

# Técnicas de Clustering - 3



Cuántos clusters?





# Introducción

- En algunas aplicaciones el número de clusters está predefinido
  - Ejemplo: Hay que dividir  $n$  ciudades de Argentina entre  $k$  vendedores
- La mayoría de las veces no
- Lamentablemente, los algoritmos siempre devuelven una solución, aunque no tenga sentido
- El número de clusters es otra información que tenemos que sacar de los datos muchas veces

# Introducción



- Cuántos clusters hay en un dataset es una pregunta difícil.
  - No hay “verdad” contra la que comparar
  - No hay métodos con fuerte teoría detrás
- La mayoría de los métodos son empíricos

# Métodos a discutir

- El “salto” en la función costo
- Gap statistic
- Estabilidad





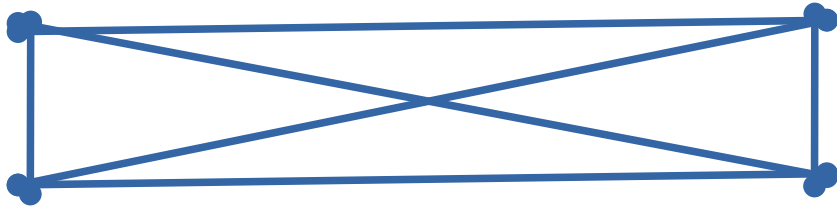
# Método del salto

- Un criterio general para evaluar la calidad de la solución en la suma de las distancias dentro del cluster  $W_k$ 
  - Es lo que se minimiza en la mayoría de los métodos



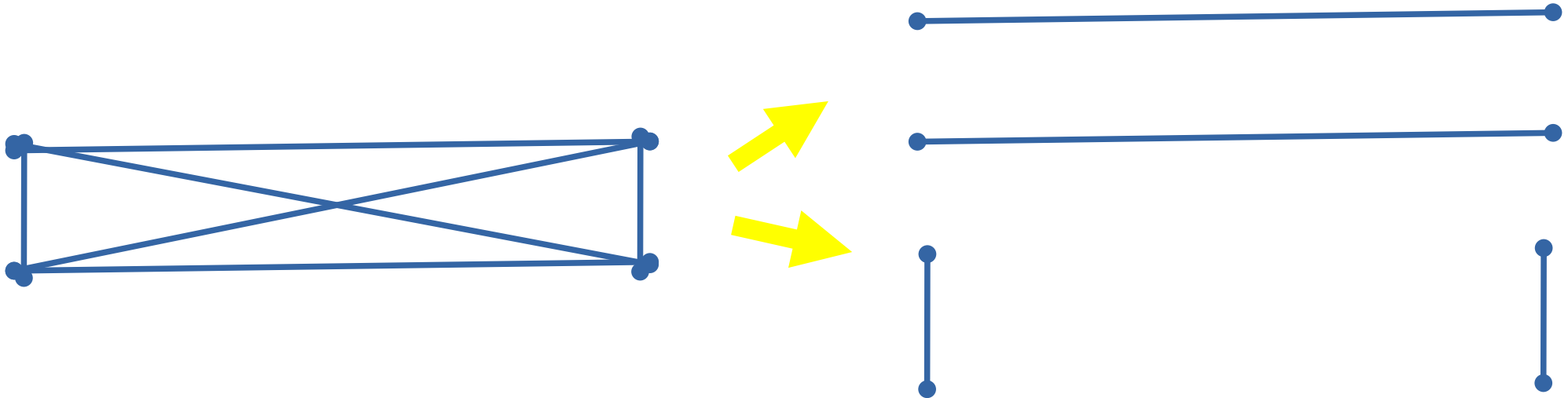
# Método del salto

- Un criterio general para evaluar la calidad de la solución en la suma de las distancias dentro del cluster  $W_k$ 
  - Es lo que se minimiza en la mayoría de los métodos



# Método del salto

- Un criterio general para evaluar la calidad de la solución en la suma de las distancias dentro del cluster  $W_k$ 
  - Es lo que se minimiza en la mayoría de los métodos





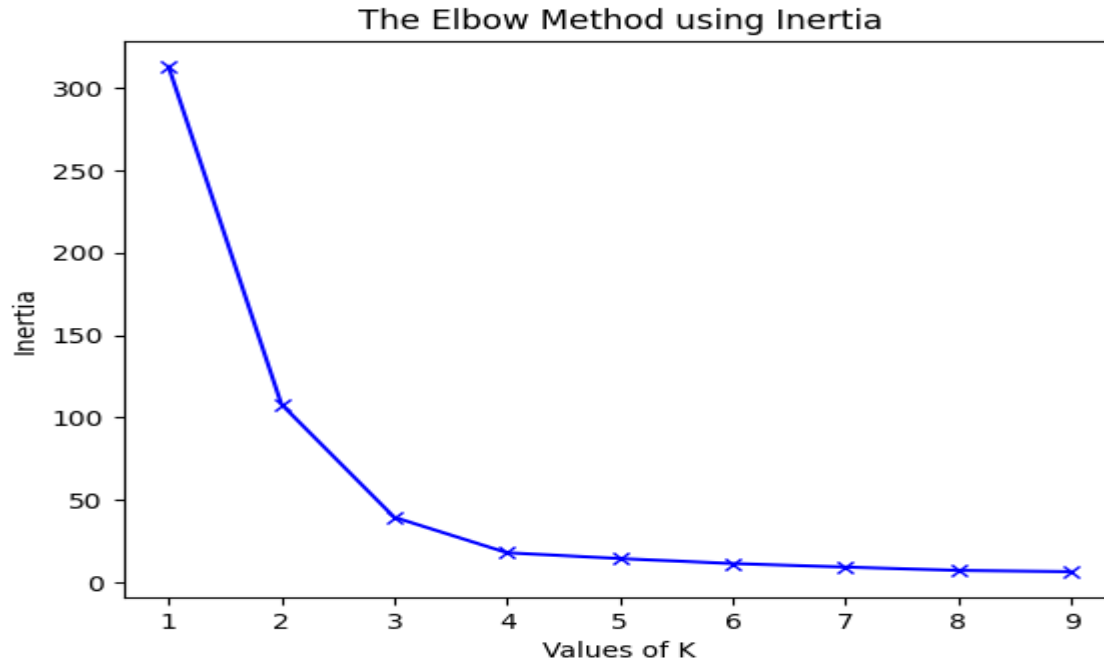


# Método del salto

- Un criterio general para evaluar la calidad de la solución en la suma de las distancias dentro del cluster  $W_k$ 
  - Es lo que se minimiza en la mayoría de los métodos
- $W_k$  decrece siempre. No sirve buscar el mínimo.
- Pero decrece más cuando separa 2 clusters verdaderos que cuando parte en 2 un cluster “natural”
  - Elimina distancias “largas” en el primer caso



# Método del salto



- Hay que buscar un “lomo” o un cambio completo de pendiente

# Método del salto

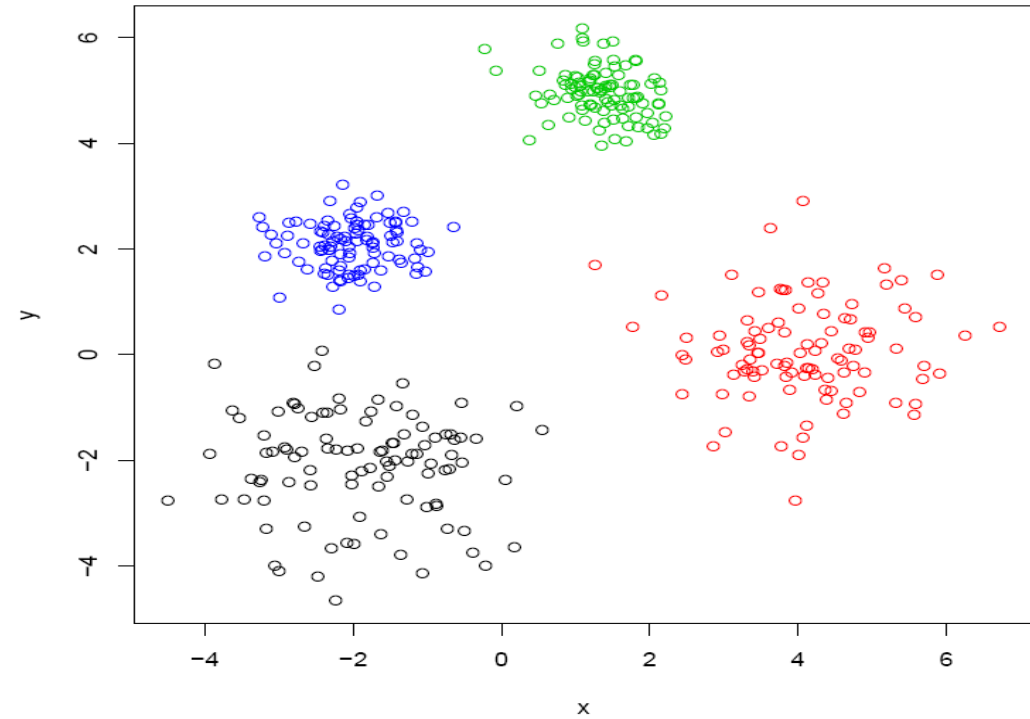


```
#cuatro clusters de dist. gaussianas  
tot.puntos<-100  
  
gap=2  
  
x<-rnorm(tot.puntos,mean=-gap)  
y<-rnorm(tot.puntos,mean=-gap)  
gaussianas<-cbind(x,y,rep(1,length(x)))  
  
x<-rnorm(tot.puntos,mean=2*gap)  
y<-rnorm(tot.puntos,mean=0)  
gaussianas<-rbind(gaussianas,cbind(x,y,rep(2,length(x))))  
  
x<-rnorm(tot.puntos,mean=0.7*gap,sd=0.5)  
y<-rnorm(tot.puntos,mean=2.5*gap,sd=0.5)  
gaussianas<-rbind(gaussianas,cbind(x,y,rep(3,length(x))))  
  
x<-rnorm(tot.puntos,mean=-gap,sd=0.5)  
y<-rnorm(tot.puntos,mean=gap,sd=0.5)  
gaussianas<-rbind(gaussianas,cbind(x,y,rep(4,length(x))))  
  
plot(gaussianas[,1:2],col=gaussianas[,3])
```

