Clustering: K-means

-Applied Multivariate Analysis-

Lecturer: Darren Homrighausen, PhD

Clustering Introduction

When clustering, we seek to simplify the data via a small(er) number of summarizing variables

PCA looks to find a low dimensional representation of the observations that explain a good fraction of the sums of squares

Alternative clustering approaches instead look to find subgroups among the observations in which they are similar

Clustering Introduction

We will focus on two particular clustering algorithms

- K-means: Seeks to partition the the observations into a pre-specified number of clusters.
- Hierarchical: Produces a tree-like representation of the observations, known as a dendrogram.

There are advantages (disadvantages) to both approaches.

We can cluster observations on the basis of the covariates in order to find subgroups of observations.

(It is common in clustering to refer to covariates as features)

Just as easily, we can find clusters of features based on the observations to find subgroups in the features.

We will focus on clustering the observations. You can cluster features by transposing $\mathbb{X}$ (that is, clustering on $\mathbb{X}^\top$).

K-means

1. Select a number of clusters K.
2. Let $C_1, \ldots, C_K$ partition $\{1, 2, 3, \ldots, n\}$ such that
   - All observations belong to some set C_j.
   - No observation belongs to more than one set.
3. K-means attempts to form these sets by making within-cluster variation, $W(C_k)$, as small as possible.

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k).$$

4. To Define W, we need a concept of distance. By far the most common is Euclidean

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} ||X_i - X_{i'}||_2^2.$$

That is, the average (Euclidean) distance between all cluster members.

K-means

It turns out

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k). \tag{1}$$

is too hard of a problem to solve computationally (K^n partitions!).

So, we make a greedy approximation:

1. Randomly assign observations to the K clusters
2. Iterate until the cluster assignments stop changing:
   - For each of K clusters, compute the centroid, which is the p-length vector of the means in that cluster.
   - Assign each observation to the cluster whose centroid is closest (in Euclidean distance).

This procedure is guaranteed to decrease (1) at each step.

Warning: It finds only a local minimum, not necessarily the global one. Which local min depends on step 1.

K-means: A Summary

To fit K-means, you need to

1. Pick K (inherent in the method)
2. Convince yourself you have found a good solution (due to the randomized approach to the algorithm).

It turns out that 1. is difficult to do in a principled way. We will discuss these shortly.

For 2., a commonly used approach is to run K-means many times with different starting points. Pick the solution that has the smallest value for

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k)$$

As an aside, why can't we use this approach for picking K?

K-means: A Summary

To fit K-means, you need to

1. Pick K (inherent in the method)
2. Convince yourself you have found a good solution (due to the randomized approach to the algorithm).

It turns out that 1. is difficult to do in a principled way. We will discuss these shortly.

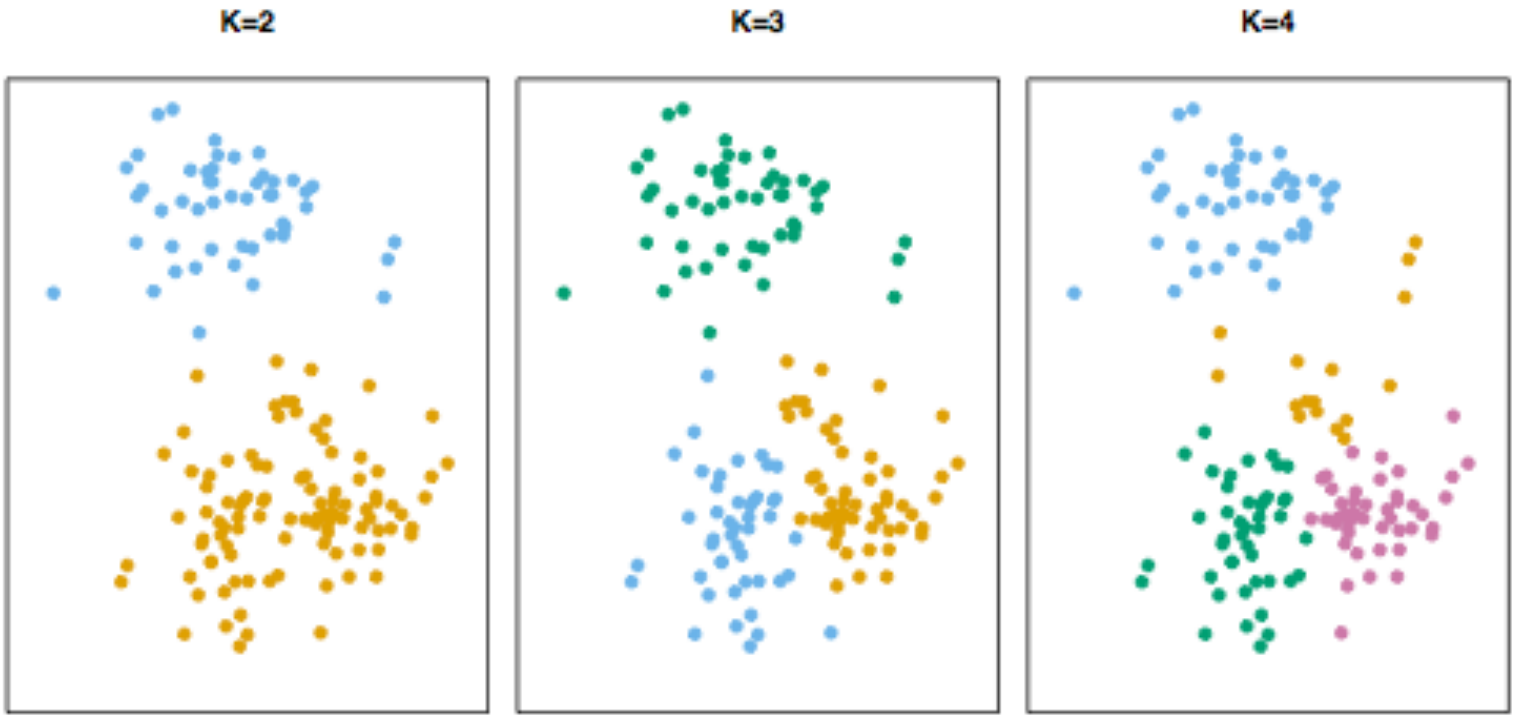
For 2., a commonly used approach is to run K-means many times with different starting points. Pick the solution that has the smallest value for

