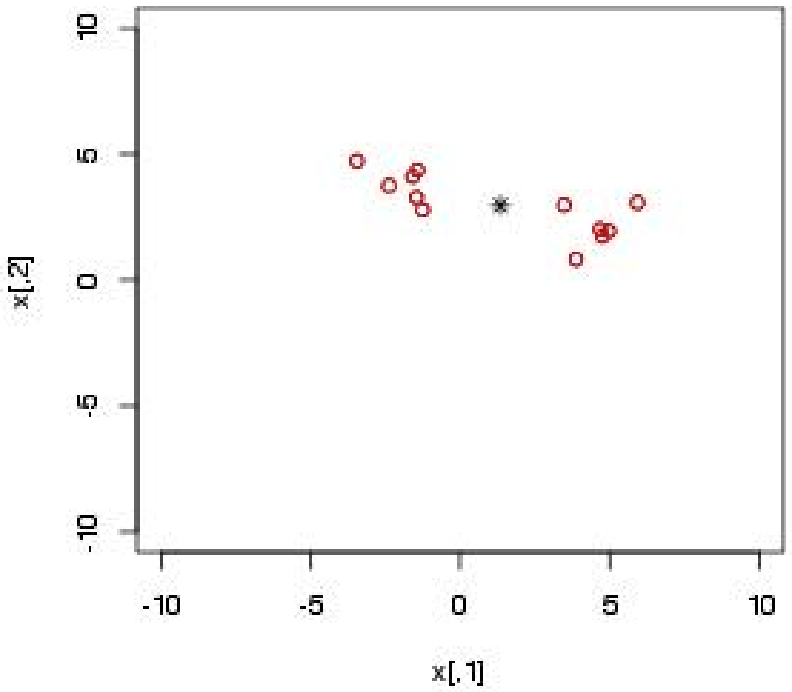
Kmeans Example

- **For K=1, using Euclidean distance, where will the cluster center be?**



Kmeans Example

- **For K=1, the overall mean minimizes Sum Squared Error (SSE), aka Euclidean distance**

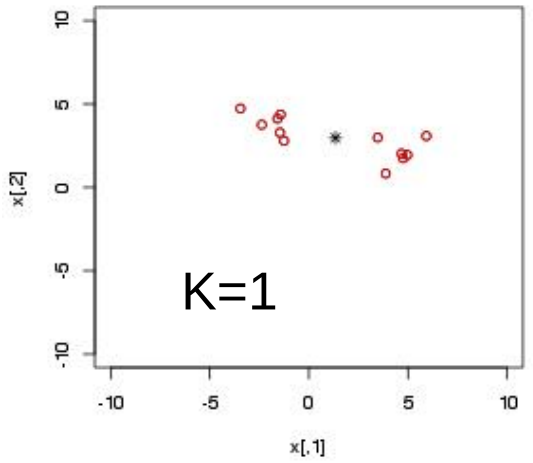


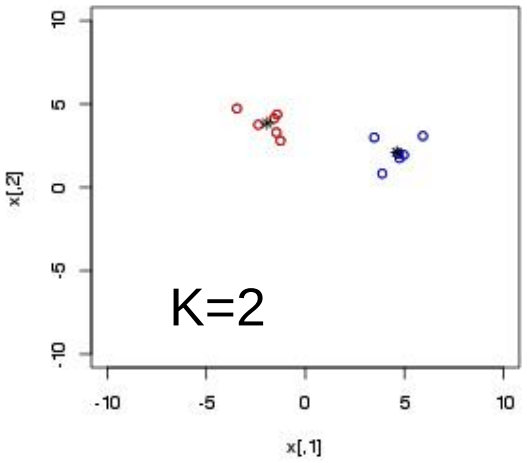
Essential R commands:
Kresult = kmeans(X,1,10,1)

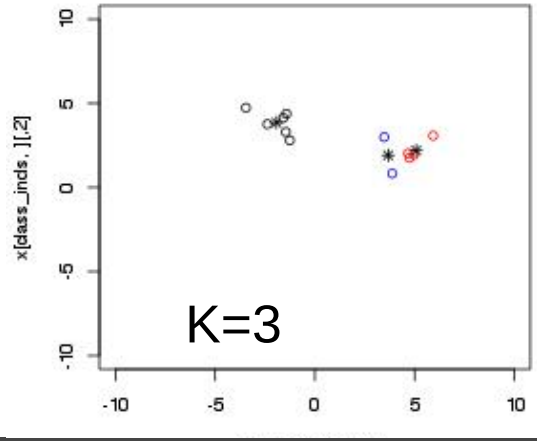
#choose 1 data point as initial K centers
#10 is max loop iterations
#1 is number of initial sets to try

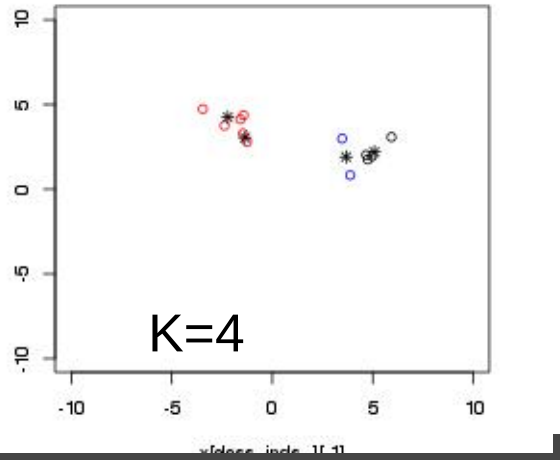
#Kresult is an R object with subfields
Kresult$cluster #cluster assignments
Kresults$tot.withinss # tot within SSE

Kmeans Example







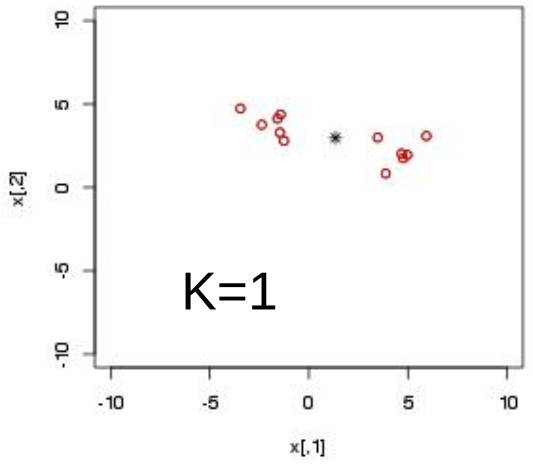


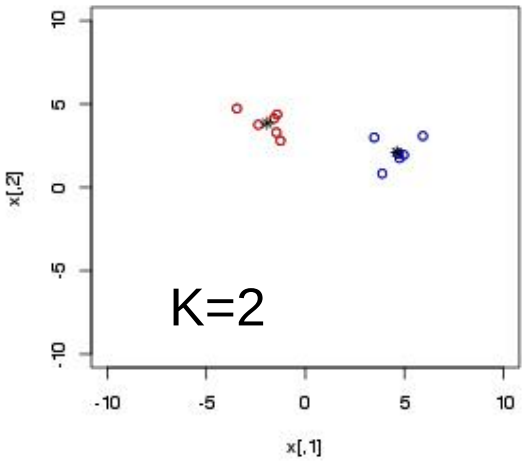
Essential R commands:
inds=which(Kresult$cluster==K)

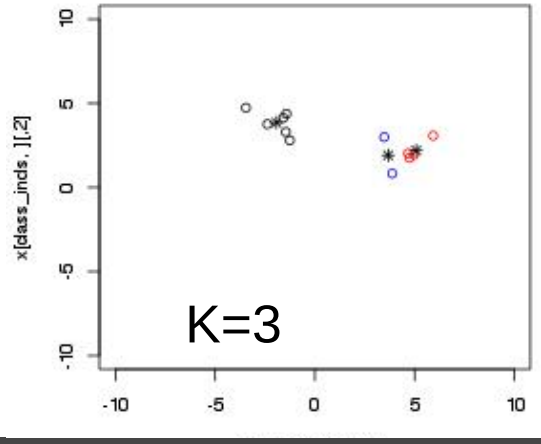
plot(X[inds,],col2use="red");

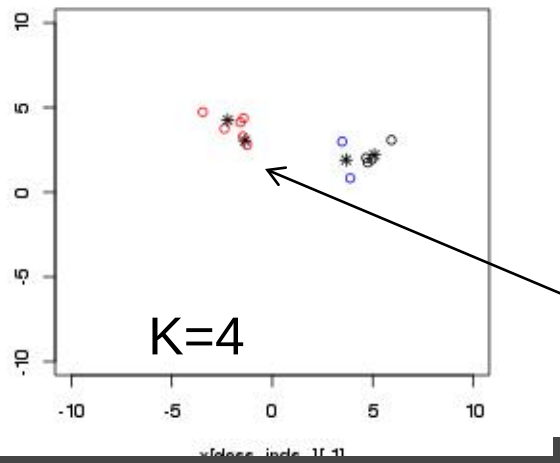
…

Kmeans Example









Essential R commands:
inds=which(Kresult$cluster==K)

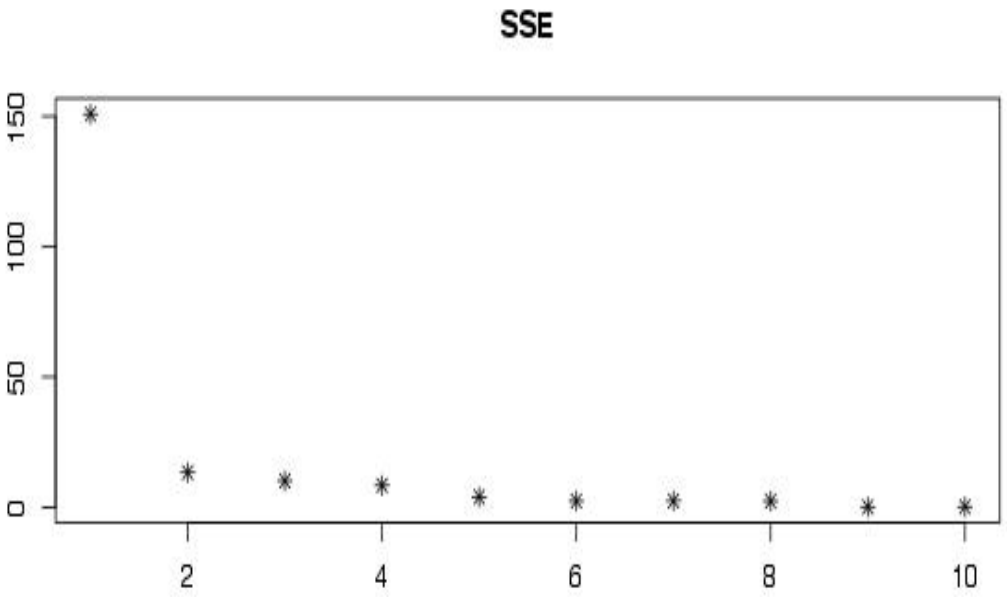
plot(X[inds,],col2use="red");