# Método del salto



```
#cuatro clusters de dist. gaussianas  
tot.puntos<-100  
gap=2  
x<-rnorm(tot.puntos,mean=-gap)  
y<-rnorm(tot.puntos,mean=-gap)  
gaussianas<-cbind(x,y,rep(1,length(x)))  
x<-rnorm(tot.puntos,mean=2*gap)  
y<-rnorm(tot.puntos,mean=0)  
gaussianas<-rbind(gaussianas,cbind(x,y,rep(2,length(x))))  
x<-rnorm(tot.puntos,mean=0.7*gap,sd=0.5)  
y<-rnorm(tot.puntos,mean=2.5*gap,sd=0.5)  
gaussianas<-rbind(gaussianas,cbind(x,y,rep(3,length(x))))  
x<-rnorm(tot.puntos,mean=-gap,sd=0.5)  
y<-rnorm(tot.puntos,mean=gap,sd=0.5)  
gaussianas<-rbind(gaussianas,cbind(x,y,rep(4,length(x))))  
plot(gaussianas[,1:2],col=gaussianas[,3])
```



# Método del salto



#Ejemplo: 4 Gaussianas

```
wg = np.zeros(10)
```

```
for k in range(2, 11):
```

```
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=42)
```

```
    kmeans.fit(X)
```

```
    wg[k-1] = kmeans.inertia_ # inertia_ = suma de cuadrados dentro de clusters
```

# Método del salto

#Ejemplo: 4 Gaussianas

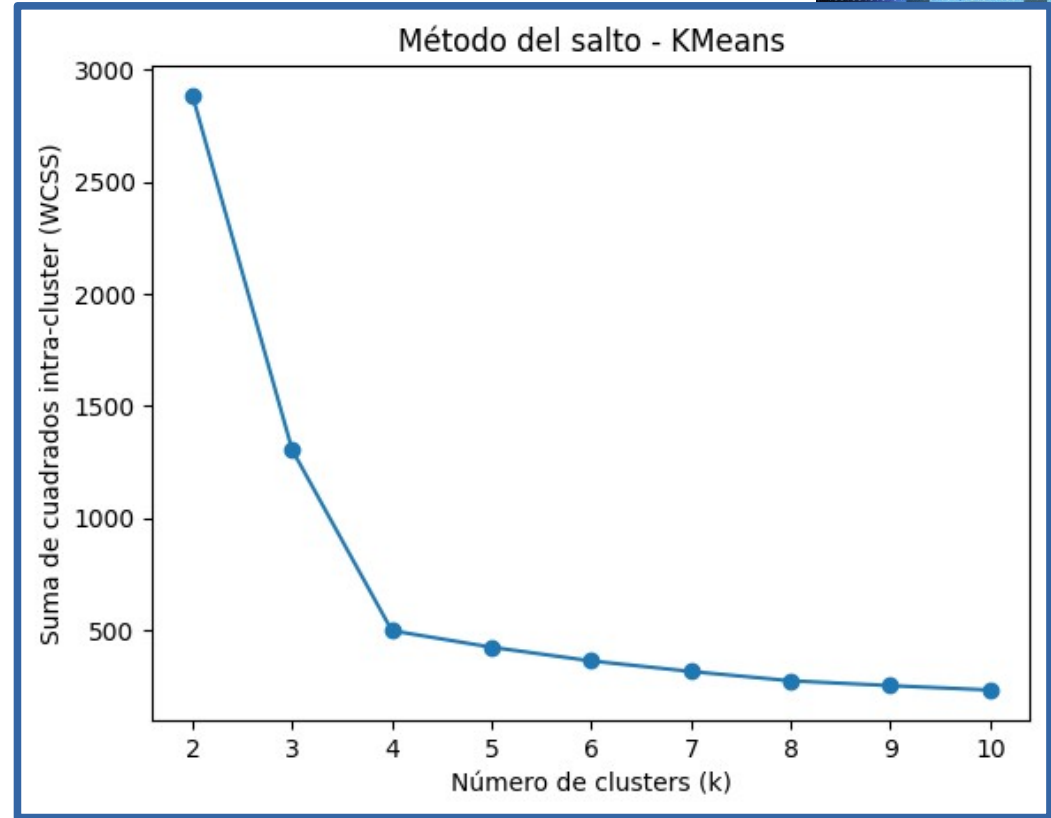
```
wg = np.zeros(10)
```

```
for k in range(2, 11):
```

```
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=42)
```

```
    kmeans.fit(X)
```

```
    wg[k-1] = kmeans.inertia_ # inertia_ = suma de cuadrados dentro de clusters
```



# Método del salto



#Ejemplo: Iris

```
from sklearn.datasets import load_iris
```

```
iris = load_iris()
```

```
X = iris.data
```

#se ve el quiebre para k=3, menos claro

# Método del salto

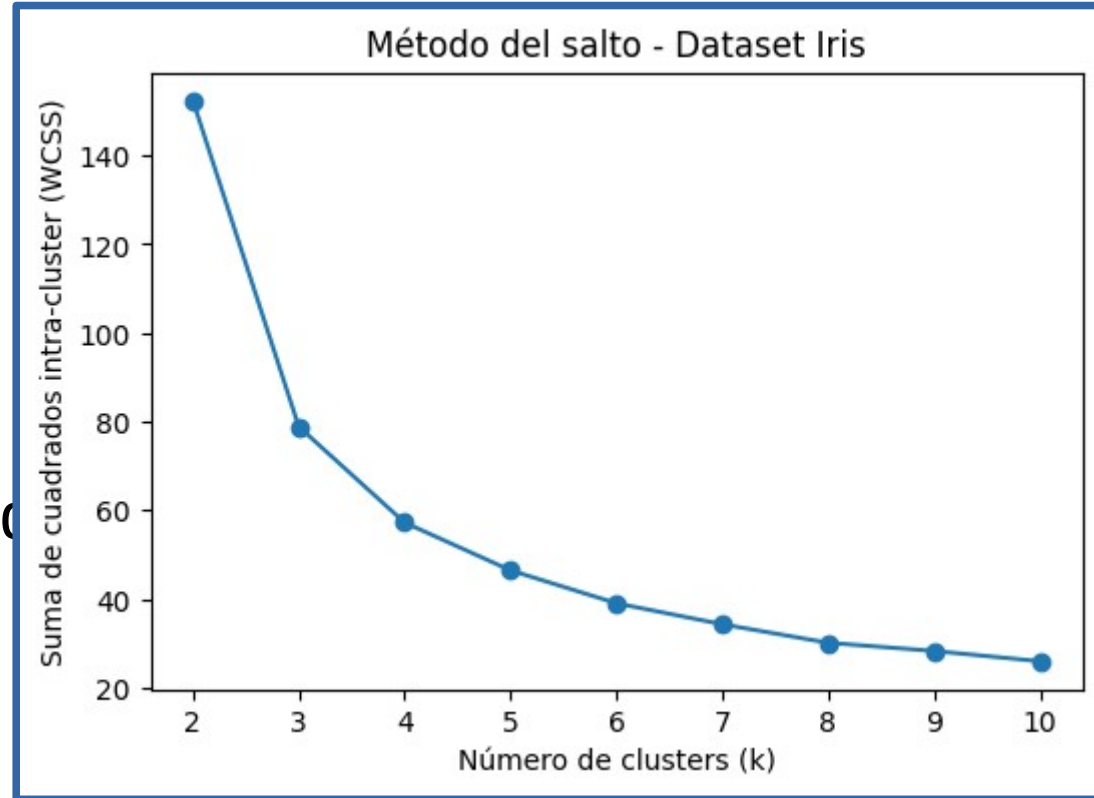
#Ejemplo: Iris

```
from sklearn.datasets import load_iris
```

```
iris = load_iris()
```

```
X = iris.data
```

#se ve el quiebre para k=3, menos claro





# Método del salto



```
#Ejemplo: Uniforme R
x<-rnorm(4*tot.puntos,)
y<-rnorm(4*tot.puntos)
gaussianas<-cbind(x,y,rep(1:4,tot.puntos))
plot(gaussianas[,1:2],col=gaussianas[,3])
wn=rep(0.0,10)
for(k in 2:10) wn[k]<-sum(kmeans(gaussianas[,1:2],k,nsta=10)$withinss)
matplot(2:10,wn[-1],type='b')
#no se ve muy diferente a IRIS
```

# Método del salto

#Ejemplo: Uniforme

```
x<-rnorm(4*tot.puntos,)
```

```
y<-rnorm(4*tot.puntos)
```

```
gaussianas<-cbind(x,y,rep(1:4,tot.puntos))
```