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k)$$

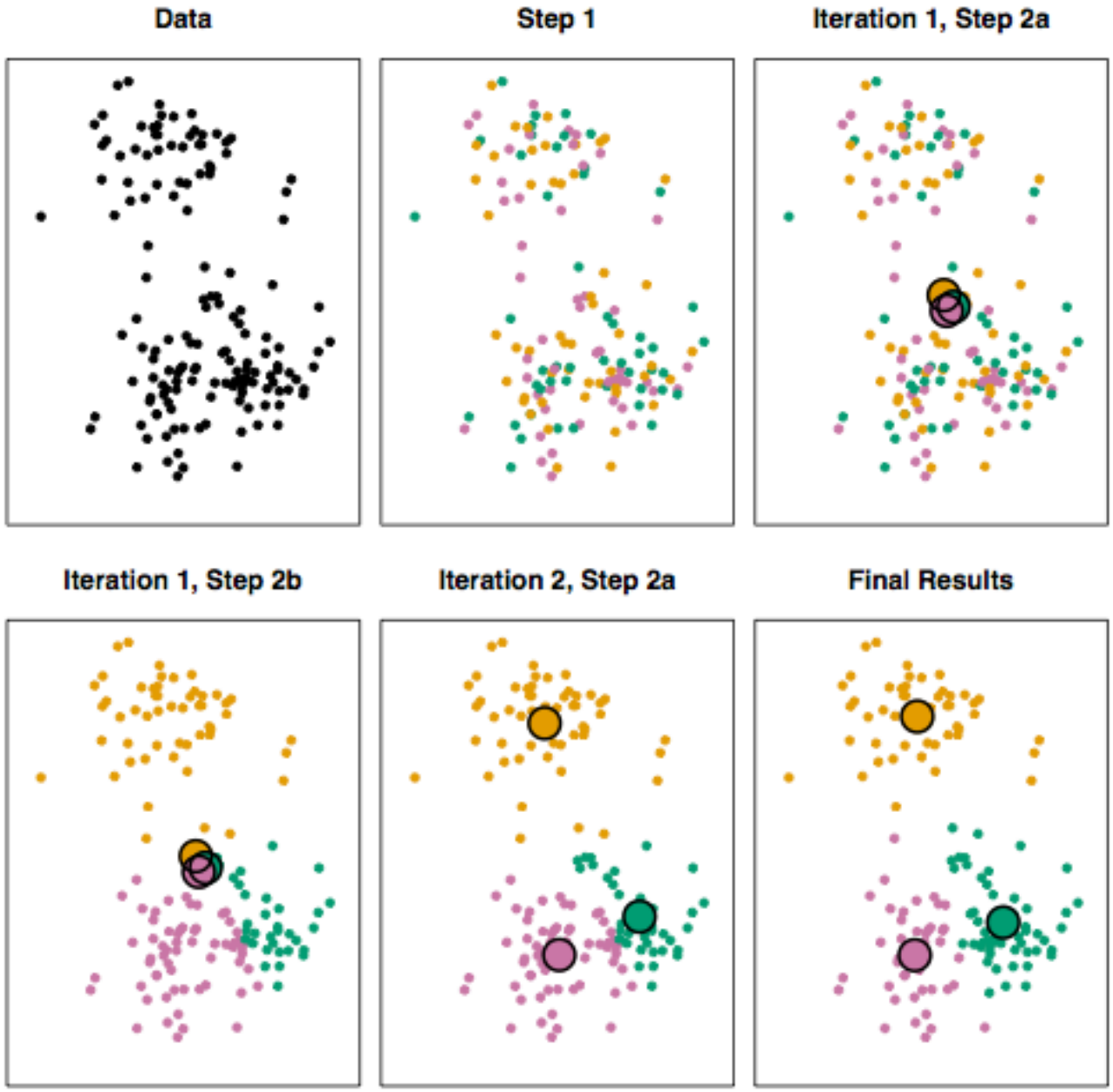
As an aside, why can't we use this approach for picking K?

(We would choose $K = n$)

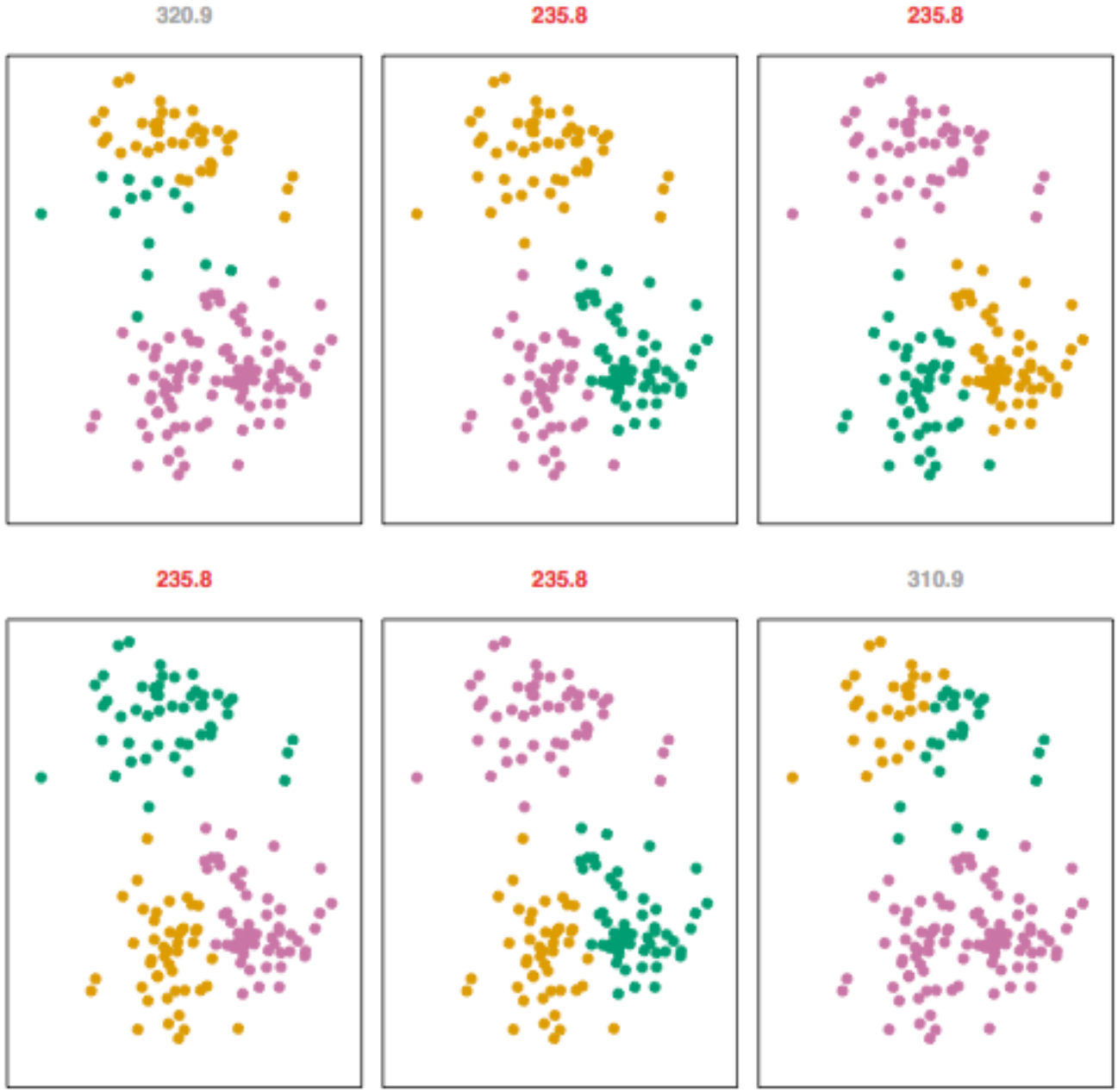
K-means: Various K's



K-means: Algorithm at Work



K-means: Finding Good Local Minimum



K-means in R

Like usual, the interface with R is very basic

```
n   = 30
X1 = rnorm(n)
X2 = rnorm(n)
X   = cbind(X1,X2)
K = 3
kmeans.out = kmeans(X, centers=K)
> names(kmeans.out)
[1] "cluster"      "centers"      "totss"      "withinss"
[5] "tot.withinss" "betweenss"    "size"
> kmeans.out$cluster
 [1] 2 2 2 2 2 2 1 1 2 2 3 1 2 1 2 2 2
 3 1 2 2 1 3 2 1 3 3 1 2 3
```