…

As K increases individual points get a cluster

Choosing K for Kmeans

Total Within Cluster SSE



K=1 to 10

Essential R commands:

```
for (num_k in 1:10) {
Kres=kmeans(X,num_k,10,1);
```

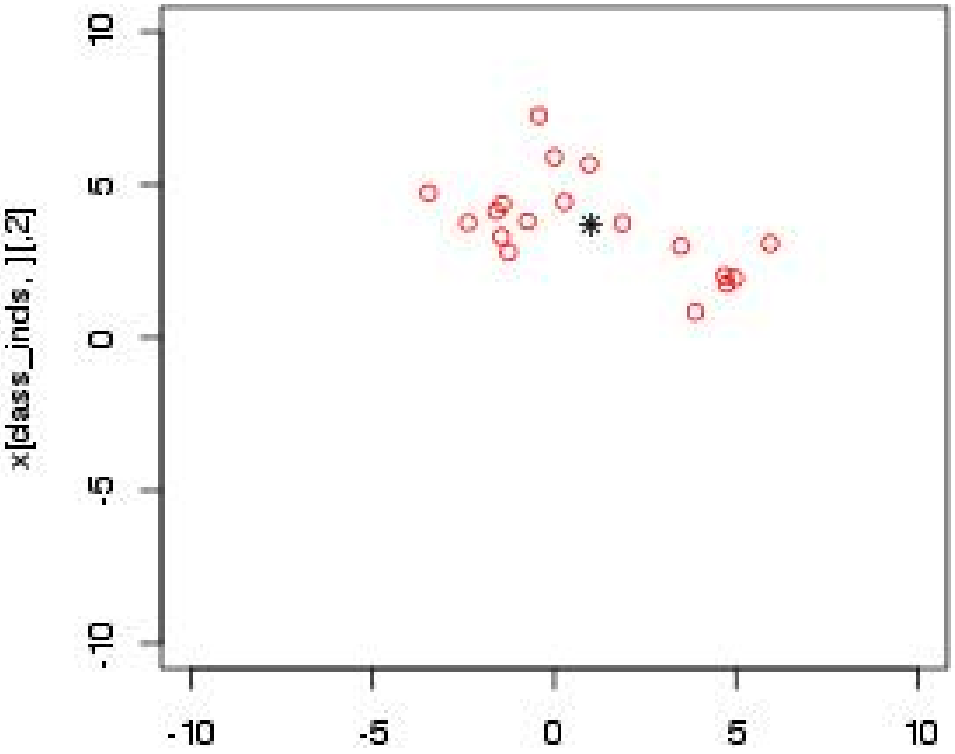
Save and then plot
Kres$tot.withinss

…

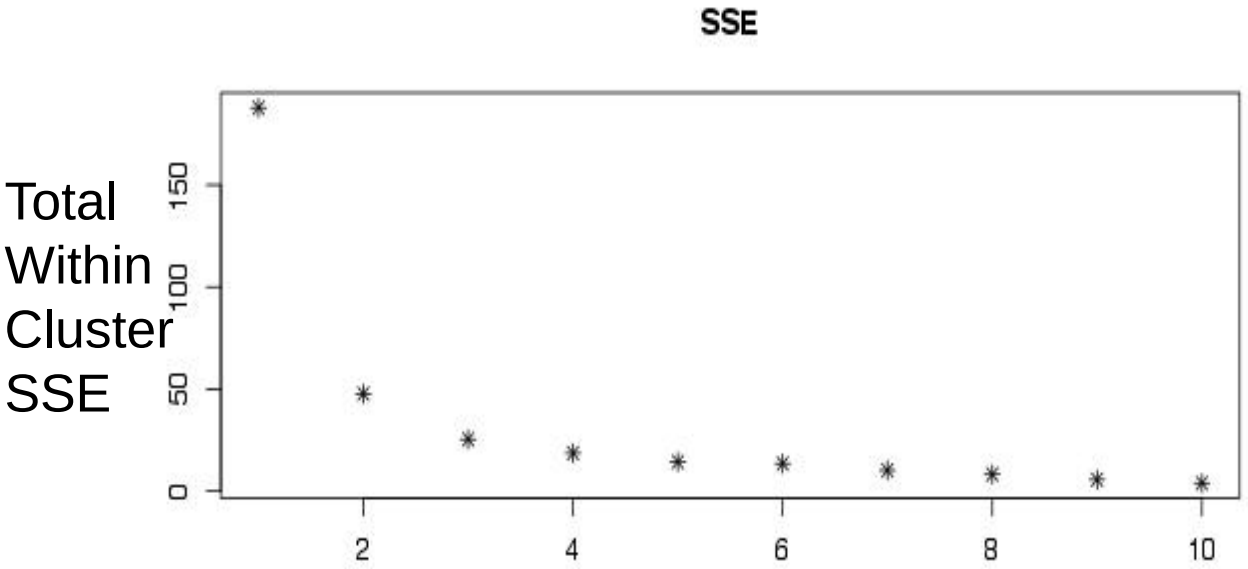
- Not much improvement after K=2 (“elbow”)

Kmeans Example – more points

How many clusters should there be?



Choosing K for Kmeans



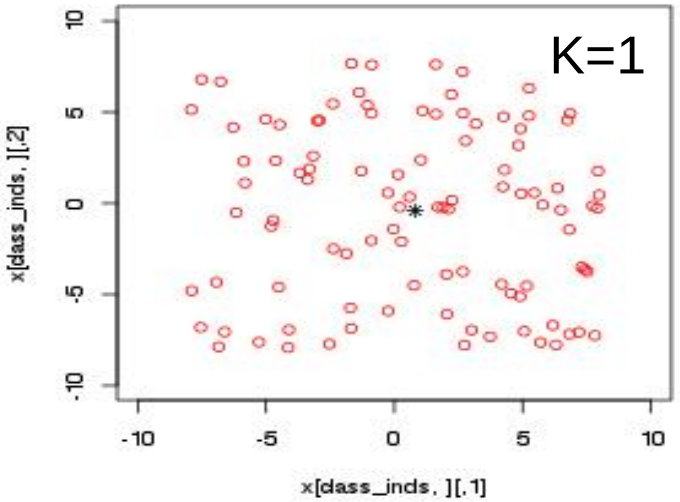
K=1 to 10

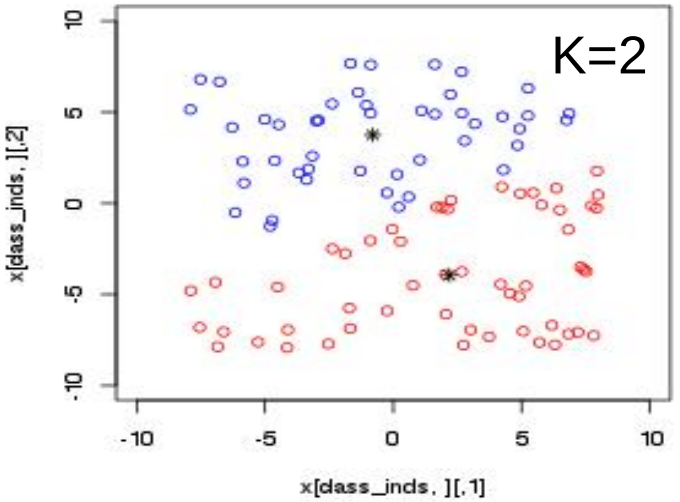
- Smooth decrease at K ≥ 2, harder to choose
- In general, smoother decrease => less structure

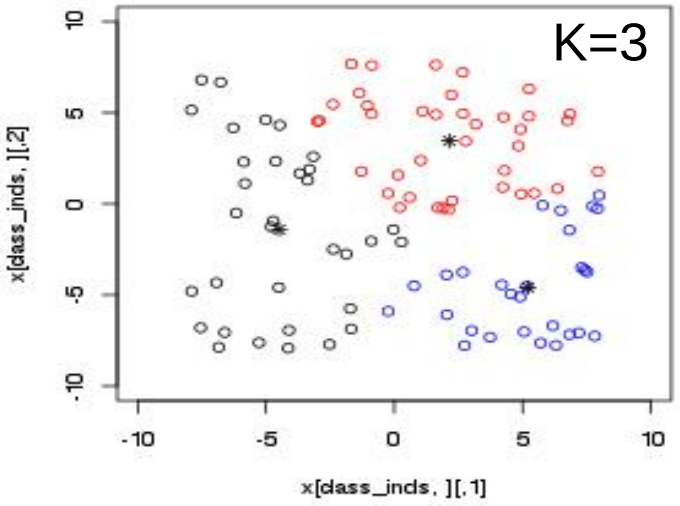
Kmeans Guidelines

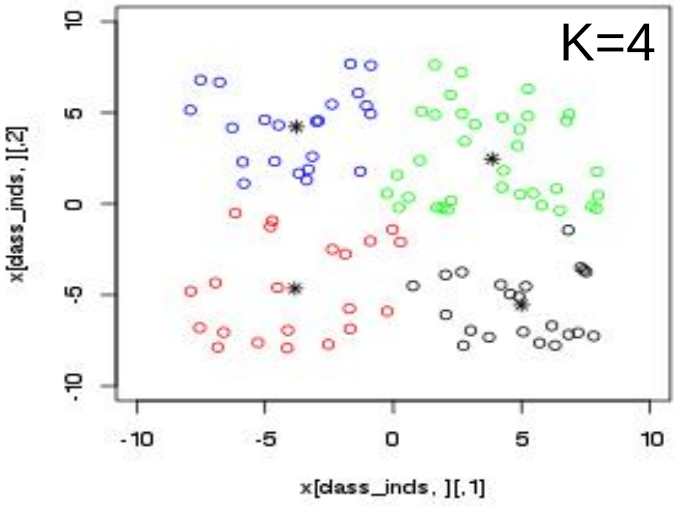
- **Choosing K:**
  - “Elbow” in total-within-cluster SSE as K=1…N
  - Cross-validation: hold out points, compare fit as K=1…N
- **Choosing initial starting points:**
  - take K random data points, do several Kmeans, take best fit
- **Stopping:**
  - may converge to sub-optimal clusters
  - may get stuck or have slow convergence (point assignments bounce around), 10 iterations is often good

Kmeans Example: uniform dist.



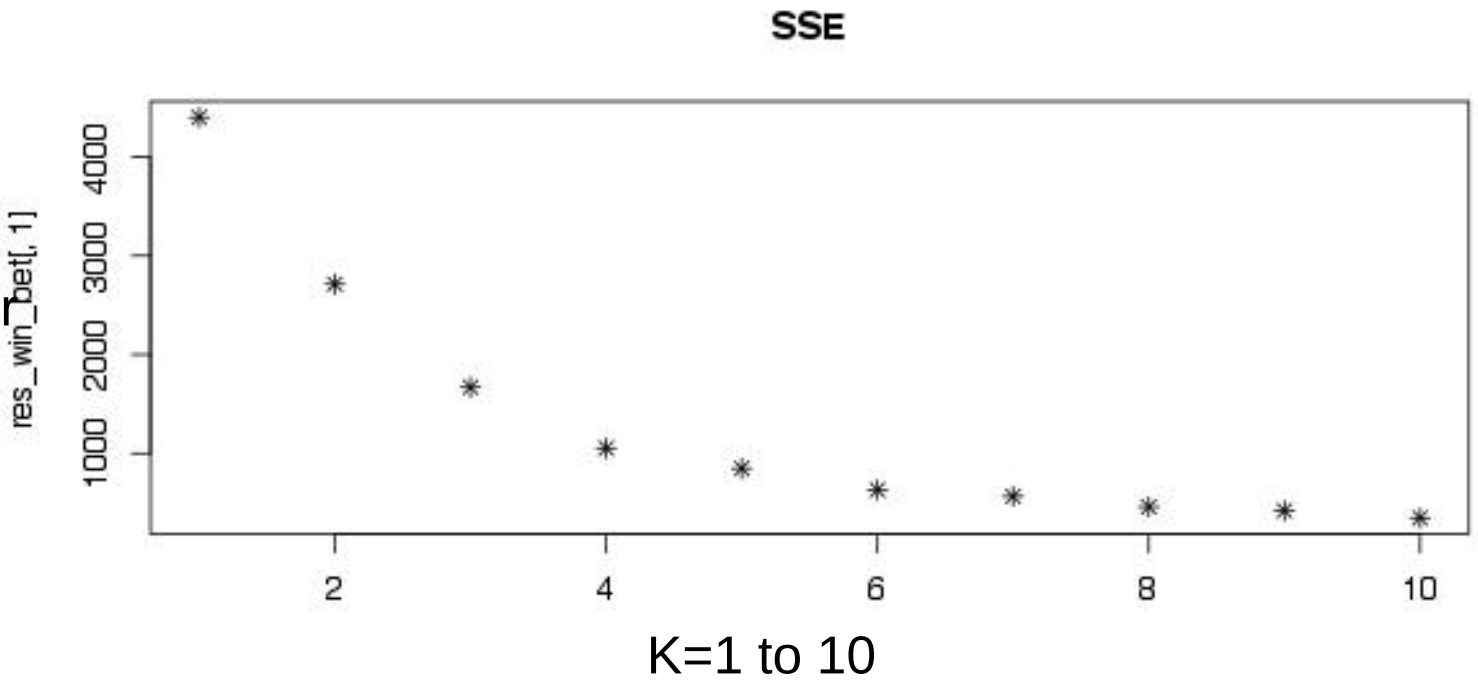






Choosing K - uniform

Total Within Cluster SSE



K=1 to 10

- Smooth decrease across K => less structure

Kmeans Clustering Issues

- **Scale:**
  - Dimensions with large numbers may dominate distance metrics (so can be good to normalize or scale data)