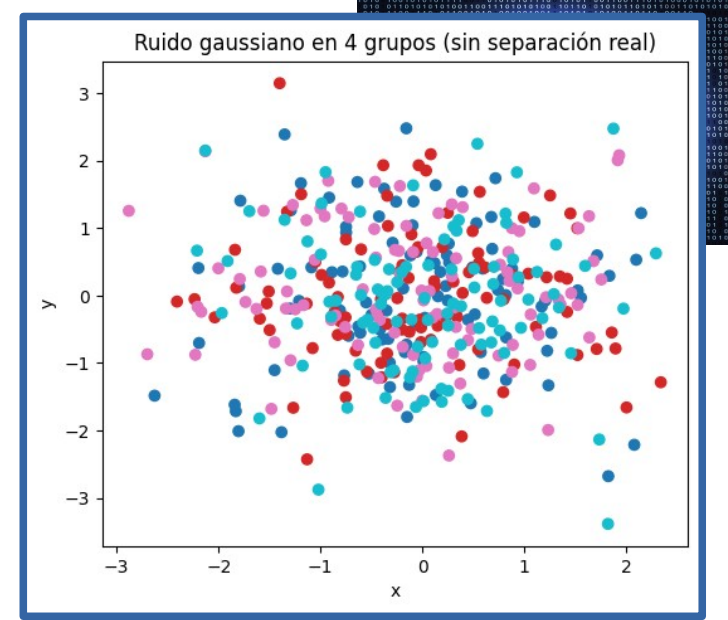
```
plot(gaussianas[,1:2],col=gaussianas[,3])
```

```
wn=rep(0.0,10)
```

```
for(k in 2:10) wn[k]<-sum(kmeans(gaussianas[,1:2],k,nsta=10)$withinss)
```

```
matplot(2:10,wn[-1],type='b')
```

#no se ve muy diferente a IRIS



# Método del salto

#Ejemplo: Uniforme

```
x<-rnorm(4*tot.puntos,)
```

```
y<-rnorm(4*tot.puntos)
```

```
gaussianas<-cbind(x,y,rep(1:4,tot.puntos))
```

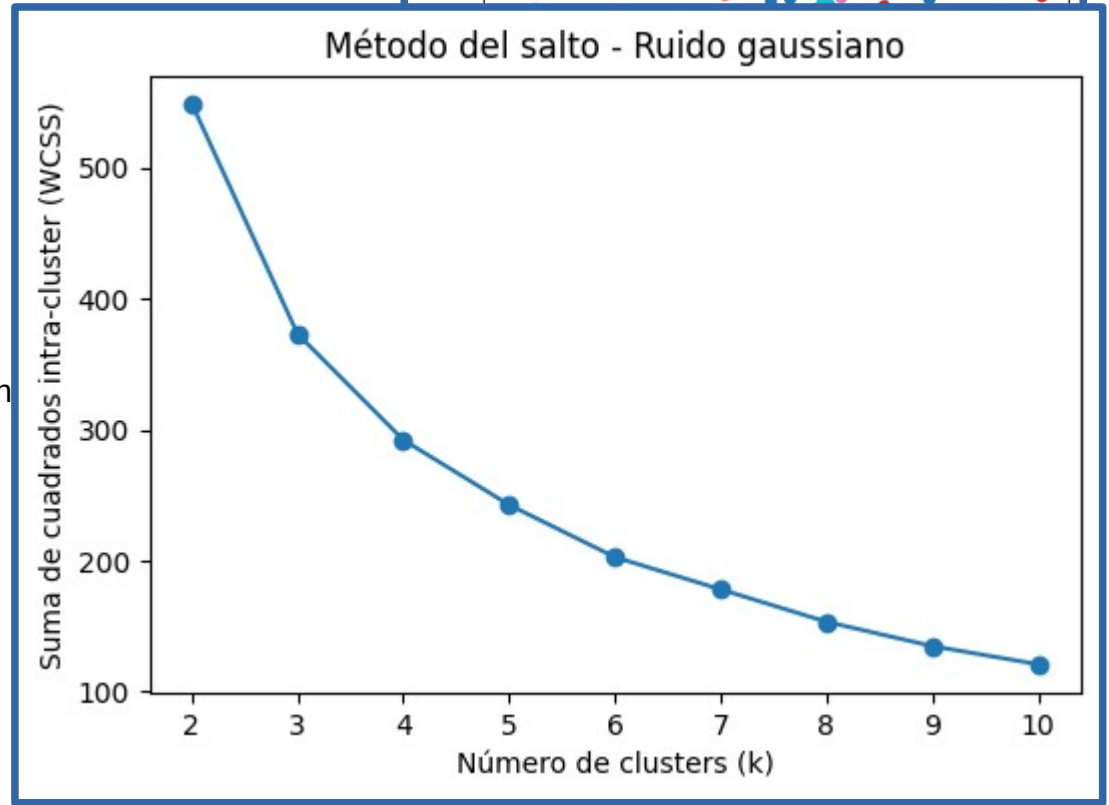
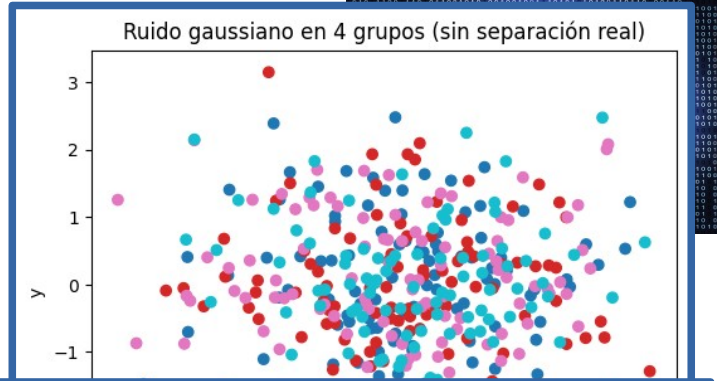
```
plot(gaussianas[,1:2],col=gaussianas[,3])
```

```
wn=rep(0.0,10)
```

```
for(k in 2:10) wn[k]<-sum(kmeans(gaussianas[,1:2],k,nsta=10)$within
```

```
matplot(2:10,wn[-1],type='b')
```

#no se ve muy diferente a IRIS



# Método del salto



- Suele no ser una medida confiable usada directamente.
- No detecta la falta de clusters.



# Gap statistic

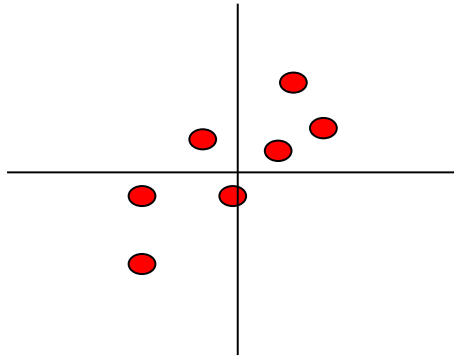
- Trata de resolver los problemas del salto
- Desarrollado por Tibshirani, Walther y Hastie, *J. Royal Statistical Soc. B*, 2001.
- Idea: comparar la curva anterior con la curva que da una distribución uniforme.
- Cuantificar el salto: Buscar la primer diferencia significativa entre las dos curvas (primer gap)



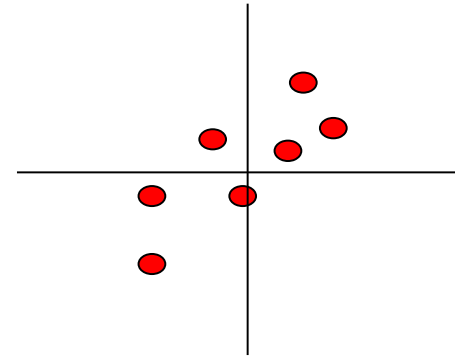
# Gap - Referencia

- Dos formas de generar la referencia:
  - a) Tomar una distribución uniforme que ocupe el mismo hiperrectángulo que la original

a)



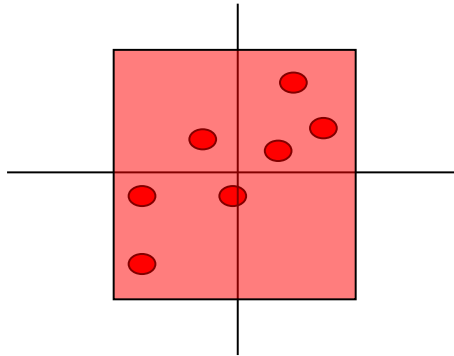
b)



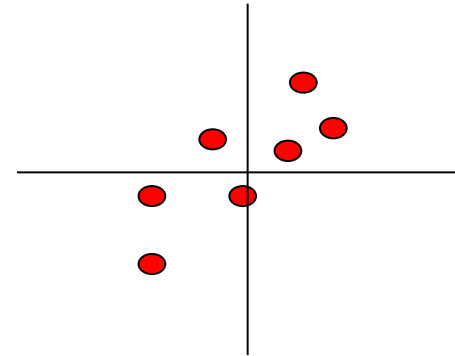
# Gap - Referencia

- Dos formas de generar la referencia:
  - a) Tomar una distribución uniforme que ocupe el mismo hiperrectángulo que la original

a)

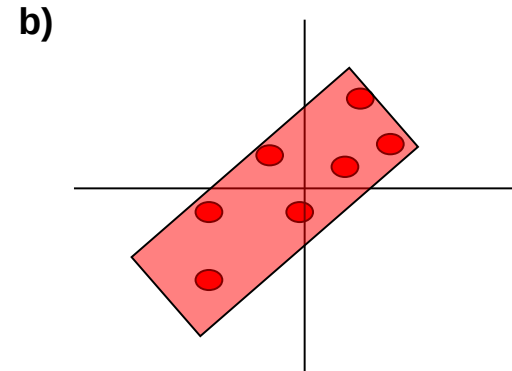
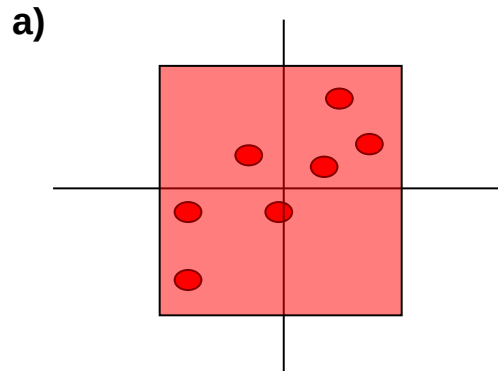


b)



# Gap - Referencia

- Dos formas de generar la referencia:
  - a) Tomar una distribución uniforme que ocupe el mismo hiperrectángulo que la original
  - b) Hacer lo mismo pero sobre una PCA de los datos.