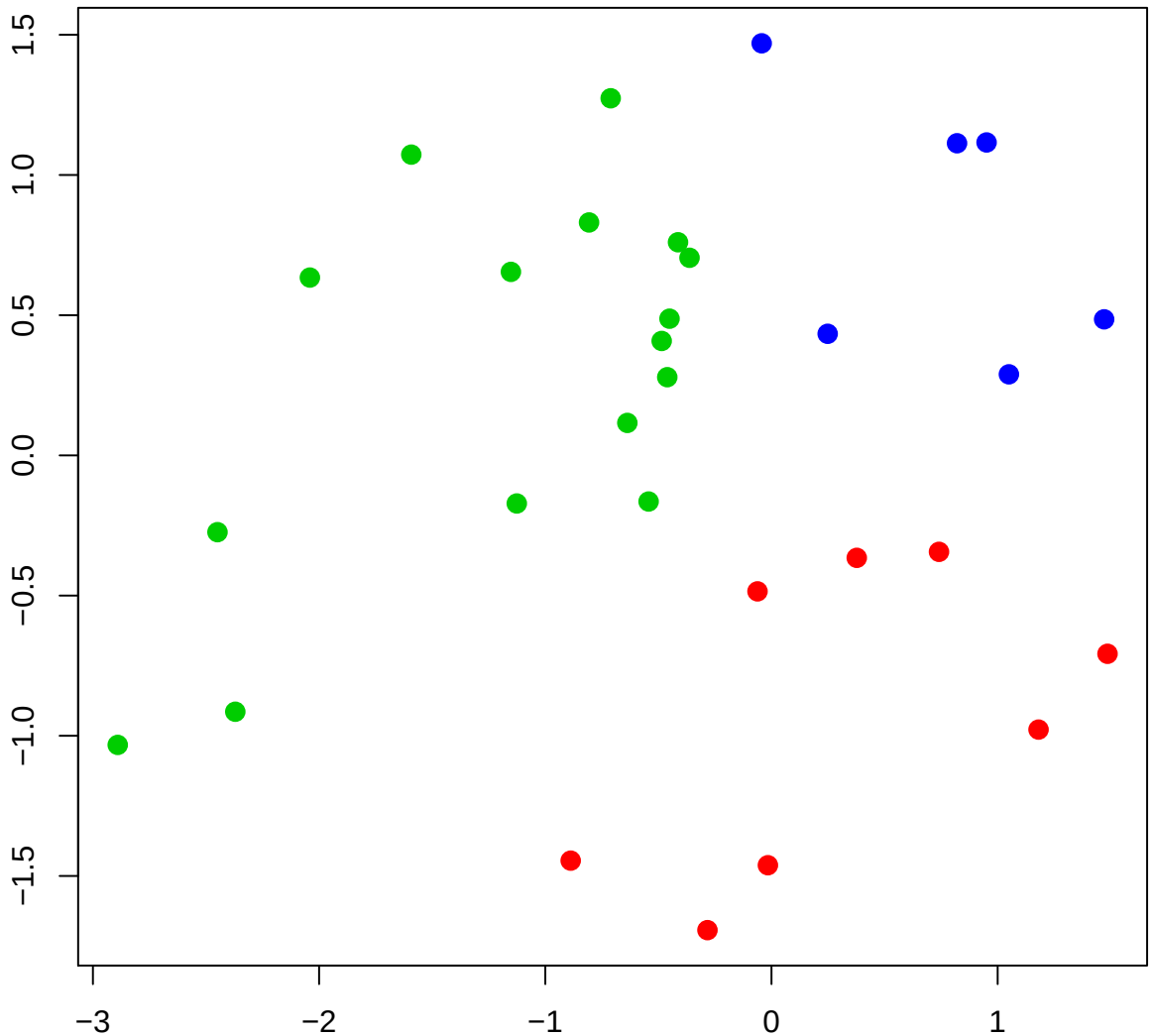
K-means in R

Like usual, the interface with R is very basic

```
n   = 30
X1 = rnorm(n)
X2 = rnorm(n)
X   = cbind(X1,X2)
K = 3
kmeans.out = kmeans(X, centers=K)
> names(kmeans.out)
[1] "cluster"      "centers"      "totss"      "withinss"
[5] "tot.withinss" "betweenss"    "size"
> kmeans.out$cluster
 [1] 2 2 2 2 2 2 1 1 2 2 3 1 2 1 2 2 2
 3 1 2 2 1 3 2 1 3 3 1 2 3

plot(X, col=(kmeans.out$cluster+1), xlab="", ylab="",
           pch=20, cex=2)
```

K-means in R



K-means in R

Another example

```
x = matrix(rnorm(50*2),ncol=2)
x[1:25,1] = x[1:25,1] + 3
x[1:25,2] = x[1:25,2] -4

kmeans.out = kmeans(x,centers=2,nstart=20)
```