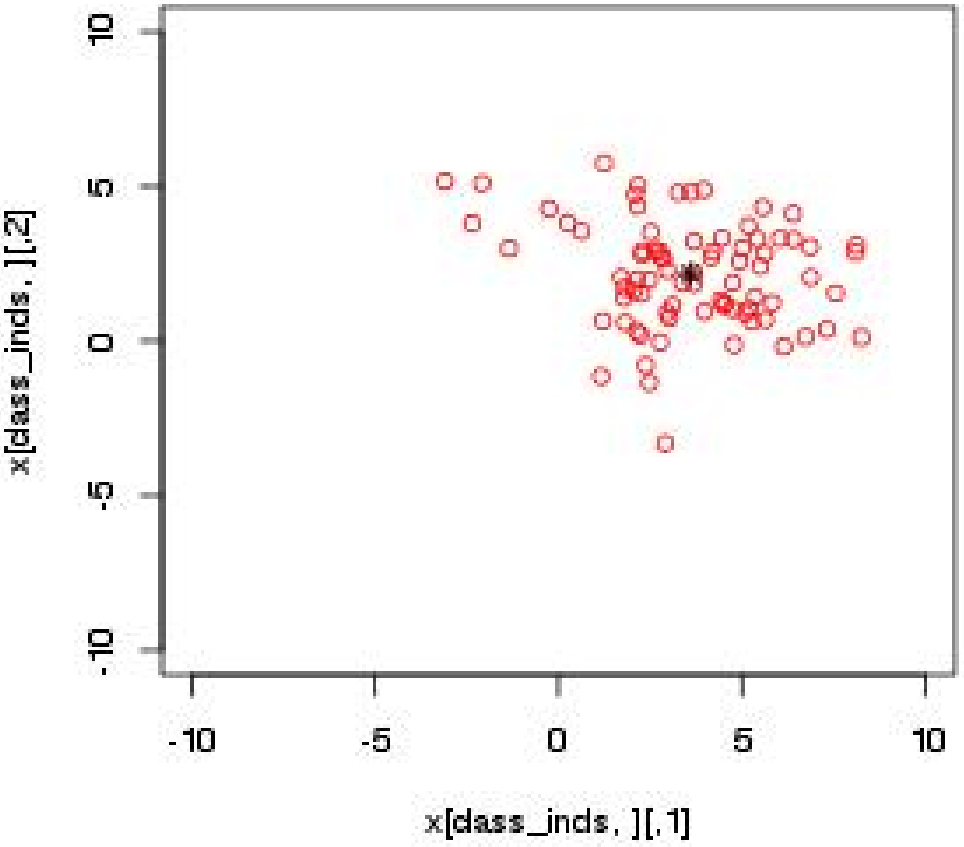
- **Outliers:**
  - Outliers can pull cluster mean

    (K-mediods uses median instead of mean)

Soft Clustering Methods

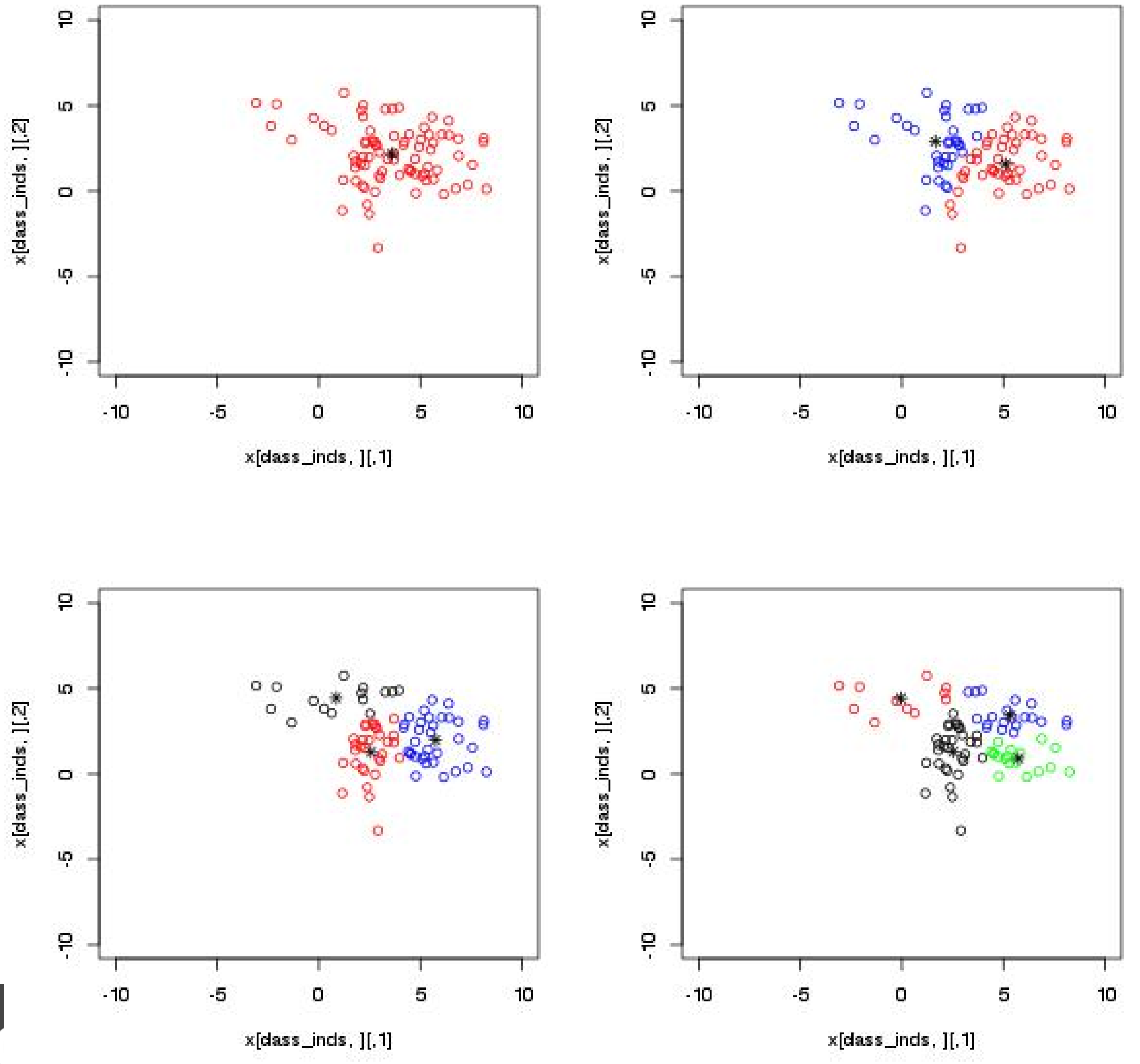
- **Fuzzy Clustering**
  - Kmeans with weighted assignments to all clusters
  - Weights depend on relative distance
  - Find min weighted SSE
- **Expectation-Maximization:**
  - Initialize a mixture of multivariate Gaussian distributions
  - Find means, variances, and mixture weights that maximize probability of data

Kmeans with unequal cluster variance and/or size

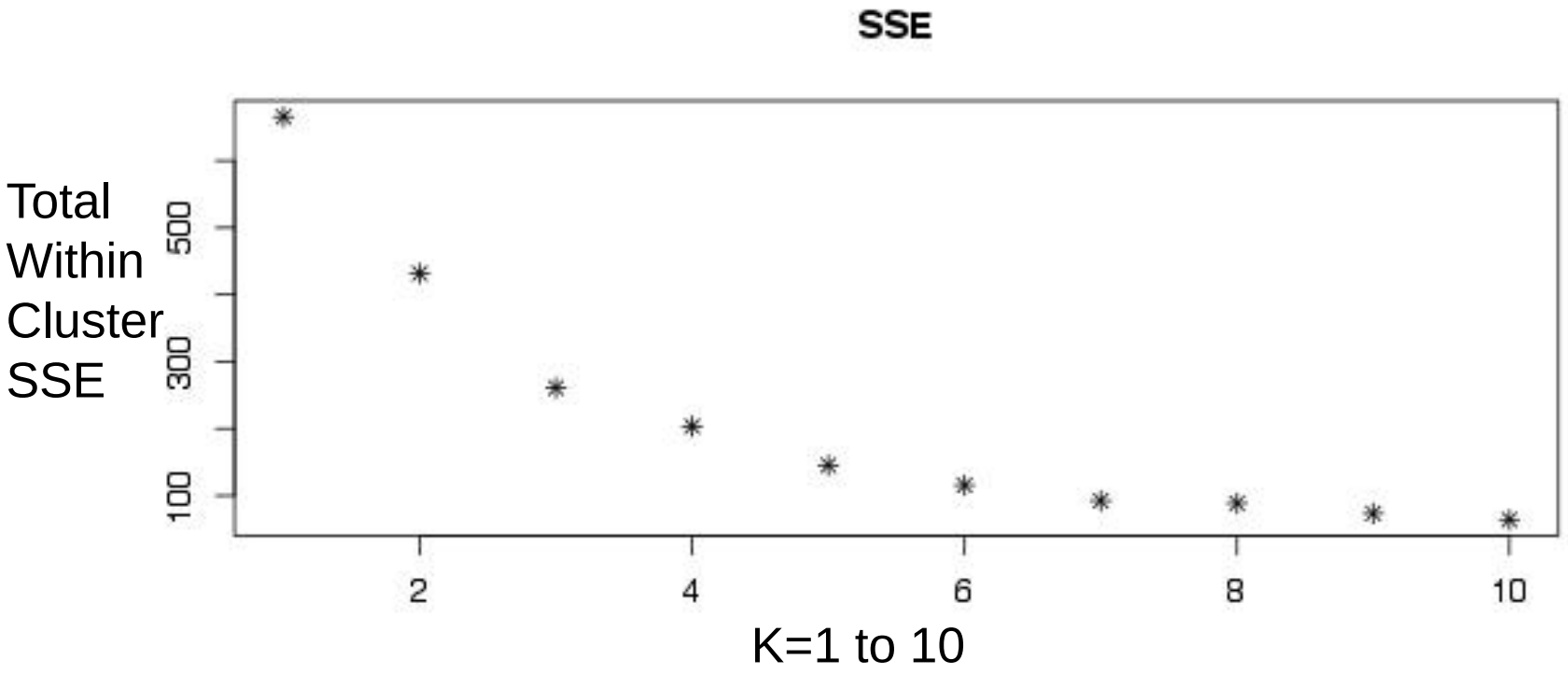


Can you guess K?