# Algoritmo

## Computation of the Gap statistic

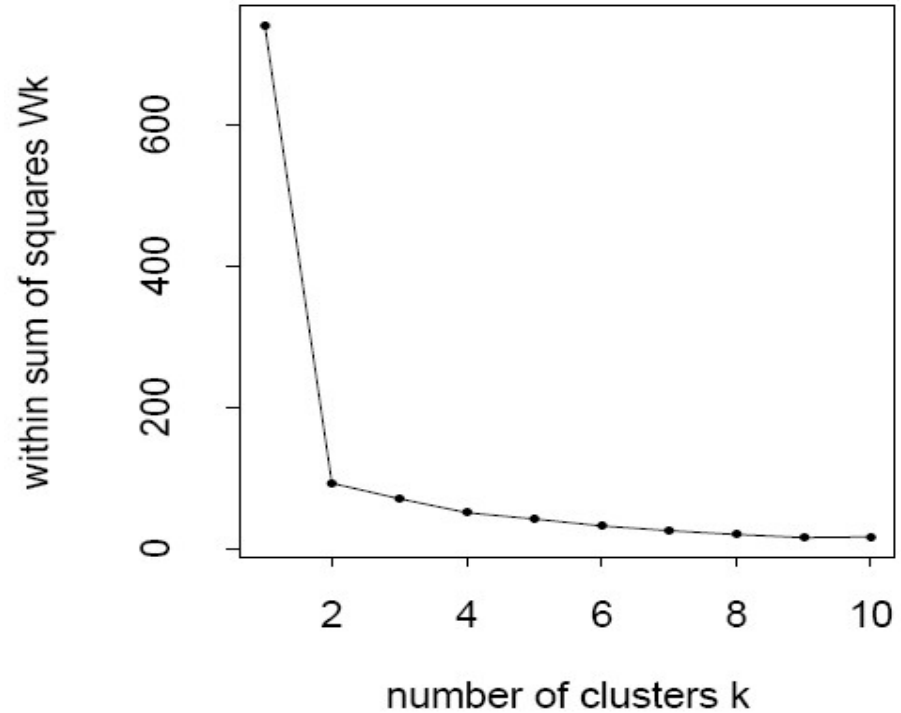
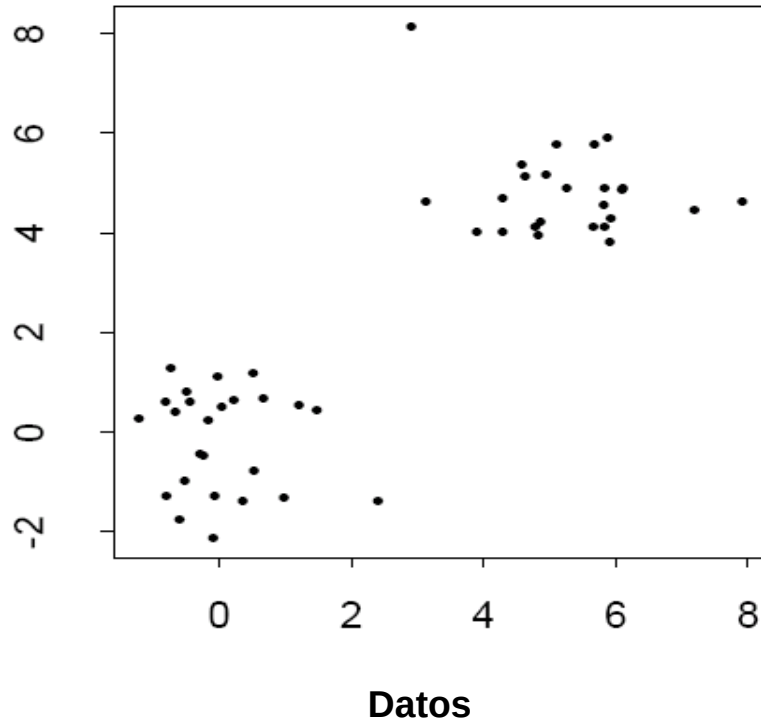
1. Cluster the observed data, varying the total number of clusters from  $k = 1, 2, \dots, K$ , giving within dispersion measures  $W_k, k = 1, 2, \dots, K$ .
2. Generate  $B$  reference datasets, using the uniform prescription (a) or (b) above, and cluster each one giving within dispersion measures  $W_{kb}^*, b = 1, 2, \dots, B, k = 1, 2, \dots, K$ . Compute the (estimated) Gap statistic:

$$\text{Gap}(k) = (1/B) \sum_b \log(W_{kb}^*) - \log(W_k)$$

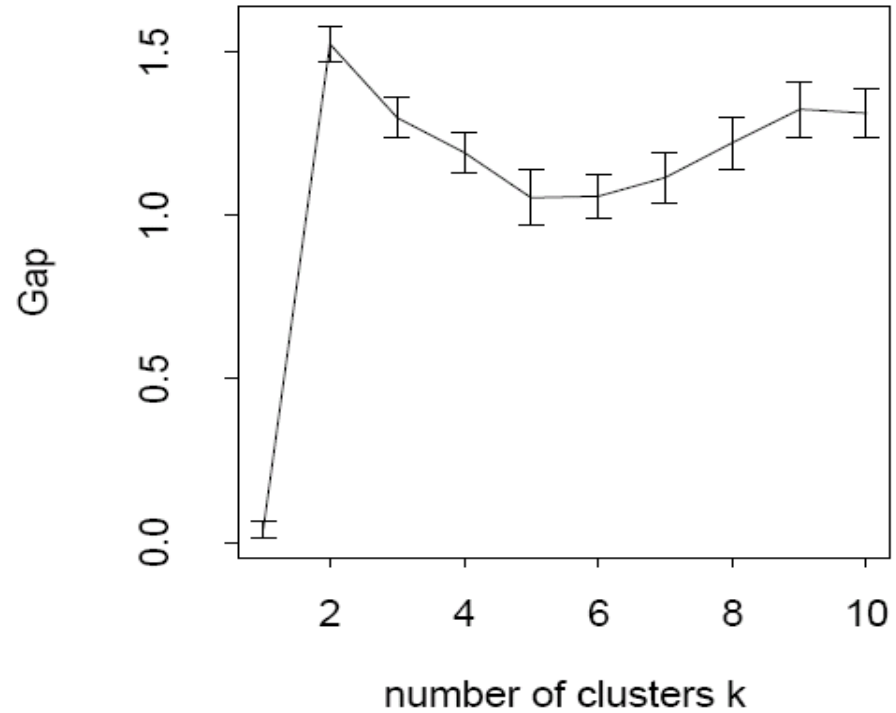
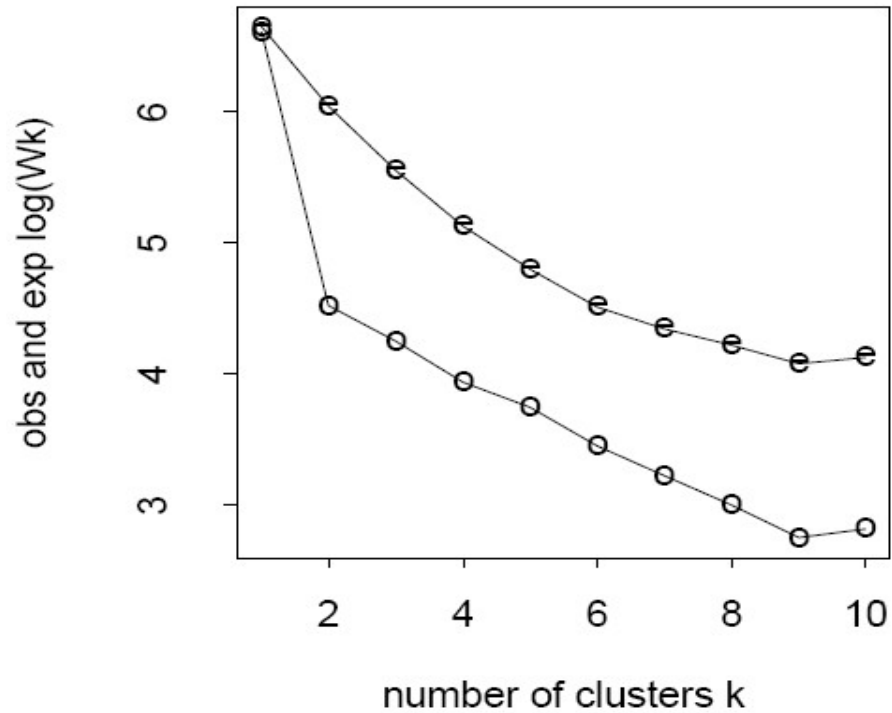
3. Let  $\bar{l} = (1/B) \sum_b \log(W_{kb}^*)$ , compute the standard deviation  $\text{sd}_k = [(1/B) \sum_b (\log(W_{kb}^*) - \bar{l})^2]^{1/2}$ , and define  $s_k = \text{sd}_k \sqrt{1 + 1/B}$ . Finally choose the number of clusters via

$$\hat{k} = \text{smallest } k \text{ such that } \text{Gap}(k) \geq \text{Gap}(k+1) - s_{k+1}$$