K-means in R

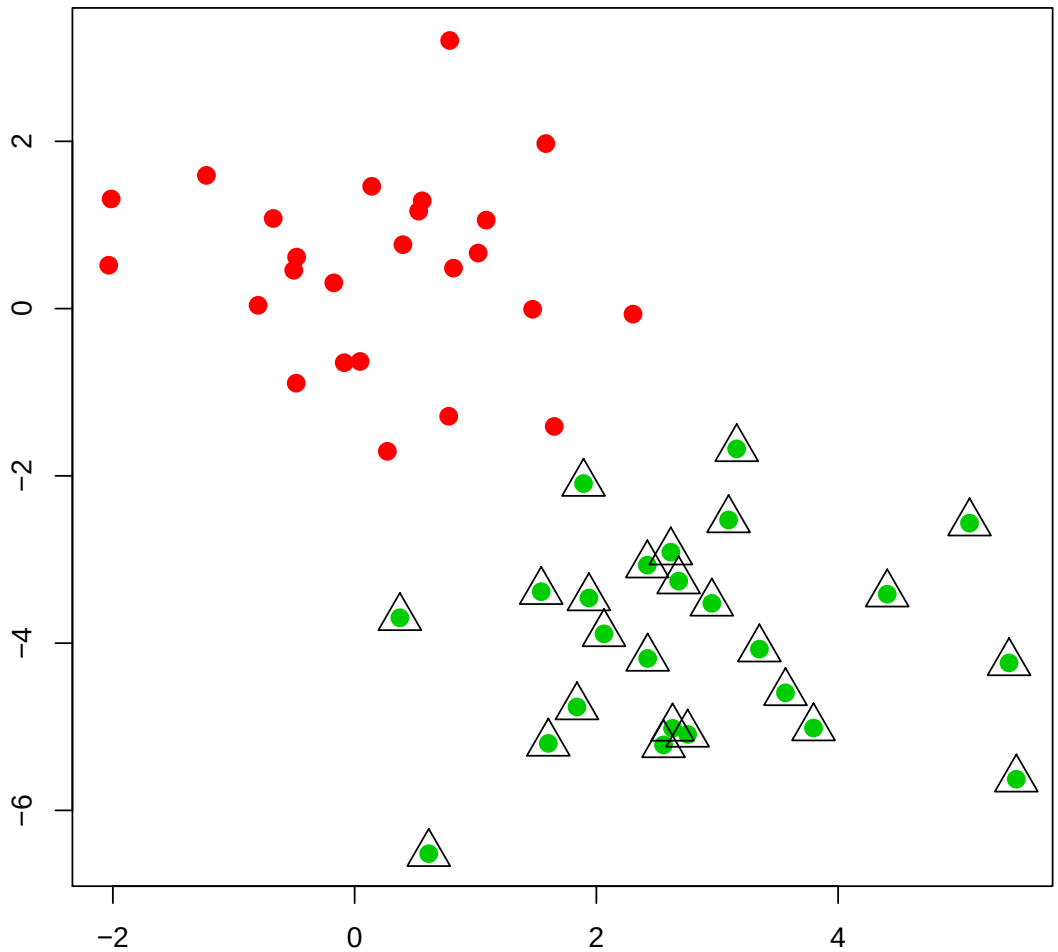


FIGURE: Two clusters (which is the true number)

K-means in R

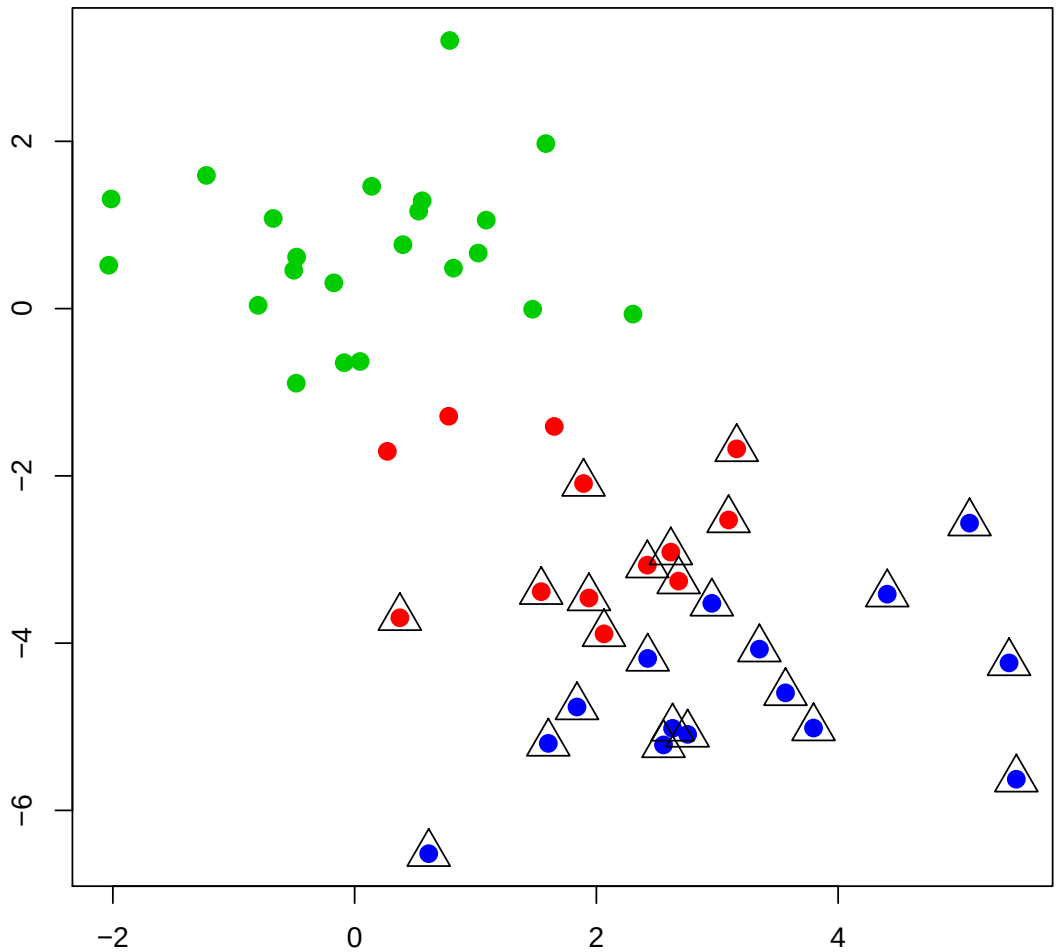


FIGURE: Three clusters

K-means in R: Comparison

R provides several objects in the kmeans output.

$W(C_k)$ is the same as: kmeans(x,centers=K)$withinss

$\sum_{k=1}^{K} W(C_k)$ is the same as: kmeans(x,centers=K)$tot.withinss

```
> kmeans(x,centers=4,nstart=1)$tot.withinss
[1] 19.12722
> kmeans(x,centers=4,nstart=20)$tot.withinss
[1] 18.5526
> kmeans(x,centers=5,nstart=20)$tot.withinss
[1] 12.01893
```

K-means using PCA

We can use the PC scores as the input to cluster as well to get a different look at the data

Note: this is fundamentally different than using PC scores to plot and visually inspect your data.

```
forest    = read.table('../data/forestfires.csv',
                       sep=',',header=T)
forestRed = forest[,-c(3,4,13)]
fires       = forest[,13]
pca.out   = prcomp(forestRed,center=T,scale=T)

cum.sum = cumsum(pca.out$sdev^2/sum(pca.out$sdev^2))
> round(cum.sum,2)
 [1] 0.29 0.44 0.57 0.69 0.79 0.86 0.90 0.95 0.98 1.00
 nComps  = min(which(cum.sum > .9))
> nComps
[1] 7
```

K-means using PCA

```
kmeans.out    = kmeans(pca.out$x[,1:nComps],centers=2,nstart=20)
Y             = rep(1,nrow(forest))
Y[fires > 0] = 2
```

(Here, we are dividing the response fires into two groups, labeling them 1 and 2 to match the output for kmeans)

```
> table(Y,kmeans.out$cluster)

Y      1   2
  1   63 184
  2   51 219
```