Kmeans – unequal cluster variance

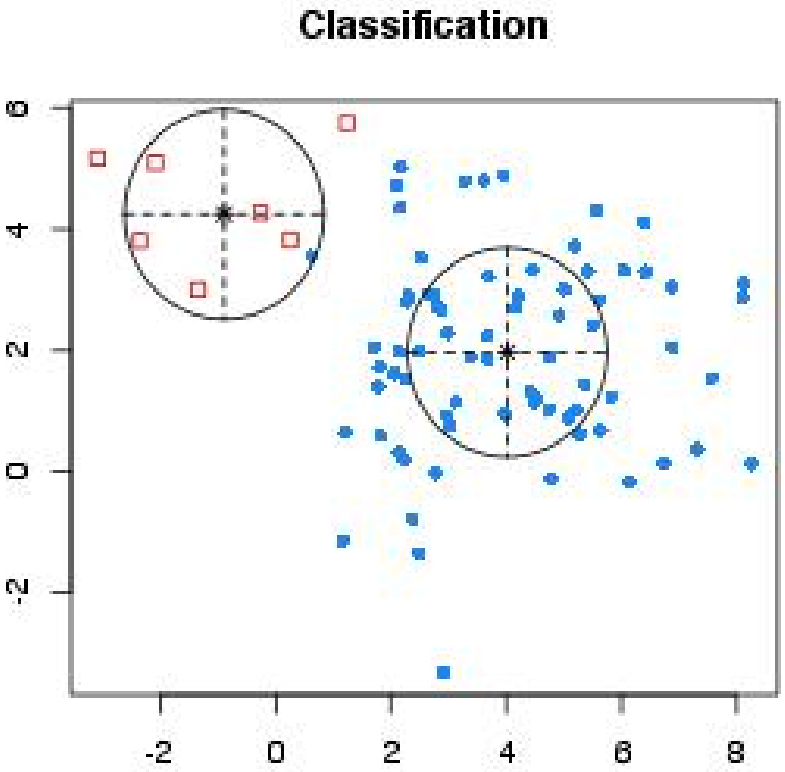


Choosing K – unequal distributions



- Smooth decrease across K => less structure

EM clustering



- Selects K=2
  (either by Information Criterion=
  min of SSE+ K*logN,
  Or by cross-validation)

- Handles unequal variance and/or size

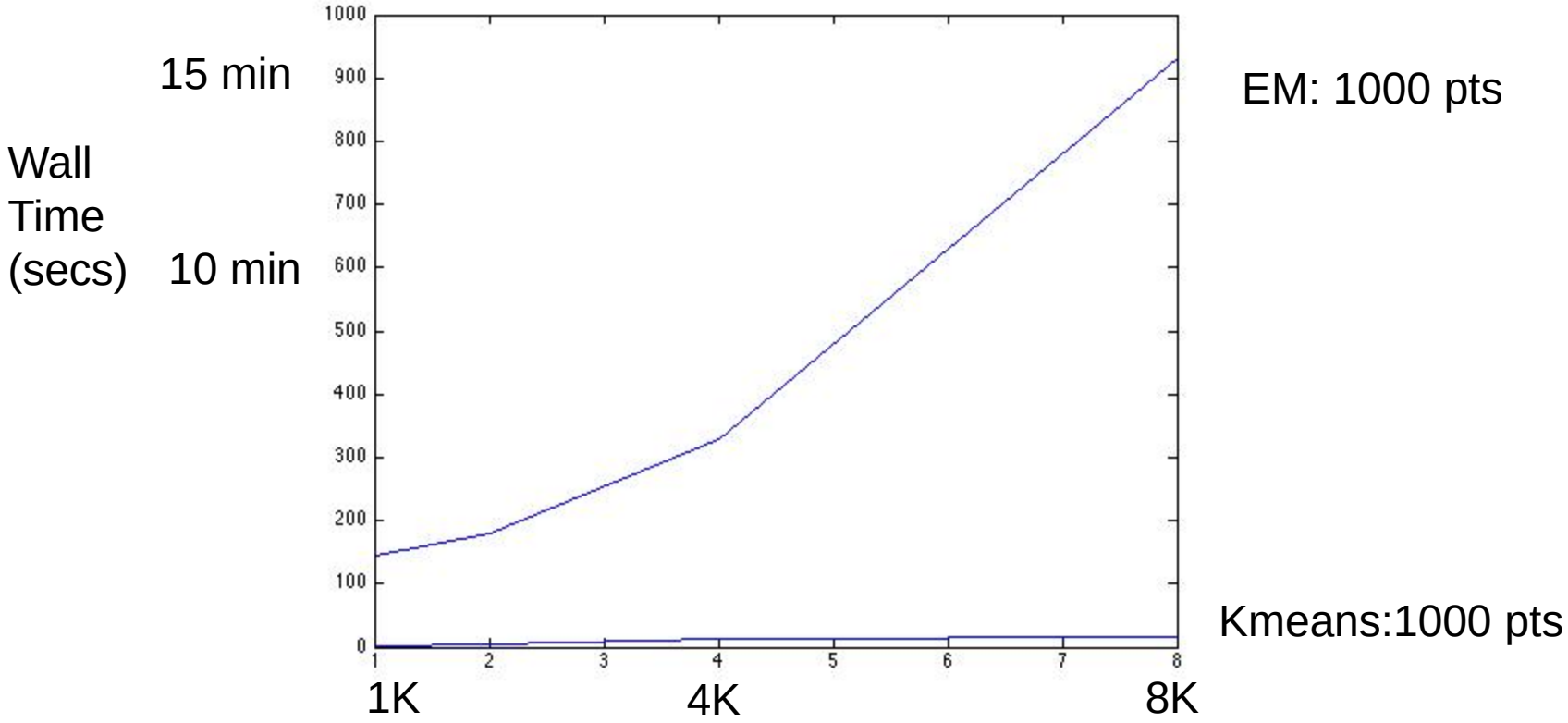
R:
library('mclust')
em_fit=Mclust(x);
plot(em_fit);

Kmeans computations

- **Distance of each point to each cluster center**
  - For *N* points, *D* dimensions: each loop requires *N*D*K* operations
- **Update Cluster centers**
  - only track points that change, get change in cluster center
- **But for EM errors to each cluster center update a probability function**

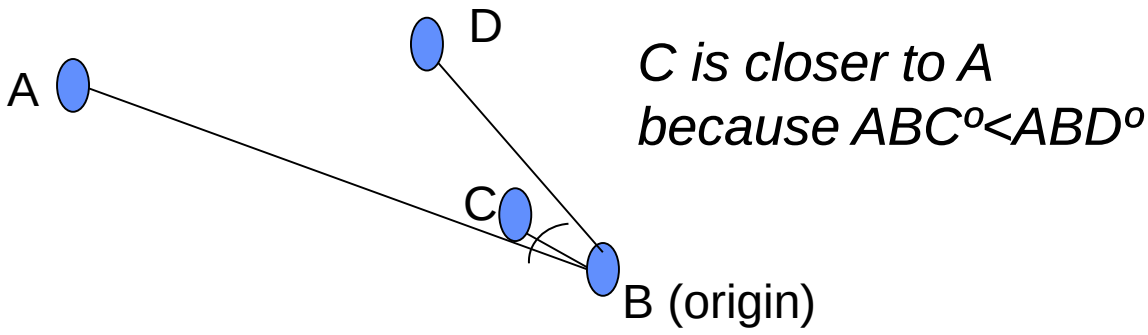
Kmeans vs EM performance

1 Gordon compute node, normal random matrices
R: system.time(Mclust())



Other distance measures

- Cosine: take angle difference (good for sparse vectors)

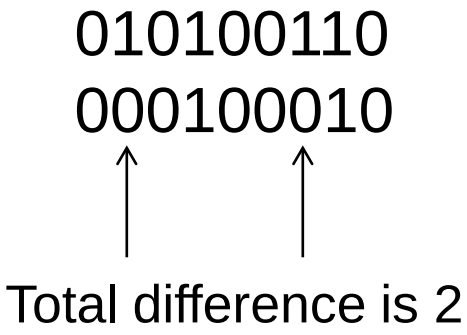


- Mahalanobis: dimensions rescaled by variance

- Jaccard (over sets A,B): 1- (|A∩B| / |AUB|)

Other distance measures

- Hamming distance: count 1 if values different
  e.g. appropriate for binary strings