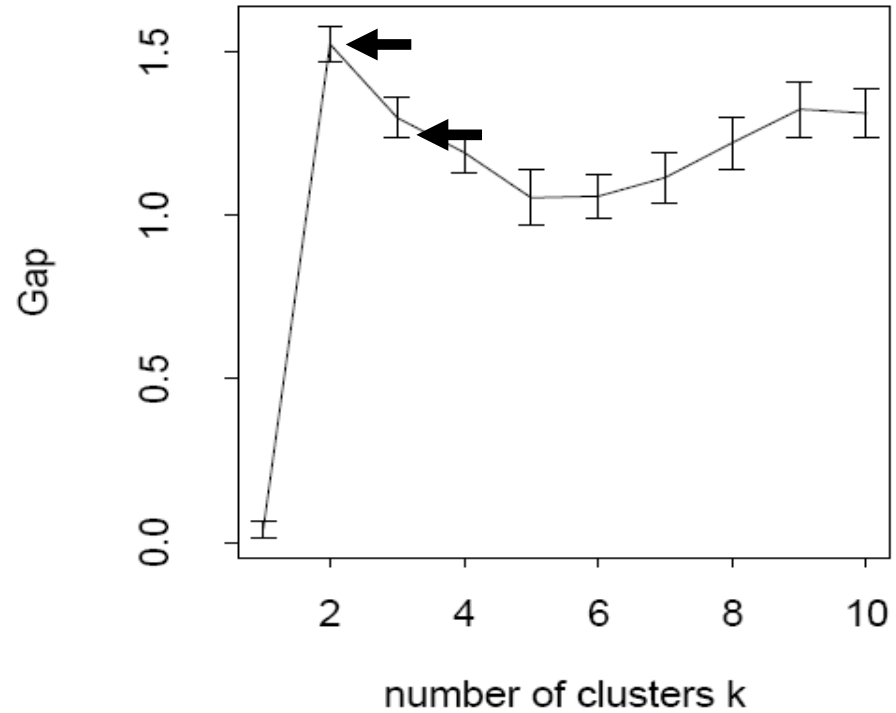
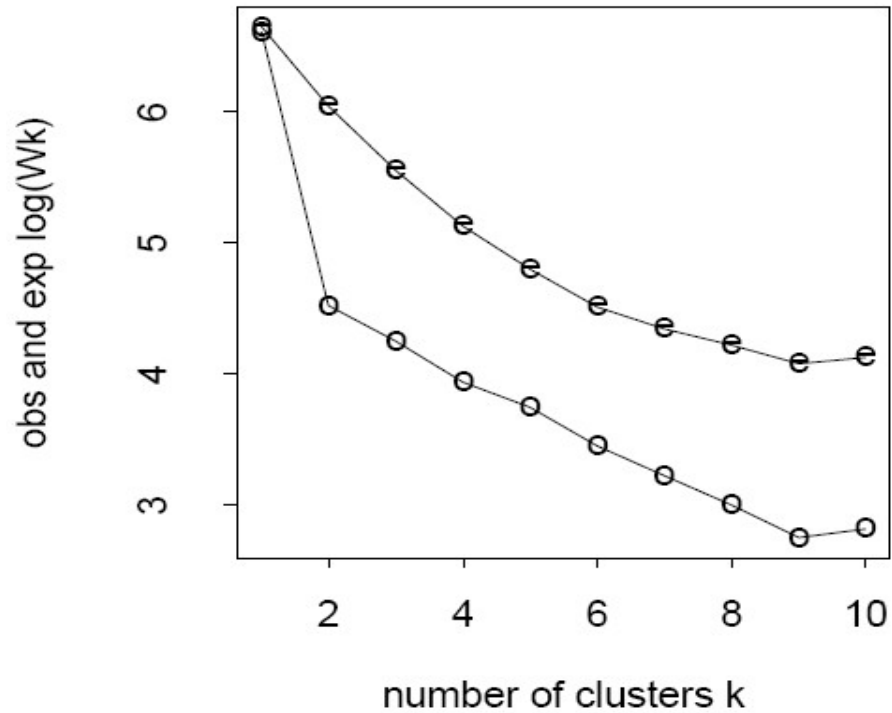
# Ejemplo: 2 clusters