K-means using PCA

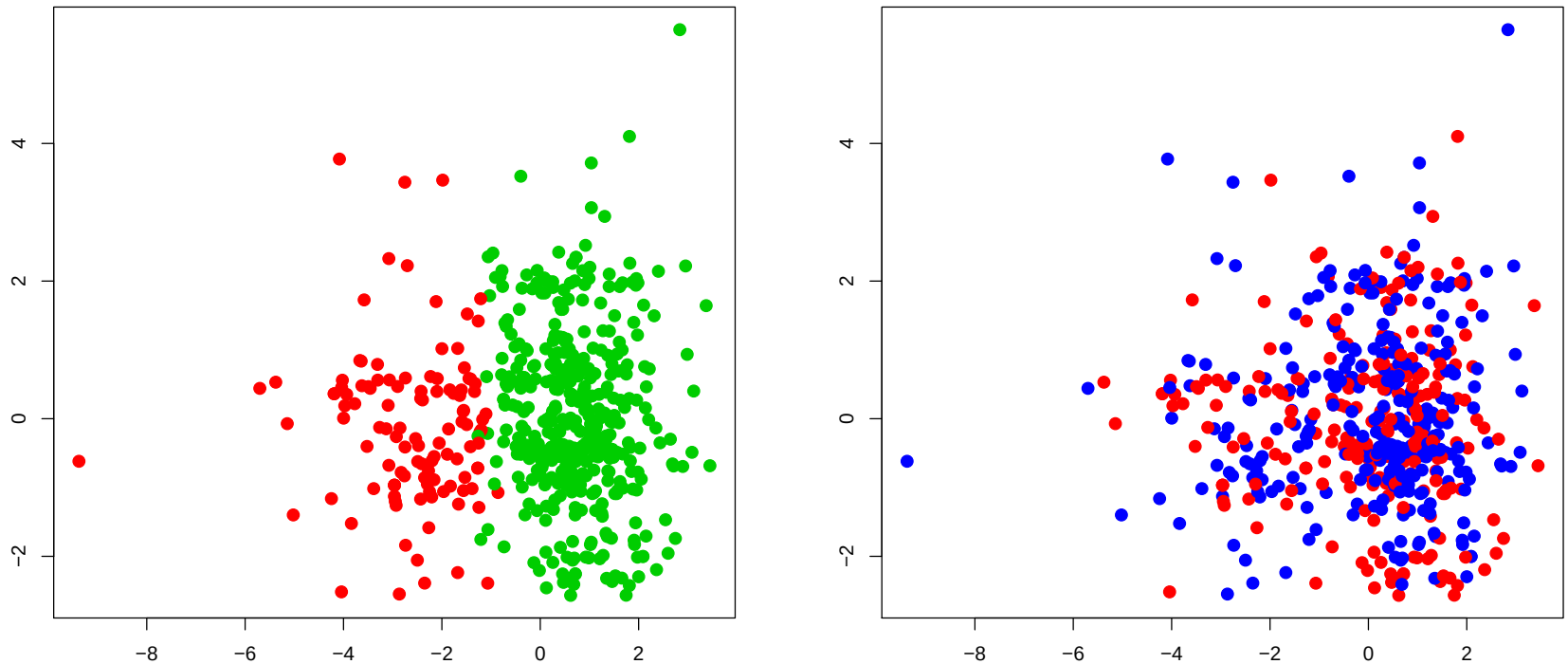


Figure:

Left: Plot of PC scores, colored by cluster.

Right: Plot of PC scores, colored by correct cluster assignment: Red means clustered to 'zero area' when 'positive area' was the label or the opposite. Blue means clustered properly.

Choosing the Number of Clusters

Sometimes, the number of clusters is fixed ahead of time:

- Segmenting a client database into K clusters for K salesmen.
- Compressing an image using vector quantization (K is the compression rate).

Most of the time, it isn't so straight forward. Why is this a hard problem?

- Determining the number of clusters is hard (for humans) unless the data is low dimensional.
- It is just as hard to explain what we are looking for

  (ie: in classification, we wanted a classifier that predicts well. In clustering, we want a clusterer to ... what?)

Choosing the Number of Clusters

Why is it important?

- It might make a big difference (concluding there are $K = 2$ cancer sub-types versus $K = 3$).
- One of the major goals of statistical learning is automatic inference. A good way of choosing K is certainly a part of this.

Reminder: What does K-means do?

Given a number of clusters K, we (approximately) minimize:

$$\sum_{k=1}^{K} W(C_k) = \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} ||X_i - X_{i'}||_2^2.$$

We can rewrite this in terms of the centroids as

$$W(K) = \sum_{k=1}^{K} \sum_{i \in C_k} ||X_i - \overline{X}_k||_2^2,$$

where $\overline{X}_k \in \mathbb{R}^?$ (what is ?).

Reminder: What does K-means do?

Given a number of clusters K, we (approximately) minimize:

$$\sum_{k=1}^{K} W(C_k) = \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} ||X_i - X_{i'}||_2^2.$$

We can rewrite this in terms of the centroids as

$$W(K) = \sum_{k=1}^{K} \sum_{i \in C_k} ||X_i - \overline{X}_k||_2^2,$$

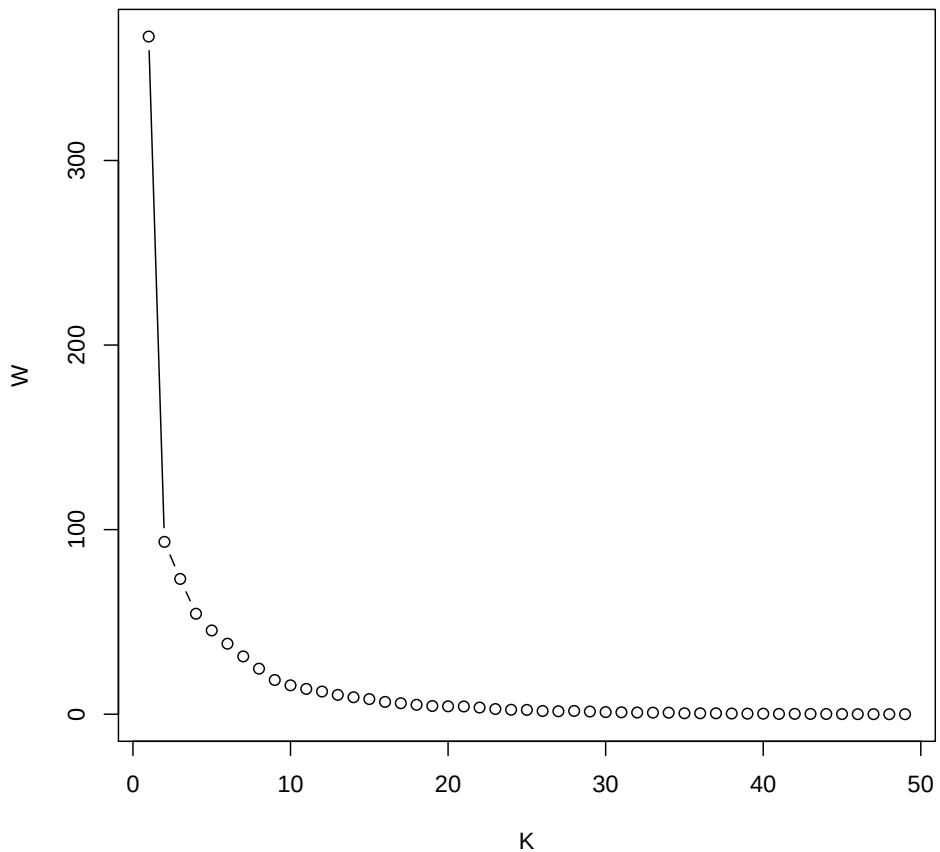
where $\overline{X}_k \in \mathbb{R}^?$ (what is ?).

Answer: $? = p$.

Minimizing W in K

Of course, a lower value of W is better. Why not minimize W?