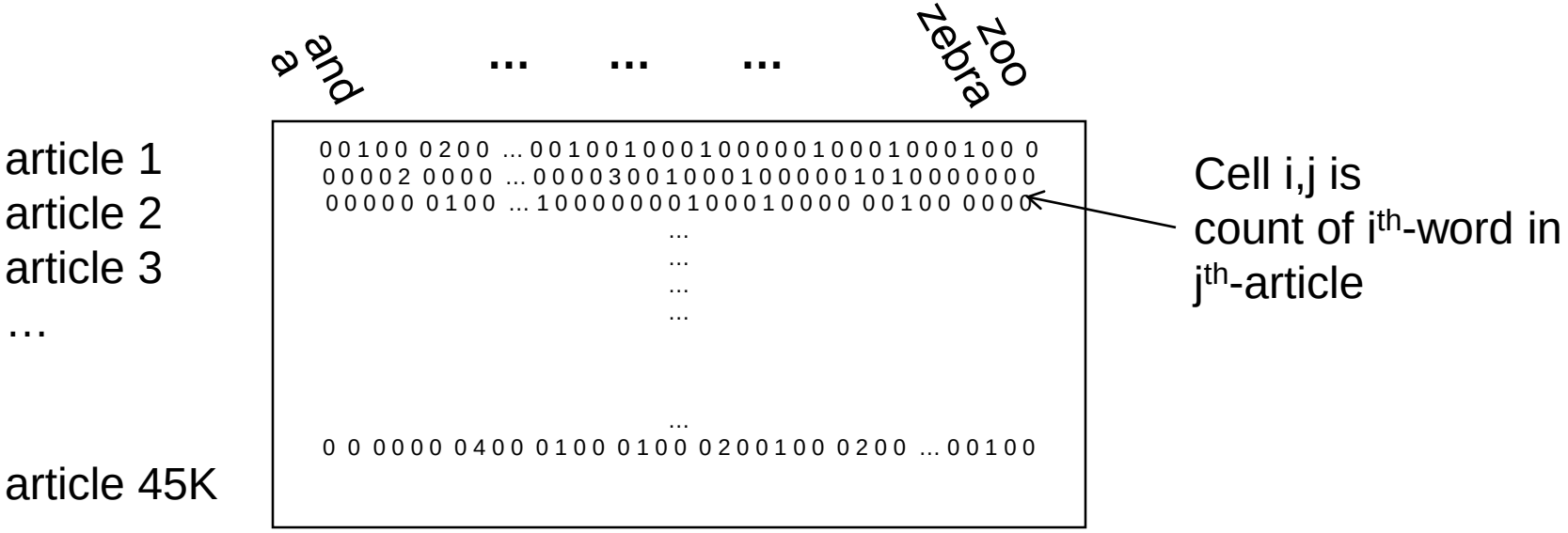


Kmeans big data example

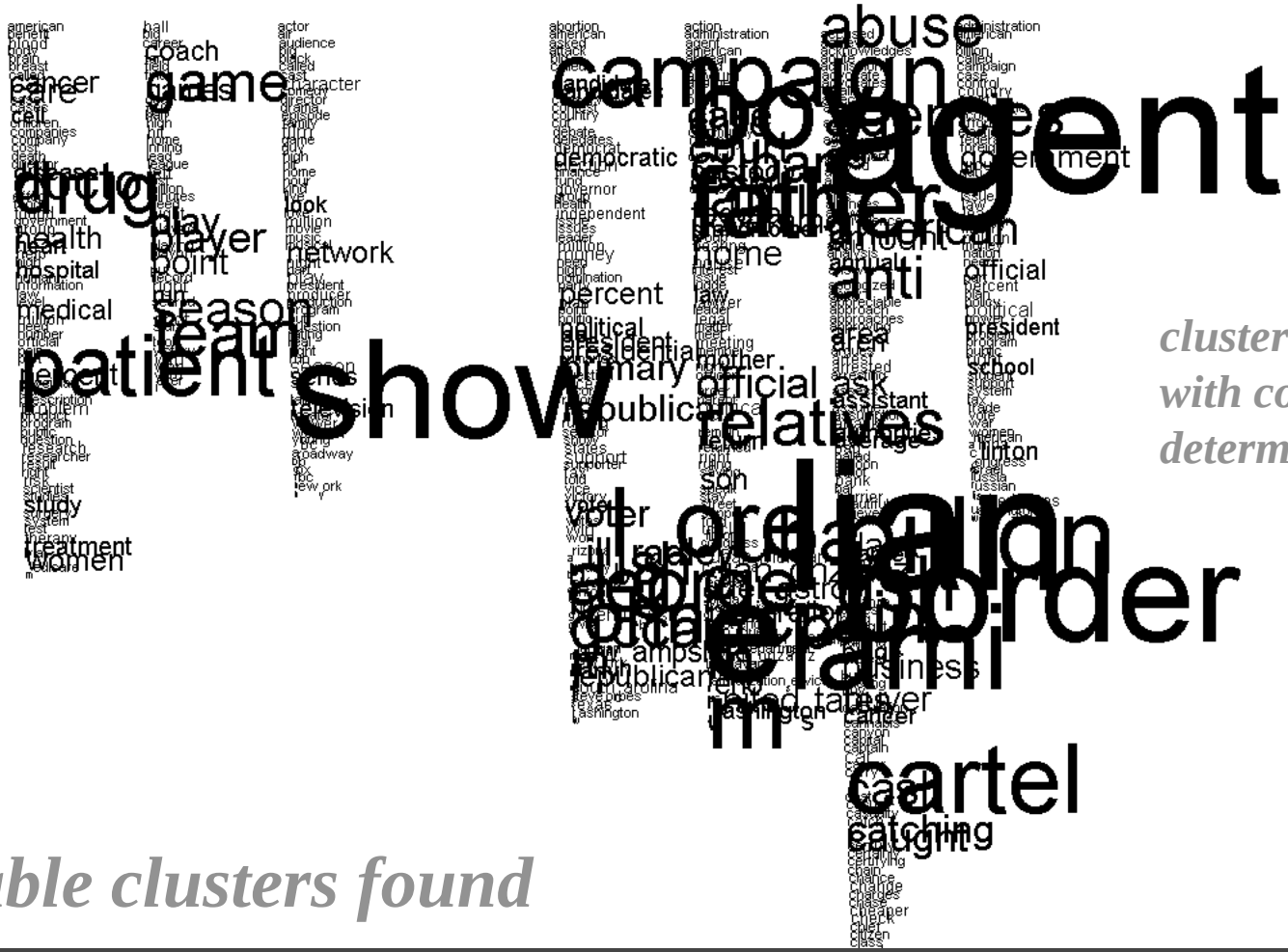
- **45,000 NYTimes articles, 102,000 unique words**

  (UCI Machine Learning repository)

- **Full Data Matrix: 45Kx102K ~ 40Gb**



Kmeans results



cluster means shown with coordinates determining fontsize

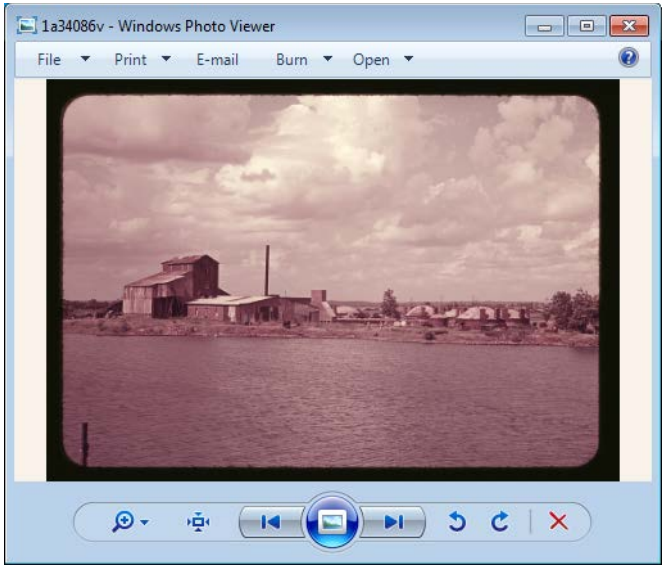
7 viable clusters found

Kmeans for image segmentation

R snippet

get packages
read 1024X718X3 RGB image
convert to matrix 1024*718 X 3
Choose K by trial and error
run Kmeans and display

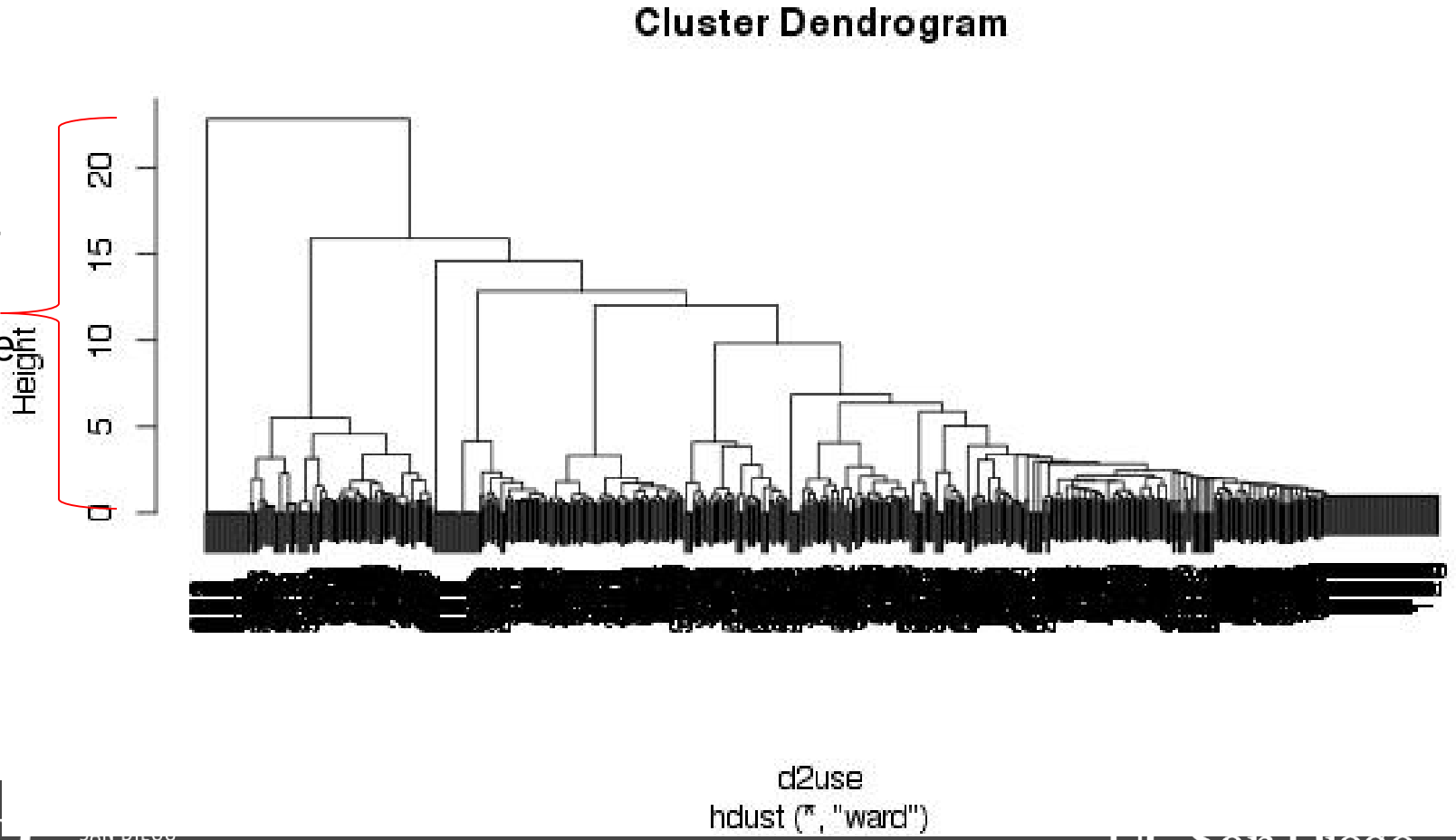
```
install.packages('ripa')
library('ripa')

source("http://bioconductor.org/biocLite.R")
biocLite()
biocLite("EBImage")

library('EBImage')
im=readImage('1a34086v.jpg')

library('ripa')
img=rgb2grey(im, coefs=c(0.30, 0.59, 0.11))

imgx1  =as.vector(img)
numk=8
km_imx1=kmeans(imgx1,numk,50,1);
img_km_mat =matrix(km_imx1$cluster,dim(im)[1],dim(im)[2])

display(img_km_mat/numk)
```