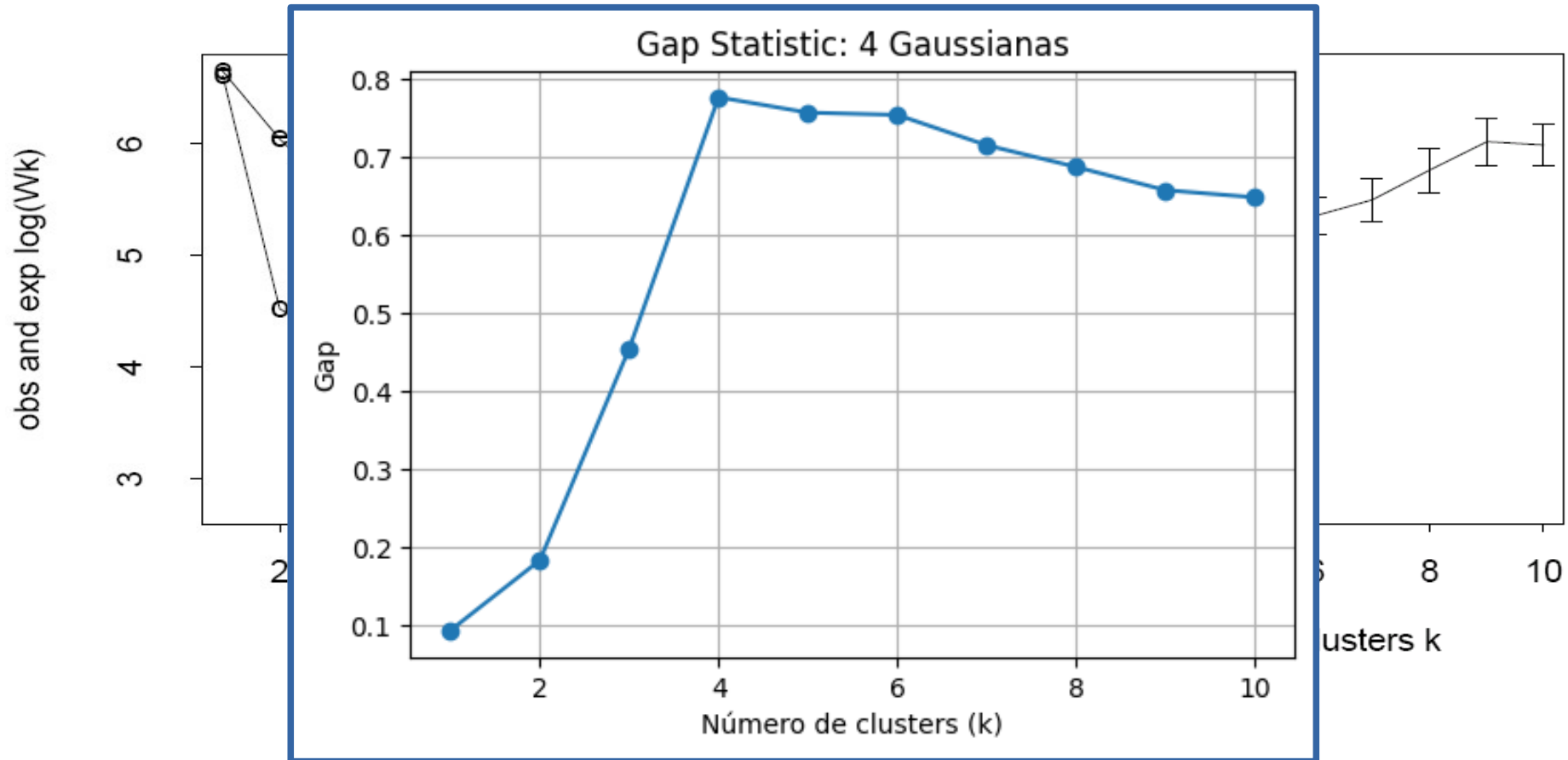
# Ejemplo: 2 clusters



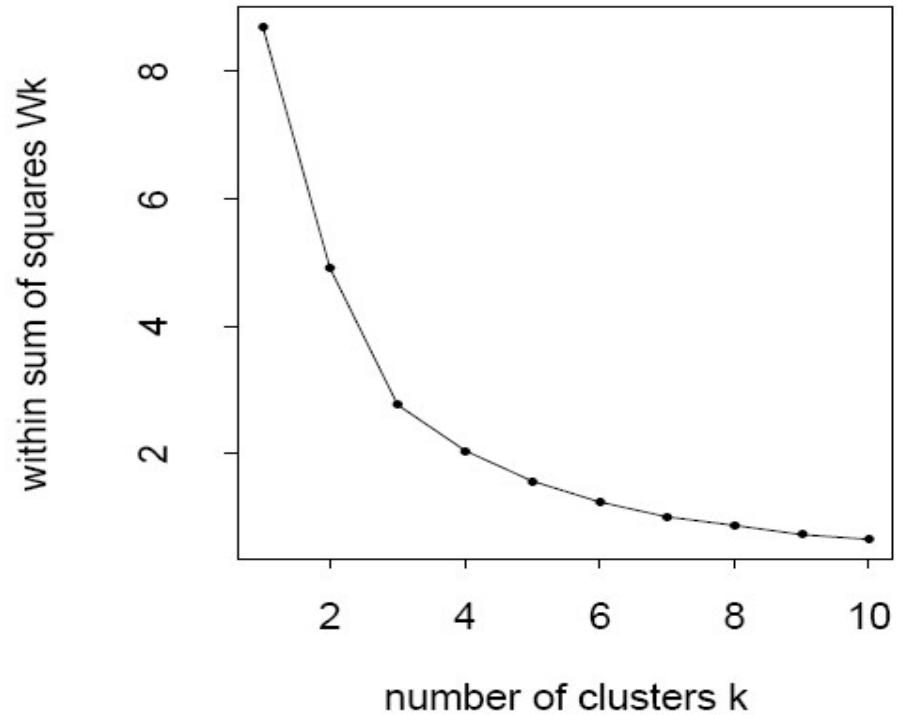
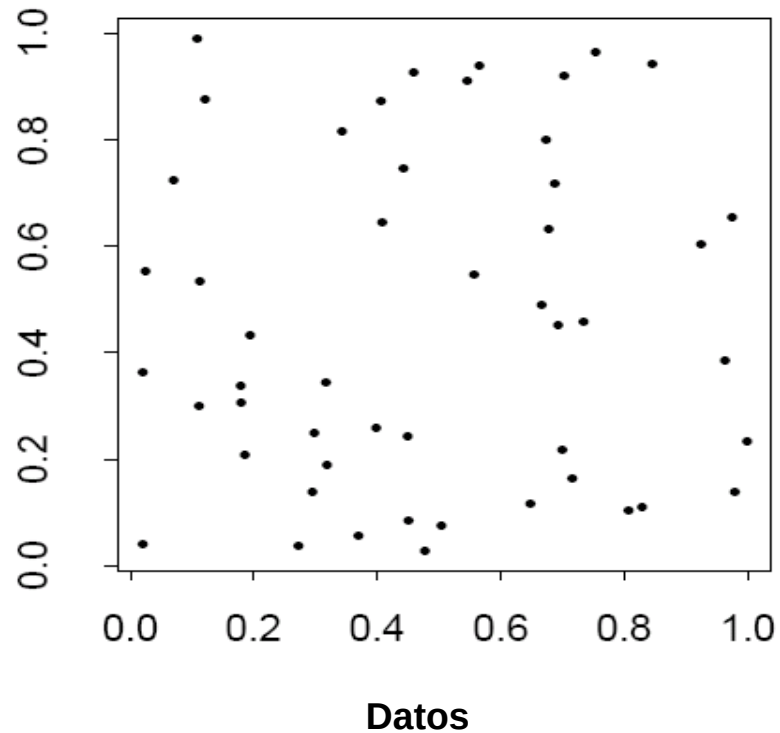
# Ejemplo: 2 clusters



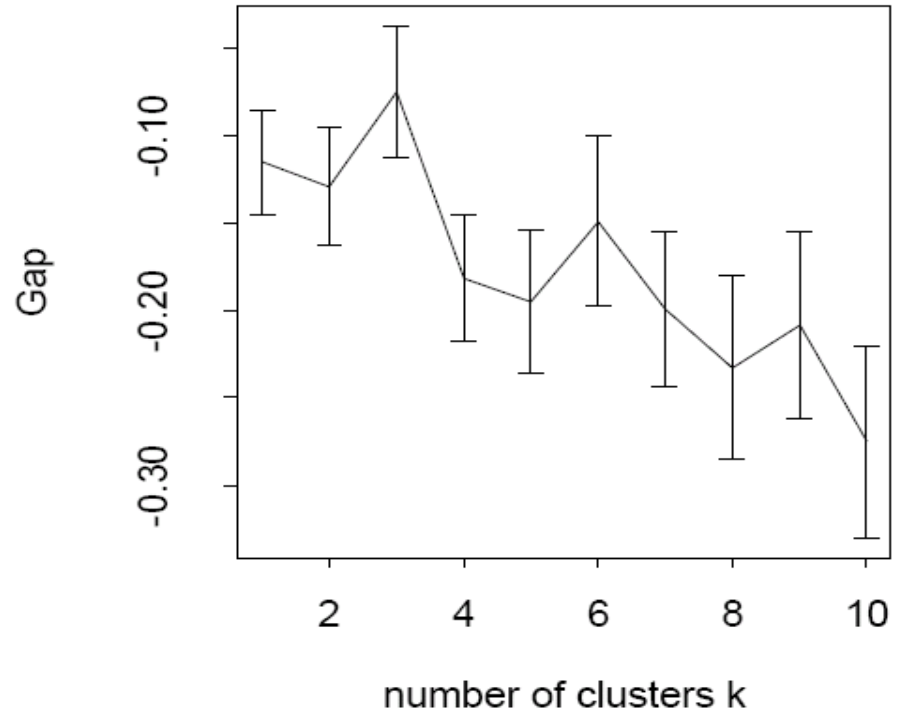
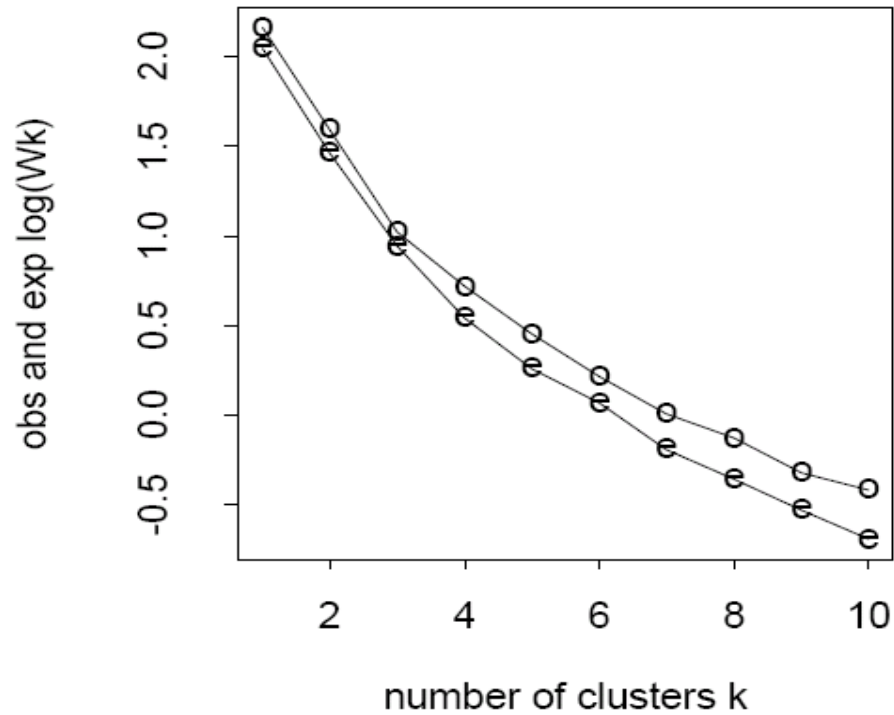
# Ejemplo: 2 clusters



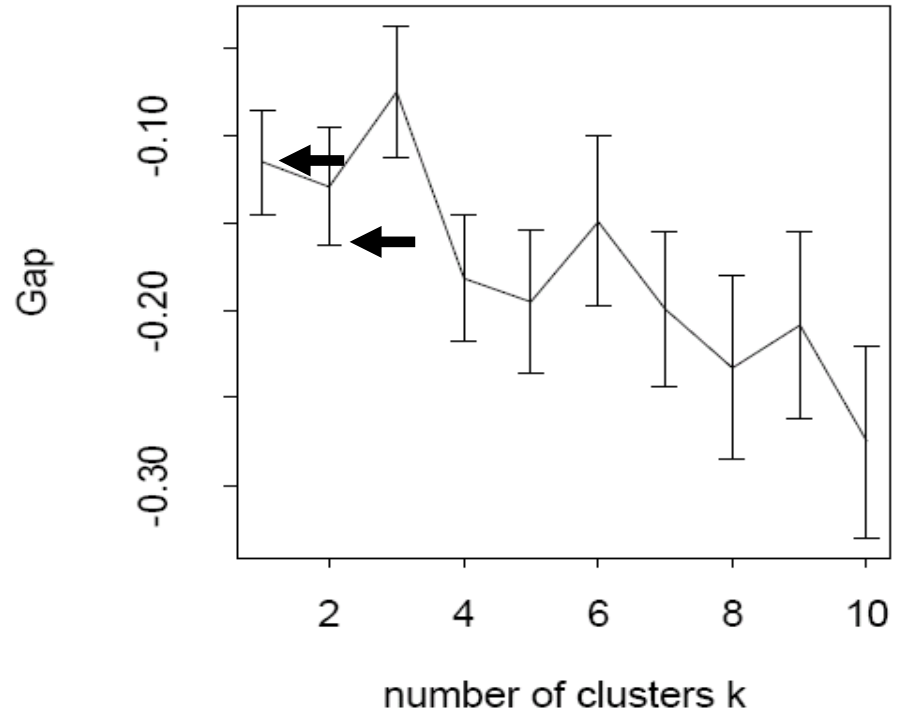
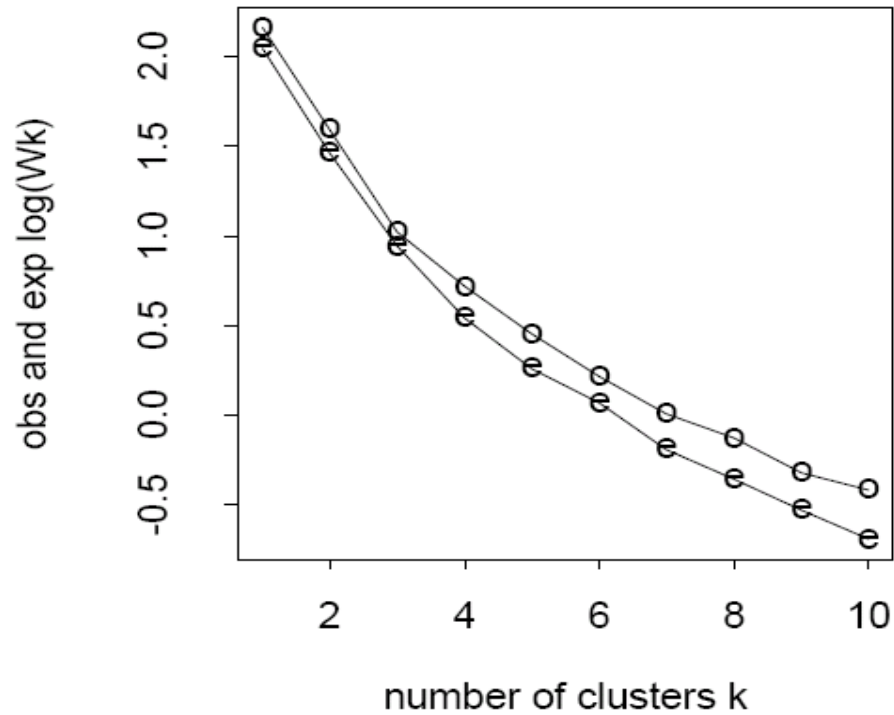
# Ejemplo: sin clusters