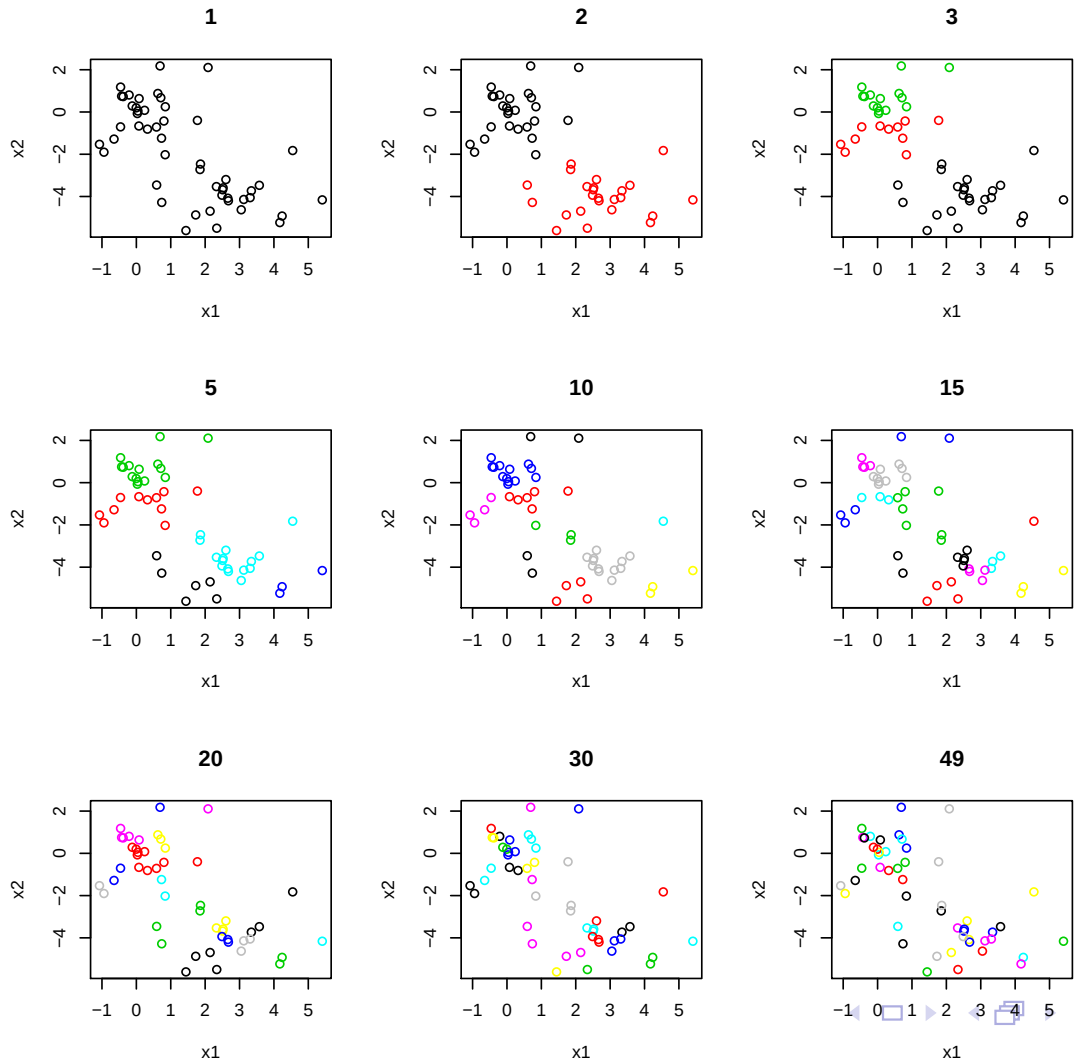
```
plotW = rep(0,49)
for(K in 1:49){
  plotW[K] =  kmeans(x,centers=K,nstart=20)$tot.withinss
}
```



Minimizing W in K

Of course, a lower value of W is better. Why not minimize W?

A look at the cluster solution



Between-cluster variation

Within-cluster variation measures how tightly grouped the clusters are. As we increase K, this will always decrease.

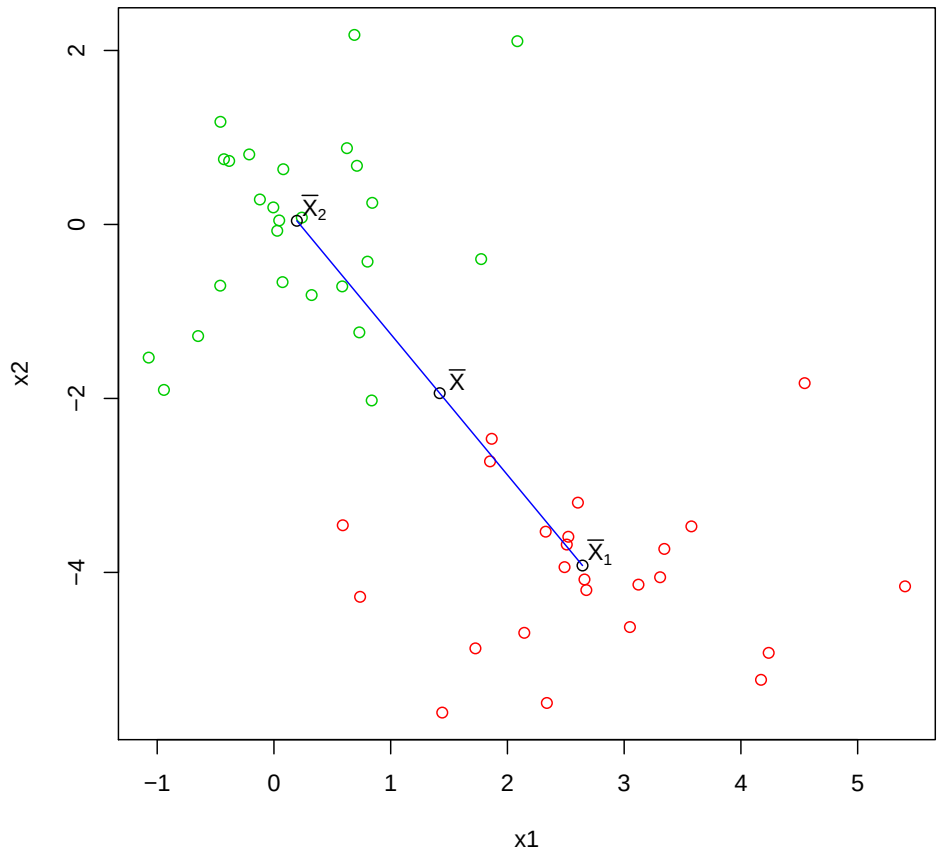
What we are missing is between-cluster variation, ie: how spread apart the groups are

$$B = \sum_{k=1}^{K} |C_k| ||\overline{X}_k - \overline{X}||_2^2,$$

where $|C_k|$ is the number of points in C_k, and $\overline{X}$ is the grand mean of all observations:

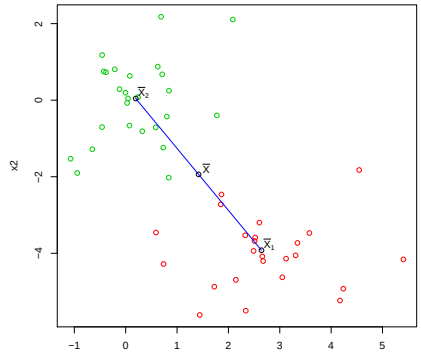
$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