Other Clustering Methods

Hierarchical Clustering

- **hclust with “Ward” distance gives spherical clusters**

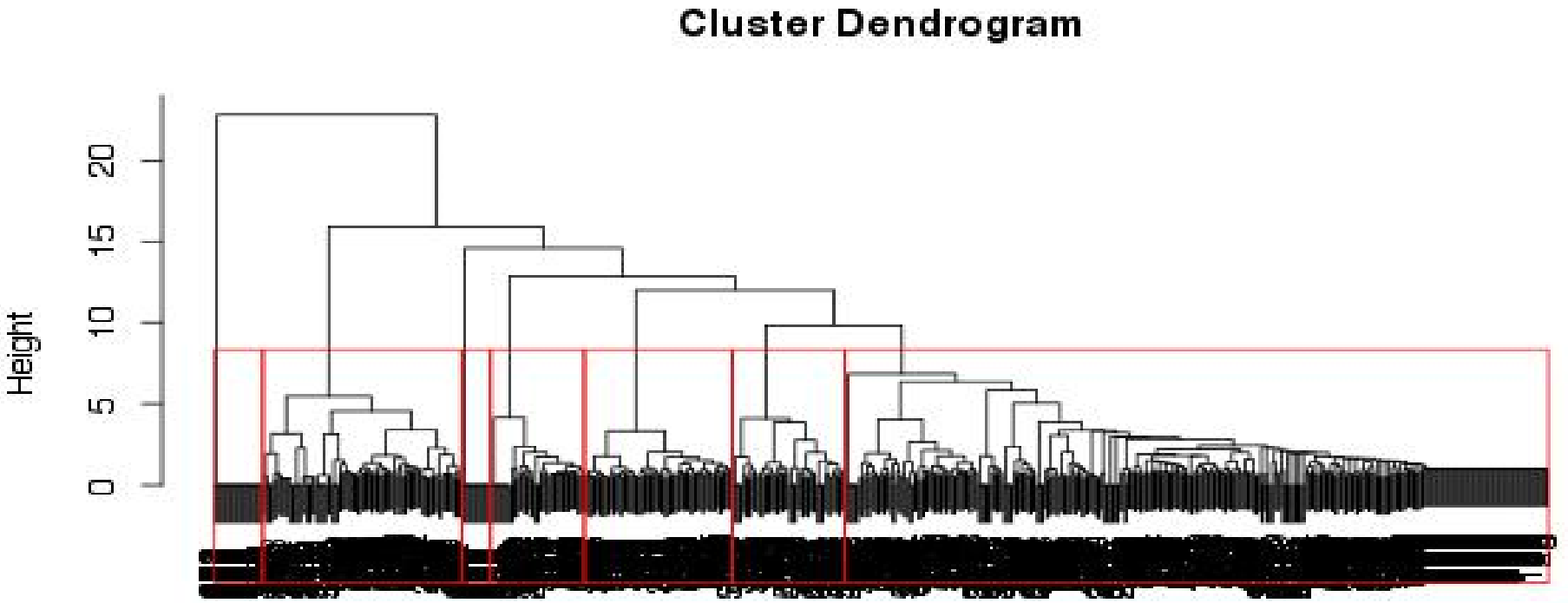


the cluster distance when 2 are combined

Hierarchical Clustering

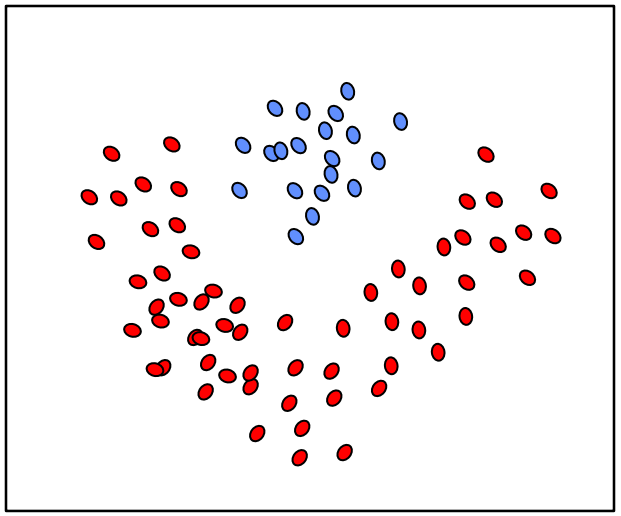
- **Where height change looks big, cut off tree**

```
groups <- cutree(fit, k=7)
rect.hclust(fit, k=7, border="red")
```



Other Clustering

- **Density based clustering**
  - build neighborhoods around seed points
  - link neighborhoods
  - Results in arbitrary cluster shapes, good for image and spatial clustering



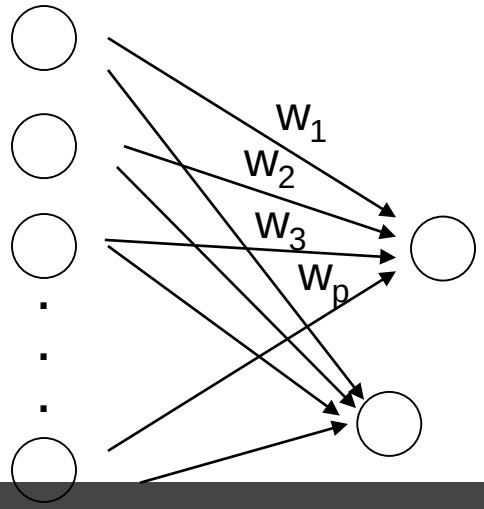
Other Clustering

- **Neural Network Based (e.g.)**

  initialize weights to coordinate values for a seed point

  set input nodes to coordinate values for data points

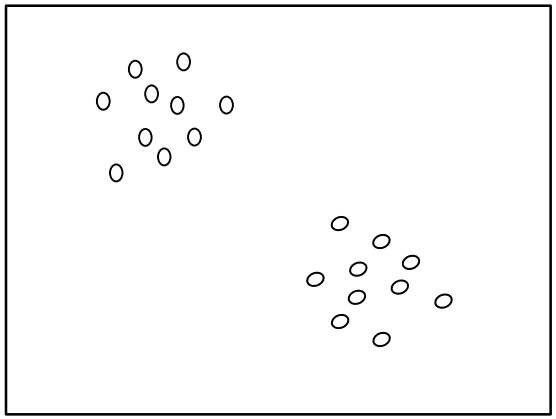
  get best match to seed for each data point and adjust weights toward the data point

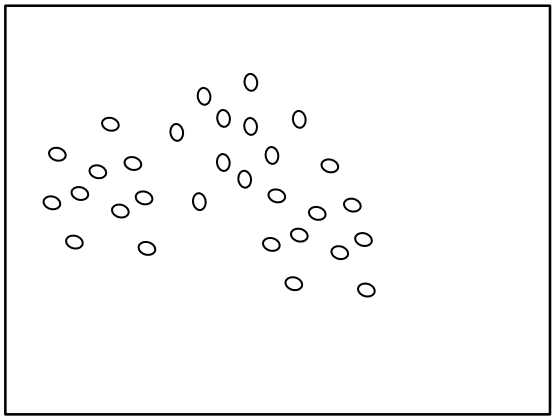


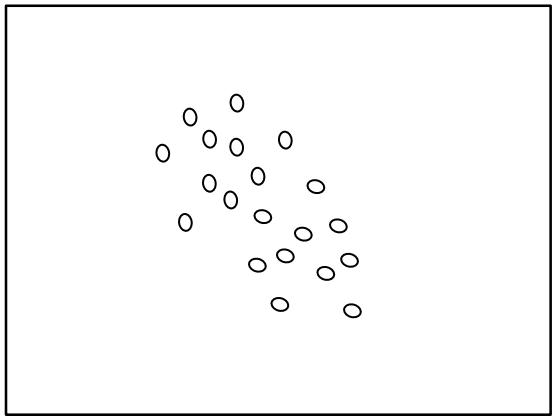
Target node(s), starts as a seed point and ends up as a cluster mean

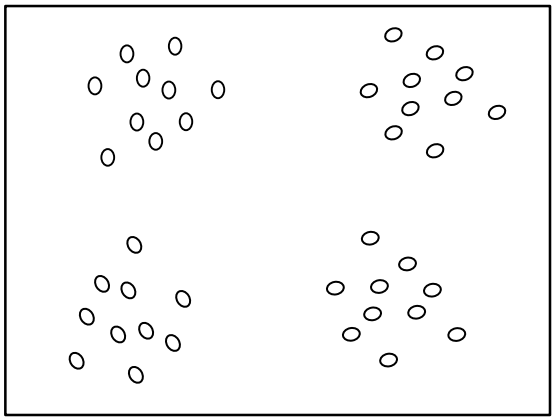
Pause

Imagine these 2 dimensional input spaces:
Which of these is easy or hard to cluster? (no class labels)