# Ejemplo: sin clusters



# Ejemplo: sin clusters





# Gap: comparación



Method	Estimate of number of clusters $\hat{k}$									
	1	2	3	4	5	6	7	8	9	10
<i>Null model in 10 dimensions</i>										
CH	0*	50	0	0	0	0	0	0	0	0
KL	0*	29	5	3	3	2	2	0	0	0
Hartigan	0*	0	1	20	21	6	0	0	0	0
Silhouette	0*	49	1	0	0	0	0	0	0	0
Gap/unif	49*	1	0	0	0	0	0	0	0	0
Gap/pc	50*	0	0	0	0	0	0	0	0	0
<i>Three cluster model</i>										
CH	0	0	50*	0	0	0	0	0	0	0
KL	0	0	39*	0	5	1	1	2	0	0
Hartigan	0	0	1*	8	19	13	3	3	2	1
Silhouette	0	0	50*	0	0	0	0	0	0	0
Gap/unif	1	0	49*	0	0	0	0	0	0	0
Gap/pc	2	0	48*	0	0	0	0	0	0	0

# Gap: comparación (cont.)

	<i>Random 4 cluster model in 10 dims.</i>									
CH	0	1	4	44*	1	0	0	0	0	0
KL	0	0	0	45*	3	1	1	0	0	0
Hartigan	0	0	2	48*	0	0	0	0	0	0
Silhouette	0	13	20	16*	5	0	0	0	0	0
Gap/unif	0	0	0	50*	1	0	0	0	0	0
Gap/pc	0	0	4	46*	0	0	0	0	0	0
	<i>Two elongated clusters</i>									
CH	0	0*	0	0	0	0	0	7	16	27
KL	0	50*	0	0	0	0	0	0	0	0
Hartigan	0	0*	0	1	0	2	1	5	6	35
Gap/unif	0	0*	17	16	2	14	1	0	0	0
Gap/pc	0	50*	0	0	0	0	0	0	0	0



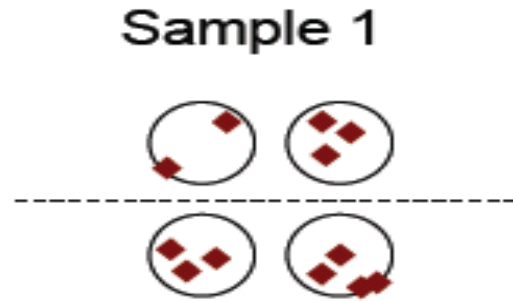
# Estabilidad - Introducción

- Idea base: Los resultados científicos tienen que ser reproducibles. Los “clusters naturales” de un problema se tienen que encontrar siempre.
- Si cambio un punto en un problema y la solución desaparece, entonces no son “clusters naturales” sino un resultado ficticio creado por mi algoritmo.

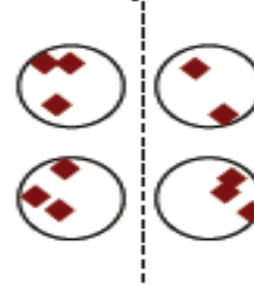
# Ejemplo a favor



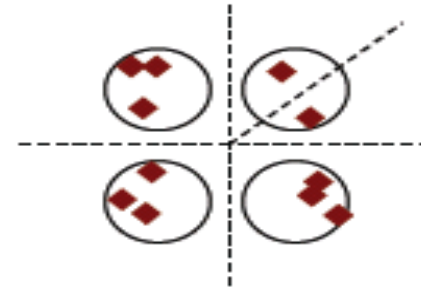
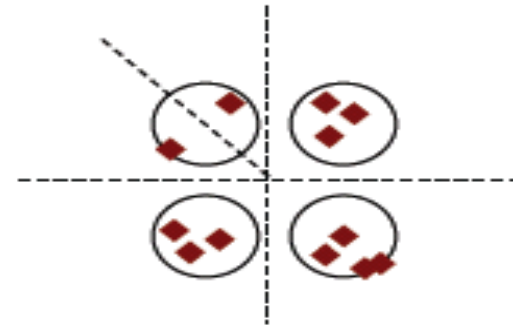
$k = 2$ :



Sample 2

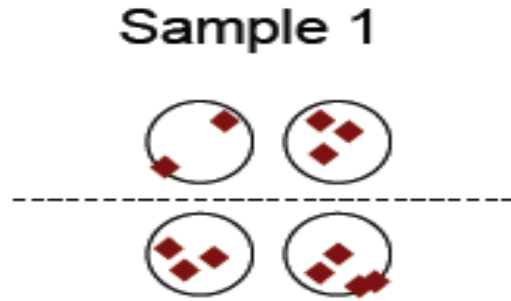


$k = 5$ :

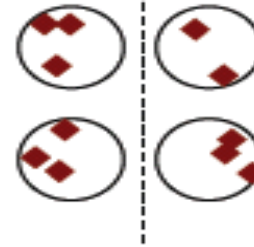


# Ejemplo a favor

$k = 2$ :

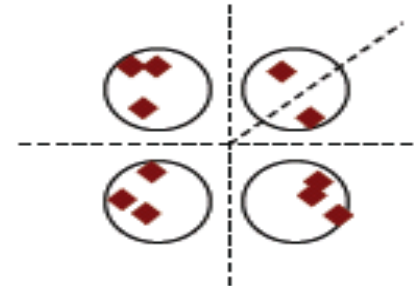
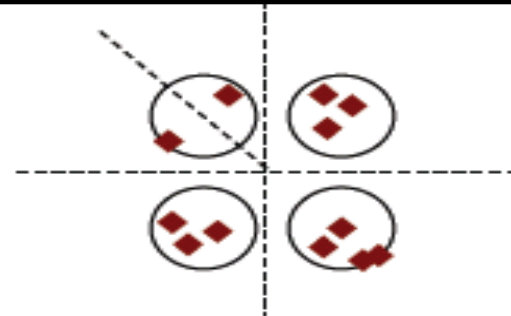


Sample 2



La división en clusters naturales suele ser la más estable

$k = 5$ :





# Estabilidad – Algoritmo base

- Variar la cantidad de clusters
- Construir muchas soluciones replicadas
- Evaluar la estabilidad de las soluciones
- Seleccionar el k con más estabilidad



# Soluciones replicadas

- Estabilidad: Problemas similares dan soluciones similares.
- Como muchos algoritmos son deterministas, se necesita generar problemas perturbados
- Hay un trade-off al perturbar:
  - Si cambio mucho puedo destruir la estructura natural
  - Si cambio poco puede parecer estable algo que no es

# Cómo generar réplicas?







# Cómo generar réplicas?

- Subsample
  - Tomo una muestra al azar de los datos originales
  - Se suele tomar del 70% al 90% de los datos
- Agregar ruido
  - Agregar ruido blanco (normal) a los datos originales
  - Bajo porcentaje de la señal (menos de 10%)
- Proyecciones
  - En datos de alta dimensionalidad, tomar proyecciones sobre un subespacio elegido al azar

# Soluciones



- Una vez que tengo los datos perturbados, tengo que clusterizarlos.
- Se usa siempre el mismo algoritmo, buscando que la única variación sean los datos
- Próximo paso: Medir cuán diferentes son las soluciones



# Evaluación: mismos datos

- Siempre se evalúan pares de soluciones
- Cuando los conjuntos de datos son iguales:
  - Cuento cuántos pares de puntos que están en un mismo cluster en la primer solución también lo están en la segunda
  - Normalizo la cuenta
  - Distintas normalizaciones dan distintos índices:
    - Jaccard
    - Rand
    - Fowlkes and Mallows

Muy similares



# Evaluación: distintos datos

- Cuando los conjuntos de datos son distintos:
  - Restringir: Evaluar la intersección
    - La más usada
    - Tiene sentido si la intersección es grande
    - Se pierde información
  - Extender: Agregar los puntos faltantes a cada solución (obtenida previamente)
    - Por ejemplo, en k-means, asignar al centro cercano
    - En single linkage asigno al primer vecino espacial
    - Menos común