Between-cluster variation: Example



$$B = |C_1| \|\overline{X}_1 - \overline{X}\|_2^2 + |C_2| \|\overline{X}_2 - \overline{X}\|_2^2$$
$$W = \sum_{i \in C_1} |C_1| \|\overline{X}_1 - X_i\|_2^2 + \sum_{i \in C_2} |C_2| \|\overline{X}_2 - X_i\|_2^2$$

R tip detour

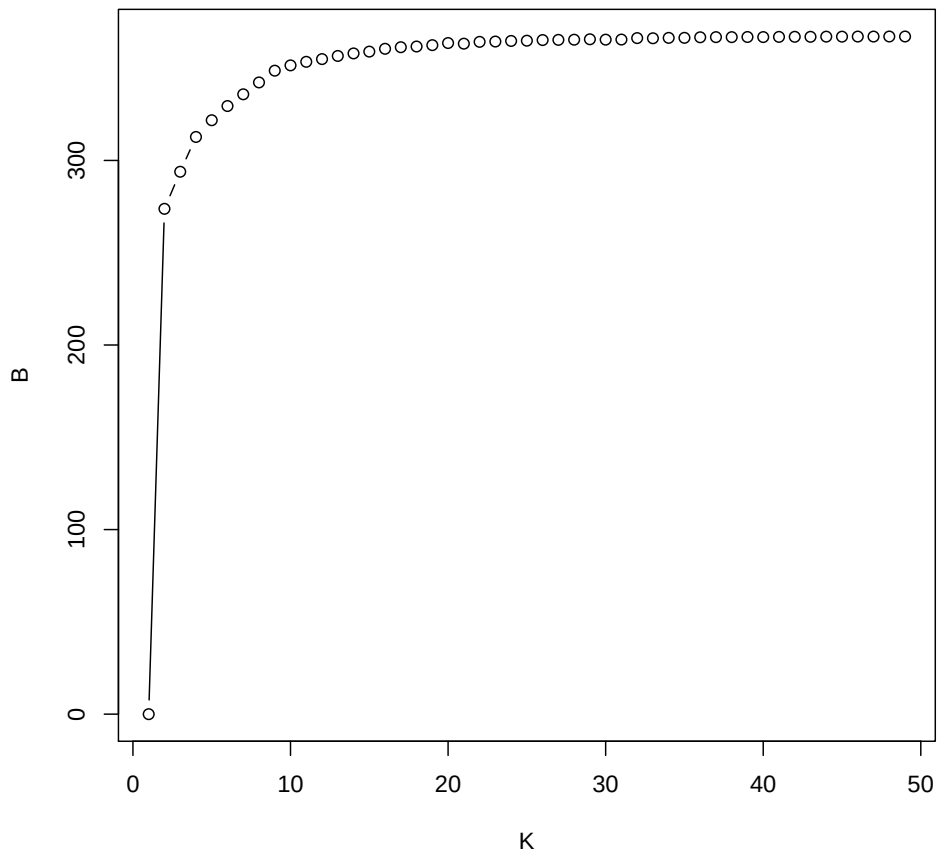



```
x1Bar = apply(x[1:25,],2,mean)
x2Bar = apply(x[26:50,],2,mean)
xBar  = apply(x,2,mean)

plot(x,xlab='x1',ylab='x2',col=kmeans(x,centers=2,nstart=20)$clu
points(x1Bar[1],x1Bar[2])
points(x2Bar[1],x2Bar[2])
points(xBar[1],xBar[2])
segments(x1Bar[1],x1Bar[2],x2Bar[1],x2Bar[2],col='blue')
text(x1Bar[1]+.15,x1Bar[2]+.15,expression(bar(X)[1]))
text(x2Bar[1]+.15,x2Bar[2]+.15,expression(bar(X)[2]))
text(xBar[1]+.15,xBar[2]+.15,expression(bar(X)))
```

Can we just maximize B?

Sadly, no. Just like W can be made arbitrarily small, B will always be increasing with increasing K.



CH INDEX

Ideally, we would like our cluster assignment to simultaneously have small W and large B.

This is the idea behind *CH* index. For clustering assignments coming from K clusters, we record *CH* score:

$$CH(K) = \frac{B(K)/(K-1)}{W(K)/(n-k)}$$

To choose K, pick some maximum number of clusters to be considered ($K_{\text{max}} = 20$, for example) and choose the value of K with the { smallest, largest } *CH* score:

CH index

Ideally, we would like our cluster assignment to simultaneously have small W and large B.

This is the idea behind *CH* index. For clustering assignments coming from K clusters, we record *CH* score:

$$CH(K) = \frac{B(K)/(K-1)}{W(K)/(n-k)}$$

To choose K, pick some maximum number of clusters to be considered ($K_{\max} = 20$, for example) and choose the value of K with the { smallest, largest } *CH* score:

$$\hat{K} = \underset{K \in \{2,\ldots,K_{\max}\}}{\arg\max} \; CH(K).$$

Note: *CH* is undefined for $K = 1$.

CH INDEX

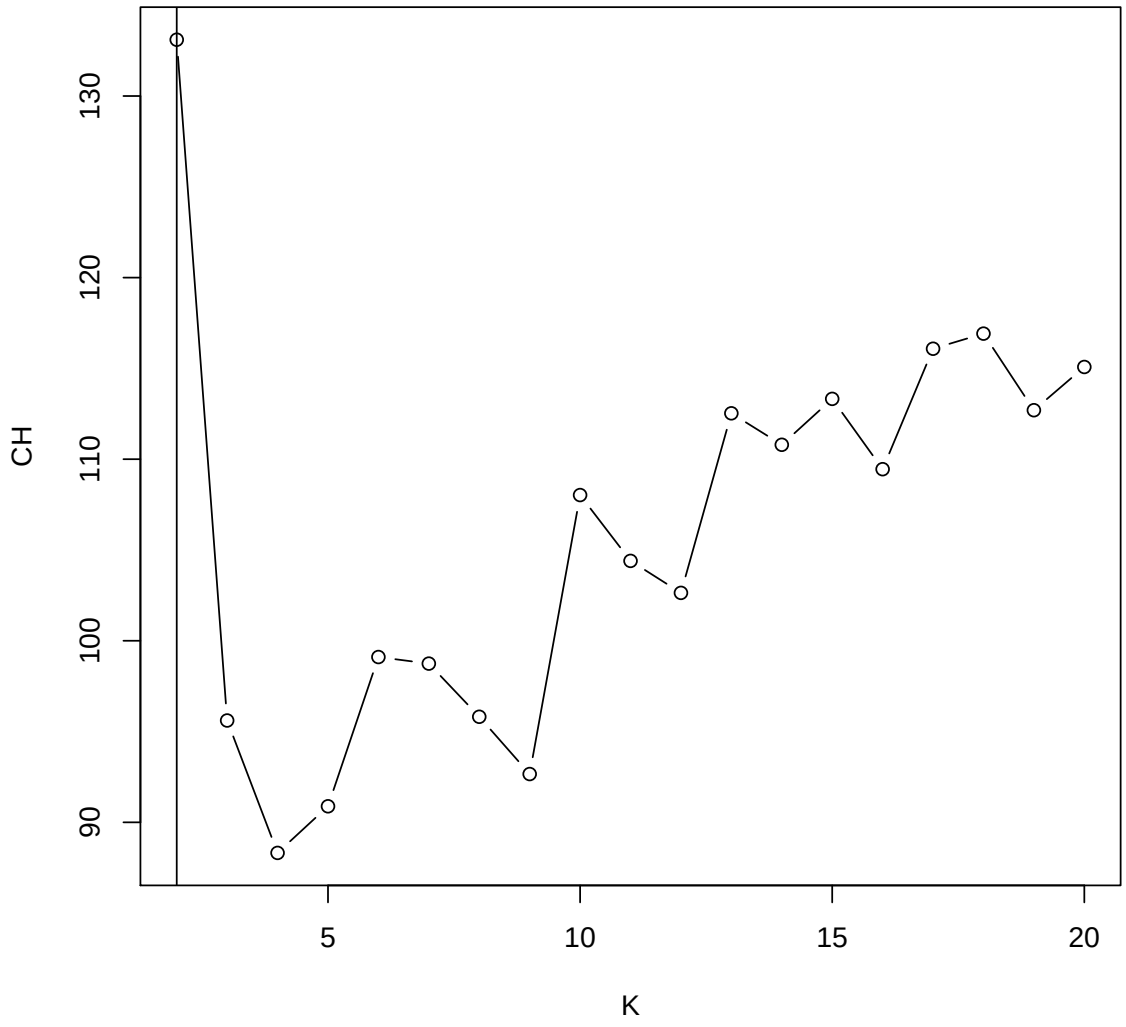
```
ch.index = function(x,kmax,iter.max=100,nstart=10,
                        algorithm="Lloyd")
{
  ch = numeric(length=kmax-1)
  n = nrow(x)
  for (k in 2:kmax) {
    a = kmeans(x,k,iter.max=iter.max,nstart=nstart,
               algorithm=algorithm)
    w = a$tot.withinss
    b = a$betweenss
   ch[k-1] = (b/(k-1))/(w/(n-k))
  }
  return(list(k=2:kmax,ch=ch))
}
```