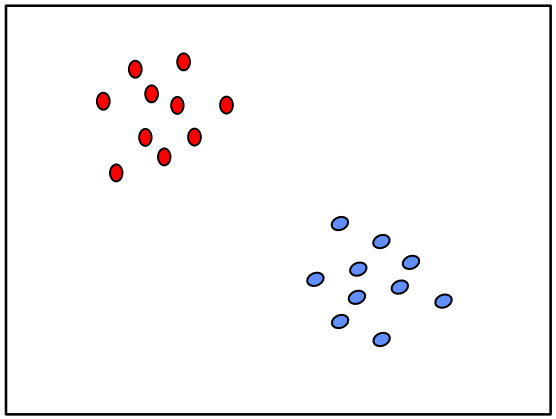


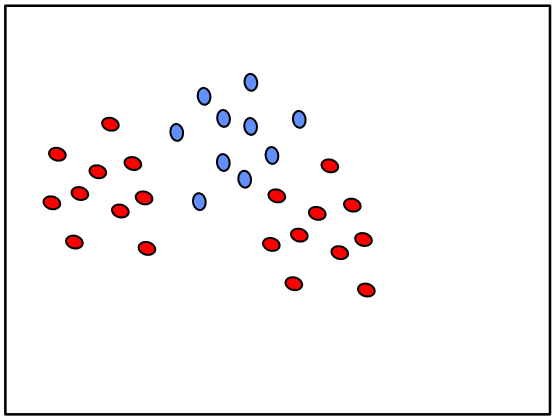


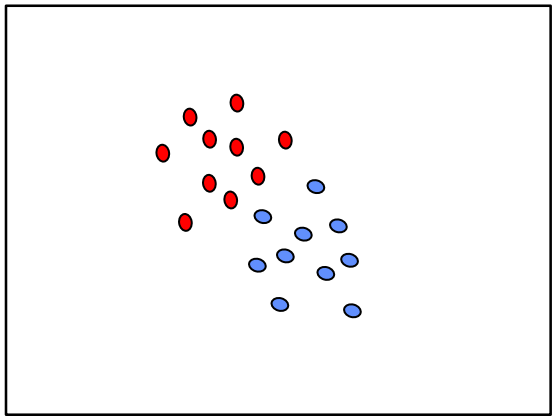


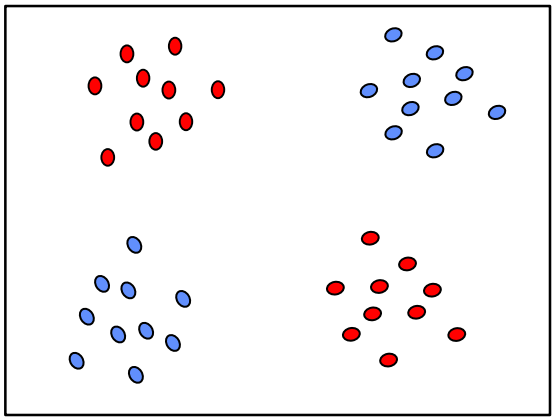


Now imaging there are two classes

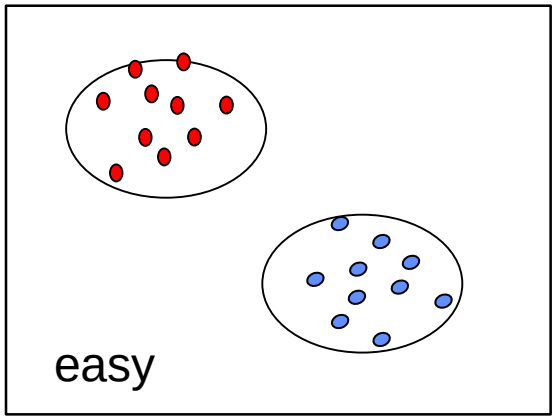


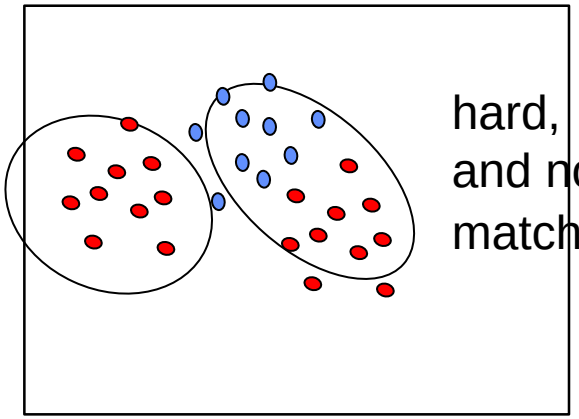






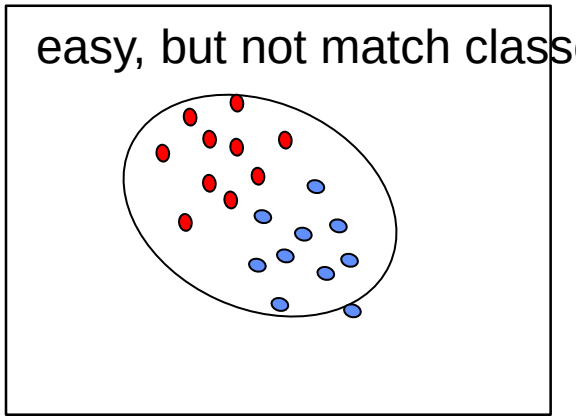
Potential clusters

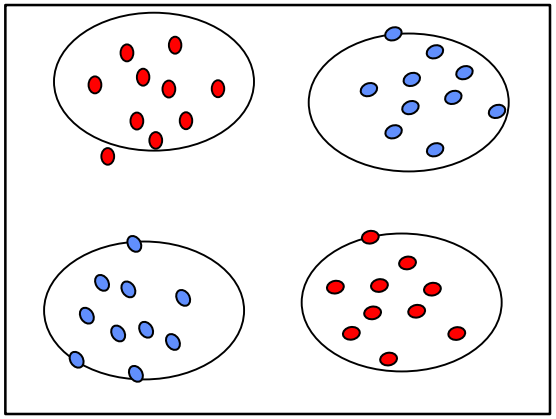




hard,
and not
match classes

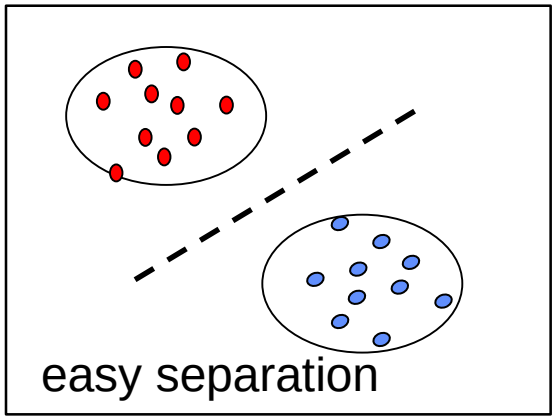
easy, but not match classes

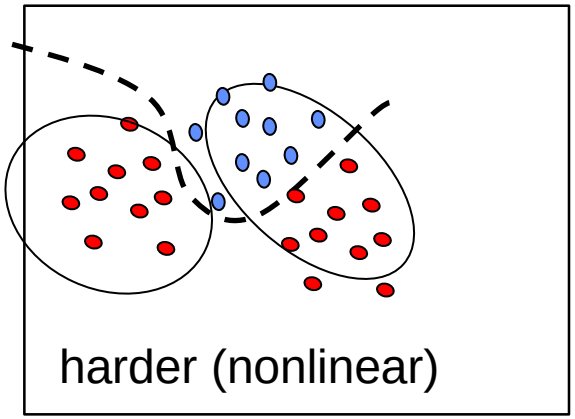


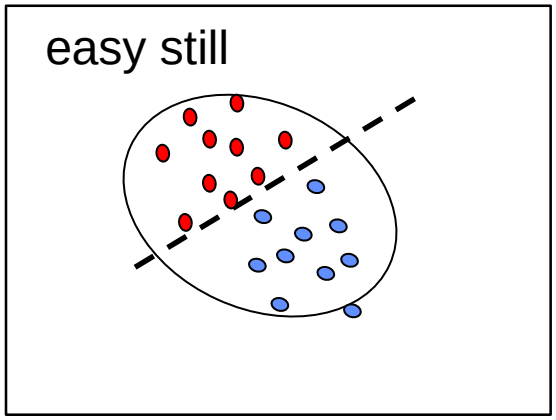


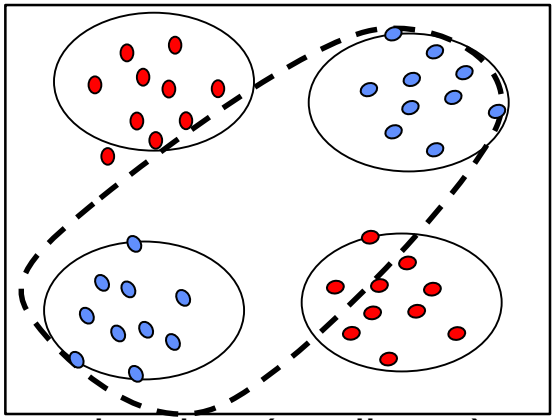
easy,
4 clusters
match 2
classes

Which are easy or hard to classify? (ie separate red or blue with lines)









Upshot:
No easy relationship between clusters and classification

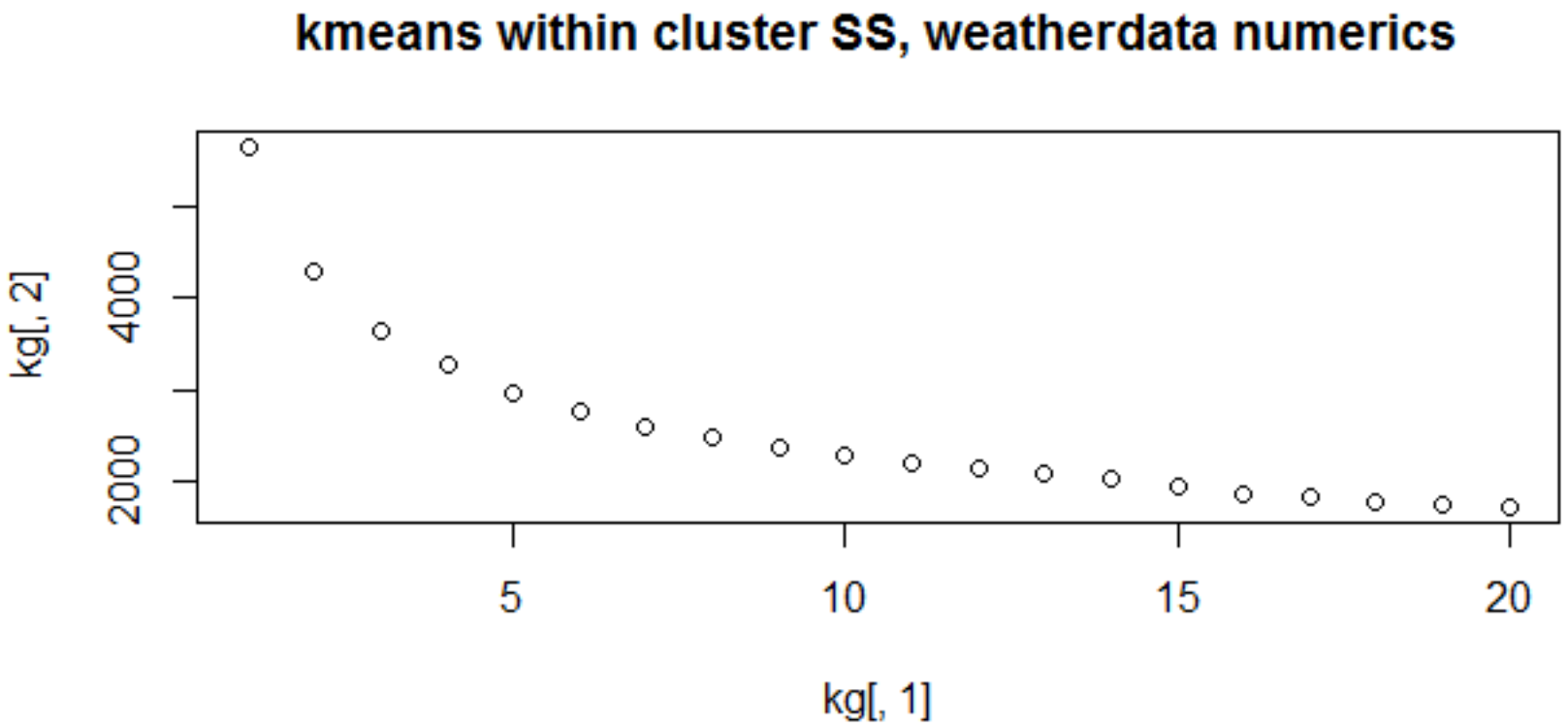
Pause

SI2016_clustering3.R

```
W_table = read.table('weather.csv', header=TRUE,sep=",",
                    stringsAsFactors = TRUE)

W_table=subset(W_table, select=-c(RISK_MM))

#2  Get numeric columns only
col_classes = sapply(W_table,class)
num_inds    = c(which(col_classes=='numeric'), which(col_classes=='integer'))
W_num       = W_table[,num_inds]


#Remomve rows that are missing data
rem_ind  = complete.cases(W_num)
print(paste('Number of incomplete cases:',length(which(rem_ind==FALSE)) ))
W_num    = W_num[rem_ind,]
```

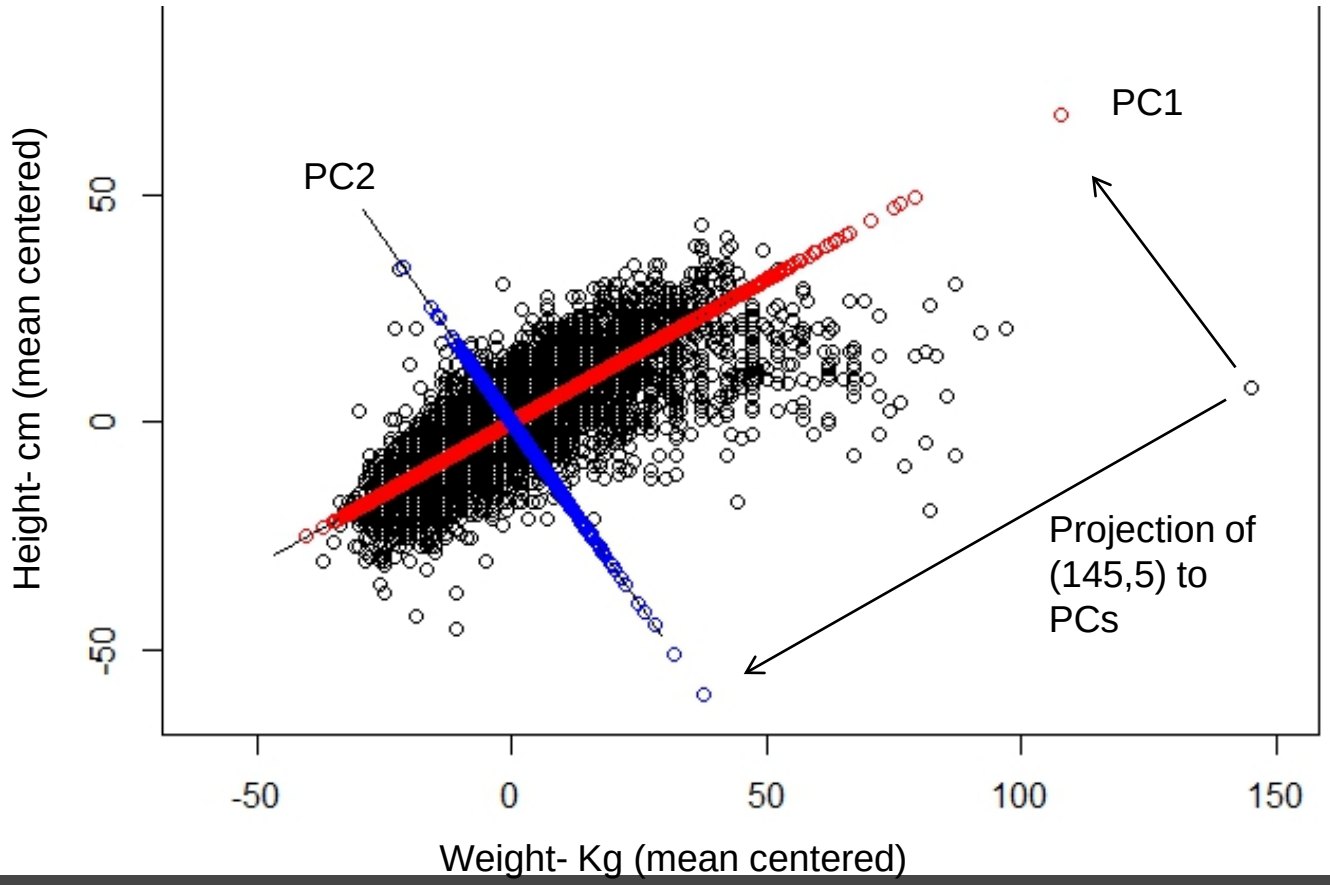
SI2016_clustering3.R

```
W_mncntr=scale(W_num,center=TRUE,scale=TRUE)

#4 Run kmeans for 20 values of K
kg=matrix(0,20,2)
for (i in 1:20){
  ktest=kmeans(W_mncntr,i,20,5);
  kg[i,1]=i;
  kg[i,2]=ktest$tot.withinss;
}
plot(kg[,1],kg[,2],main='kmeans within cluster SS, weatherdata numerics
```