Revisting simulated example

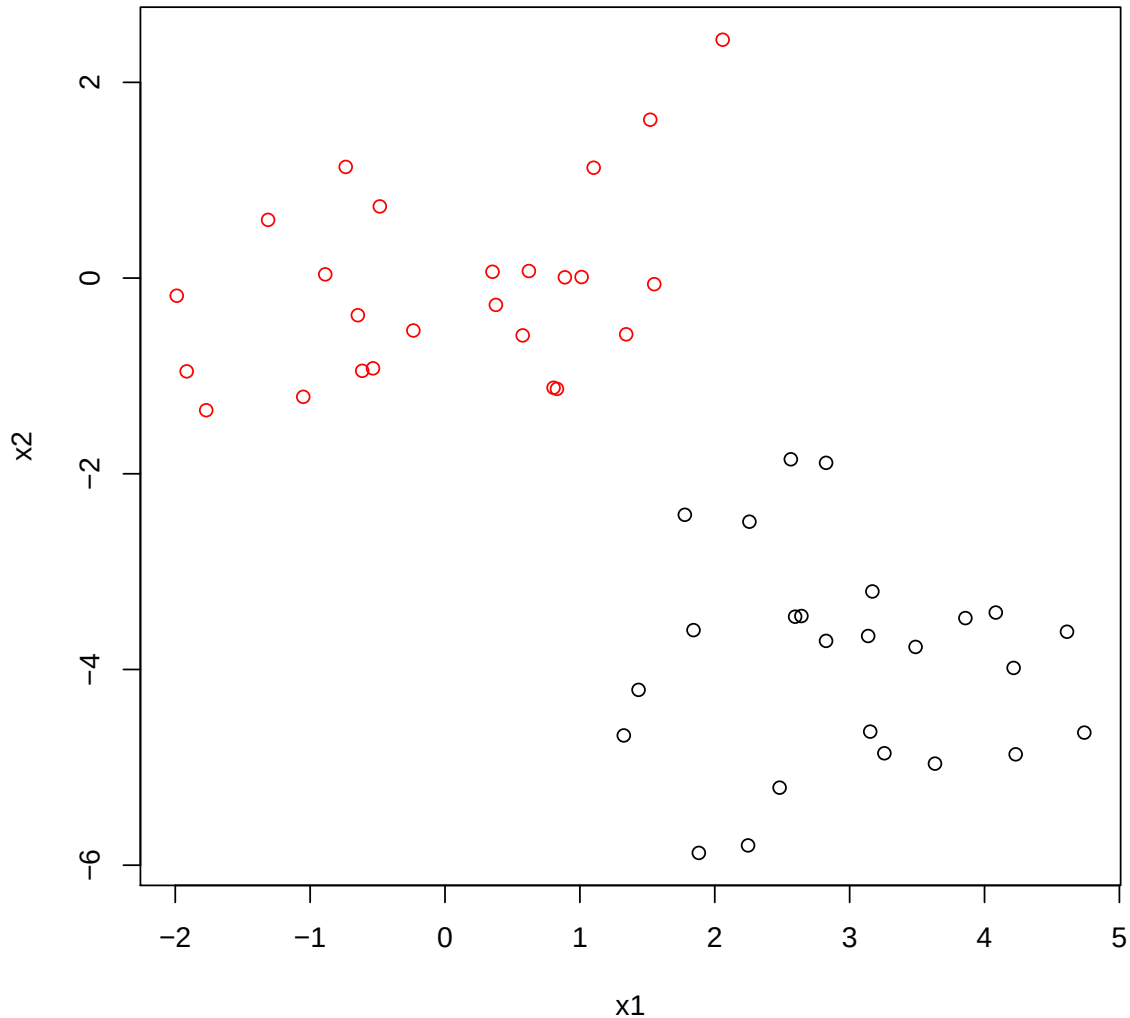
```
x = matrix(rnorm(50*2),ncol=2)
x[1:25,1] = x[1:25,1] + 3
x[1:25,2] = x[1:25,2] -4
```

We want to cluster this data set using K-means with K chosen via CH index.

CH PLOT



Corresponding solution



Alternate approach: Gap statistic

While true that $W(K)$ keeps dropping in K, how much it drops might be informative.

The gap statistic is based on this idea. We compare the observed within-cluster variation $W(K)$ to the within-cluster variation we would observe if the data were uniformly distributed $W_{unif}(K)$.

$$Gap(K) = \log W(K) - \log W_{unif}(K)$$

After simulating many $\log W_{unif}(K)$, we compute its standard deviation $s(K)$. Then, we choose K by

$$\hat{K} = \min\{K \in \{1, \ldots, K_{\max}\} : Gap(K) \geq Gap(K+1) - s(K+1)\}.$$

Gap statistic: Summary

I don't want to dwell too long on $Gap(K)$, other than to say the following:

- As $Gap(K)$ is defined for $K = 1$, it can pick the null model (all observations in 1 cluster).
- In fact, this is what it is best at (why?).
- It can be found using the `R` package `SAGx` or the package `lga`. In both cases the function is called `gap`.
- Beware: these functions are poorly documented. It's unclear what clustering method/algorithms they are using.