kmeans within cluster SS, weatherdata numerics
kg[, 2]
2000
4000
5
10
15
20
kg[, 1]

```
> str(ktest)
List of 9
 $ cluster     : Named int [1:354] 14 14 12 6 9 9 9 9 9 7 ...
  ..- attr(*, "names")= chr [1:354] "1" "2" "3" "4" ...
 $ centers     : num [1:20, 1:16] -0.532 -1.436 -0.738 1.1 1.452 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:20] "1" "2" "3" "4" ...
  .. ..$ : chr [1:16] "MinTemp" "MaxTemp" "Rainfall" "Evaporation" ...
 $ totss       : num 5648
 $ withinss    : num [1:20] 96.5 81.5 102.4 48.4 61.7 ...
 $ tot.withinss: num 1731
 $ betweenss   : num 3917
 $ size        : int [1:20] 17 20 16 13 13 5 18 25 21 32 ...
 $ iter        : int 4
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

PCA on 2012 Olympic Althetes' Height by Weight scatter plot



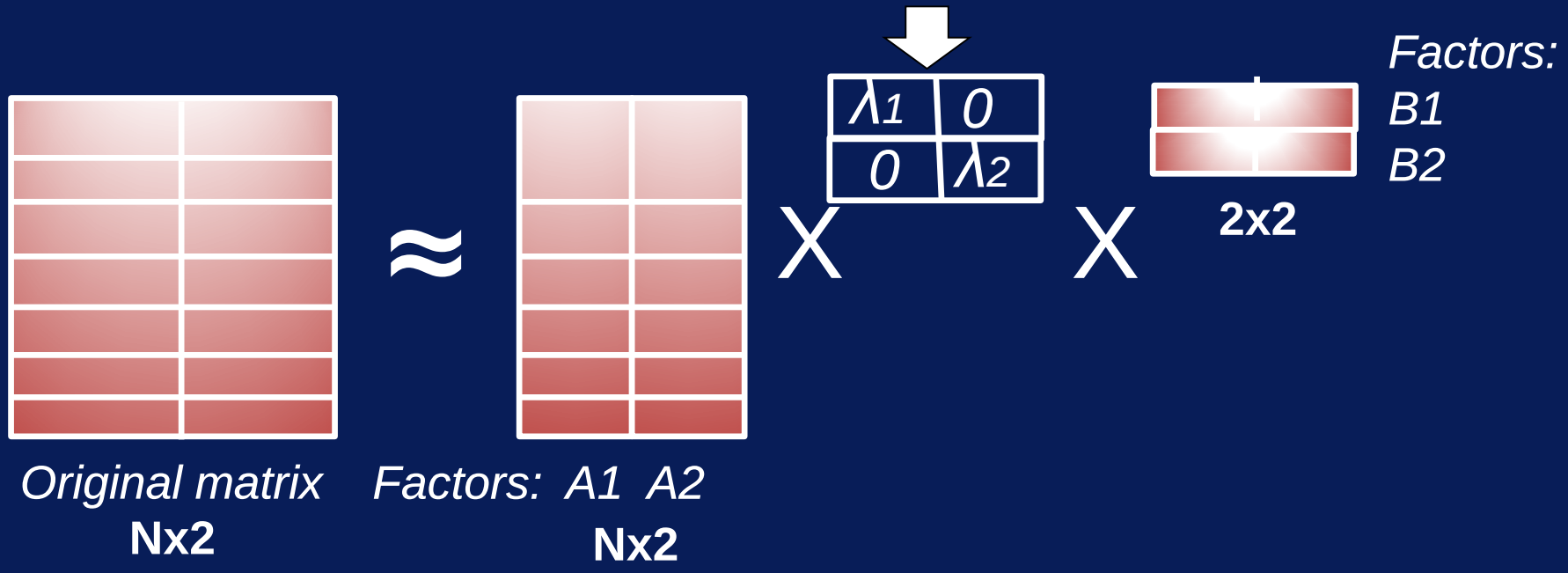
Total Variance Conserved:

Var in Weight +
Var in Height
=
Var in PC1 +
Var in PC2

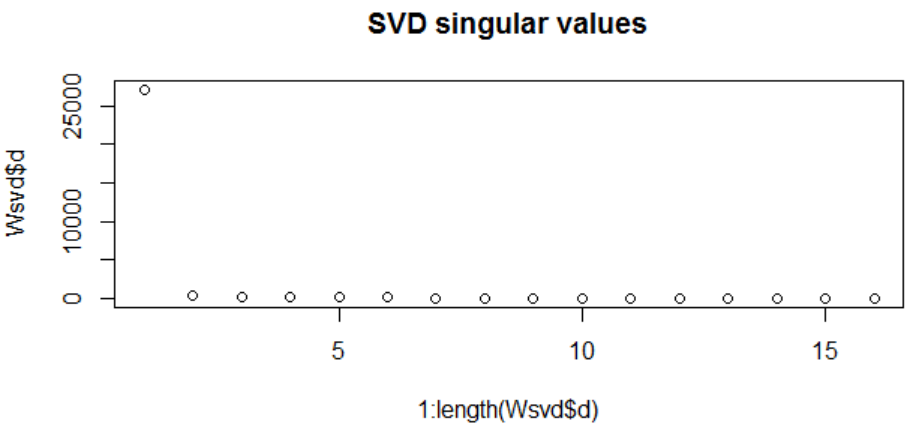
In general:

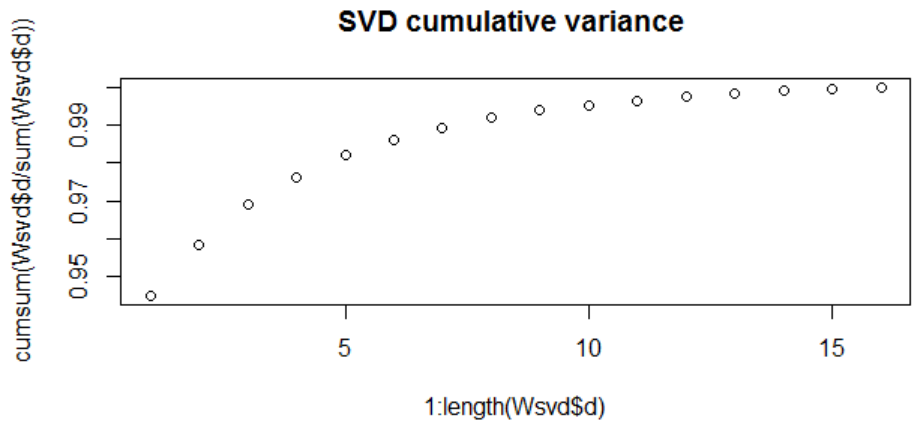
Var in PC1>
Var in PC2>
Var in PC3…

SVD: factors and ‘singular’ scale values

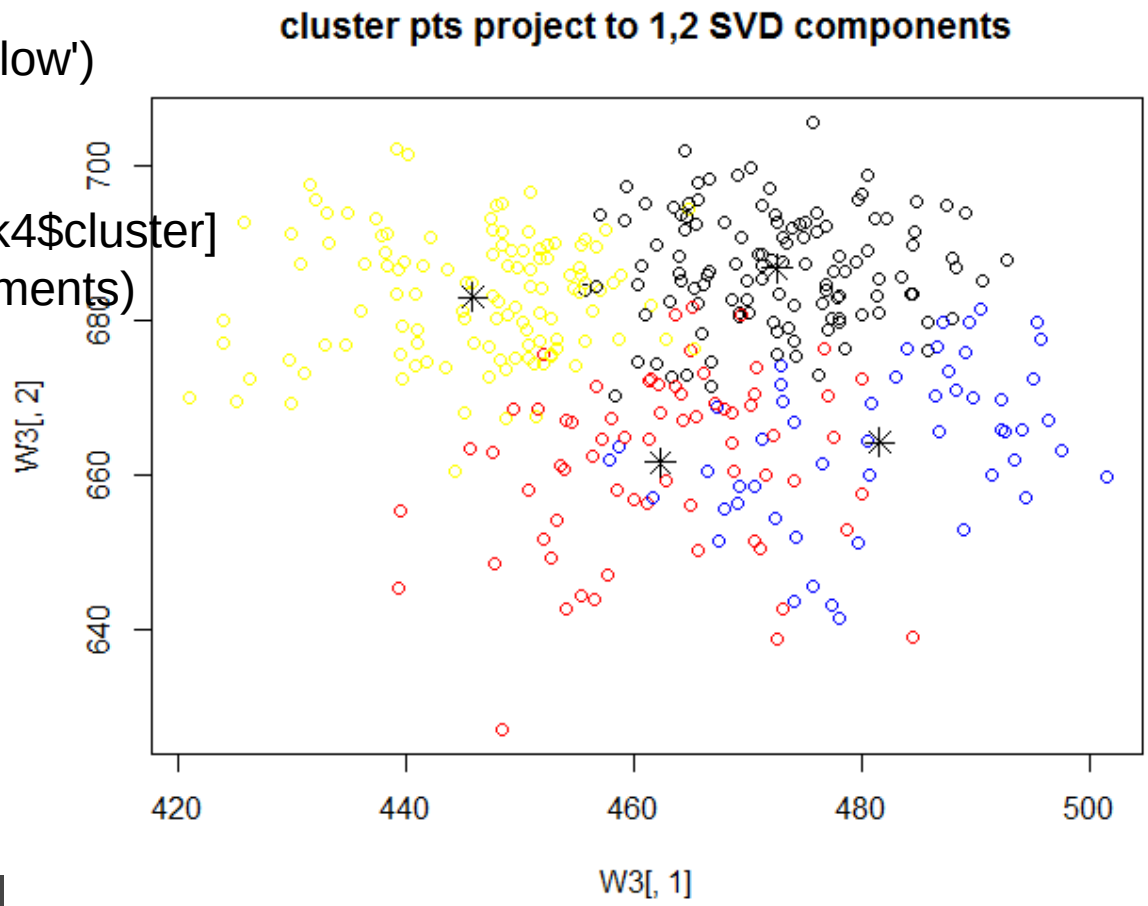


```
Wsvd=svd(W_num)

#Step 5  plot the singular values
plot(1:length(Wsvd$d),Wsvd$d,
   main='SVD singular values')

#or
plot(1:length(Wsvd$d),cumsum(Wsvd$d)/sum(Wsvd$d)),
   main='SVD cumulative variance ')
```





```
W3 = as.matrix(W_num) %*% Wsvd$v[,1:3]
                                    #notice that W_num is still a dataframe

#get Kmeans for 4 clusters
k4 = kmeans(W_num,4,20,5)

#get color scheme
col2use = c('red','blue','black','yellow')

#get cluster assignment in colors
colassignments          = col2use[k4$cluster]
plot(W3[,1],W3[,2],col=colassignments)
```



Principle Components vs Clustering

- PCA reduces dimensions, Clustering reduces to categorical groups
- In some cases, *k* PCs ⇔ *k* clusters
- It is also useful to visualize clusters in PC space

Summary

- **Having no label doesn’t stop you from finding structure in data**
- **Unsupervised methods are somewhat related**