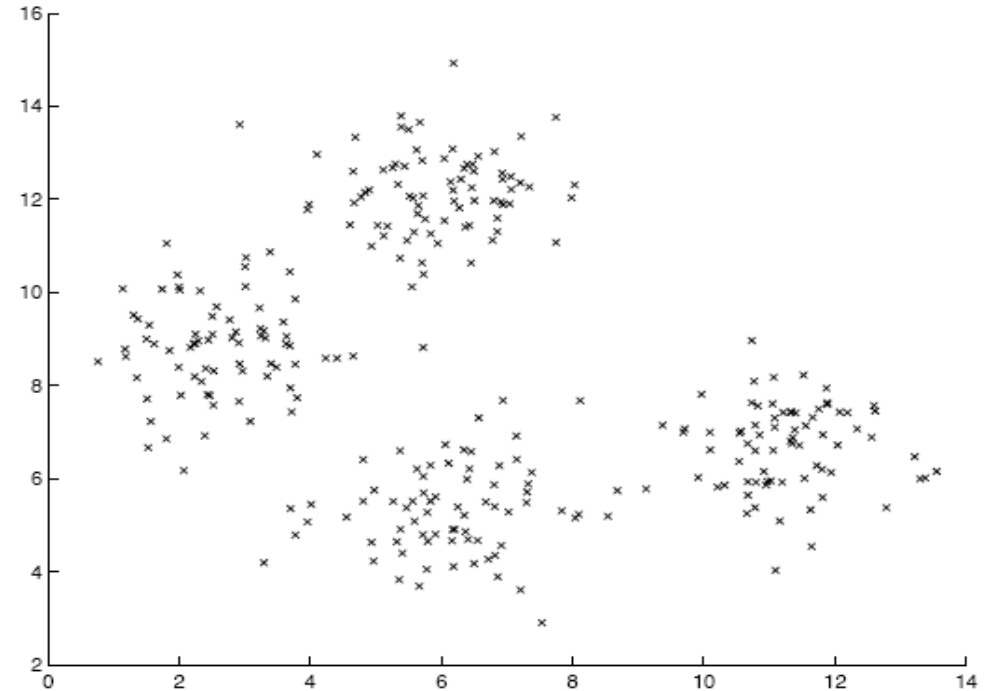
# Seleccionar el k

- Tengo una muestra de valores de similaridad entre las soluciones, para distintos k
- Evaluación más simple: tomo la (mayor) media (no es recomendable).
- Busco algo cualitativo, que muestre cuando una distribución está mucho más a la derecha
- Algunos autores miran la concentración del histograma (o el área bajo la cumulativa)

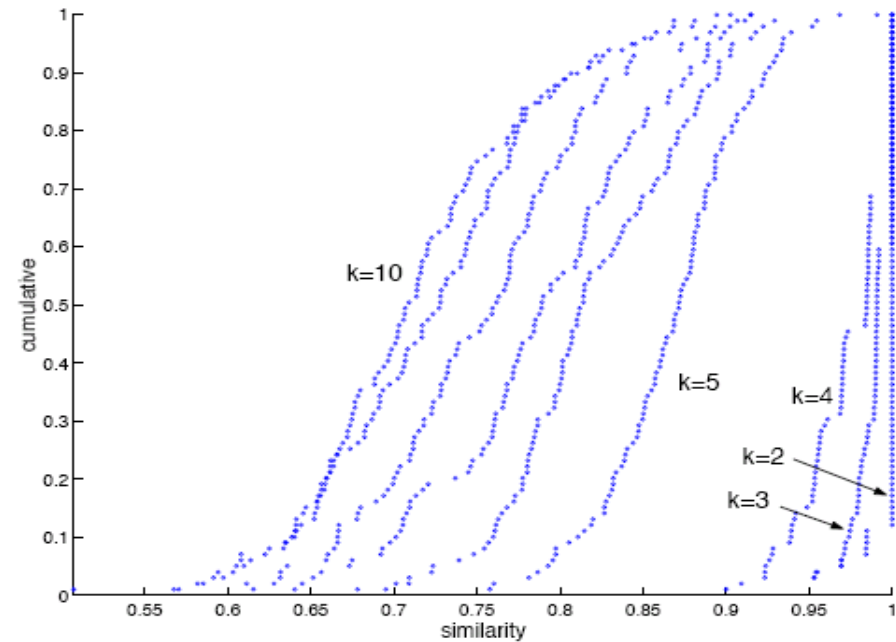
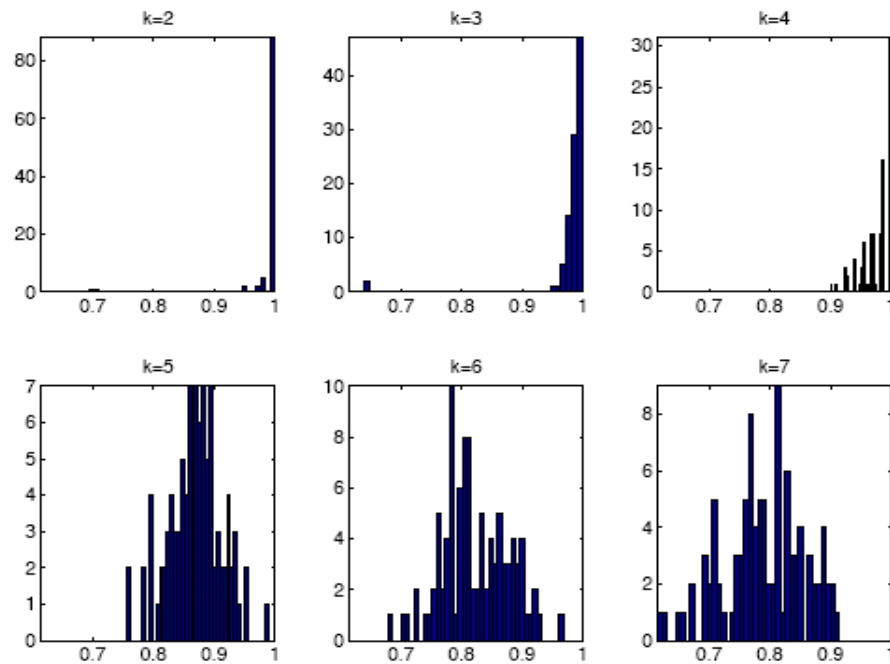
# Seleccionar el k (2)



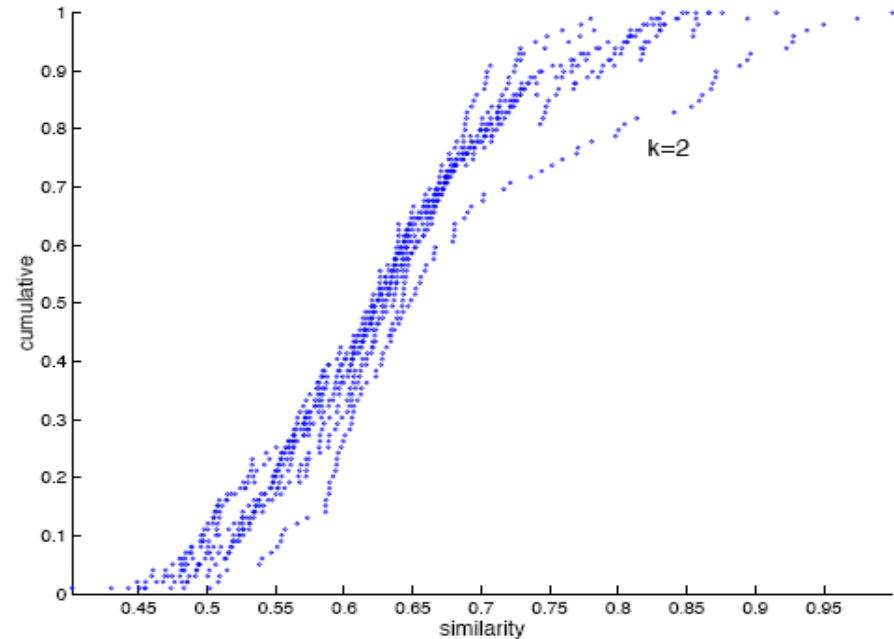
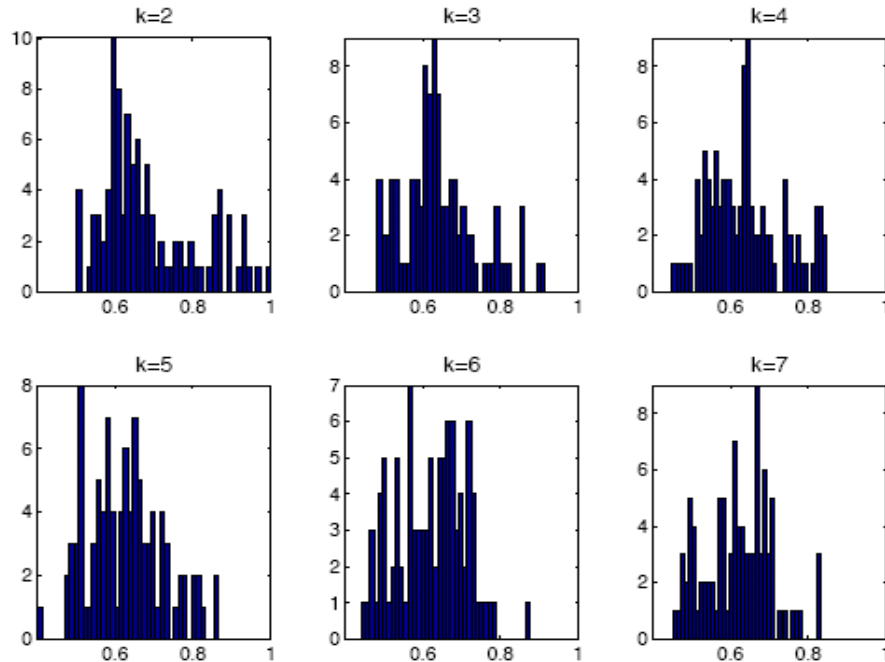
- Ejemplo:
  - 4 clusters gaussianos
  - 250 puntos en cada uno



# Seleccionar el k: 4 gaussianas



# Seleccionar el k: distribución uniforme



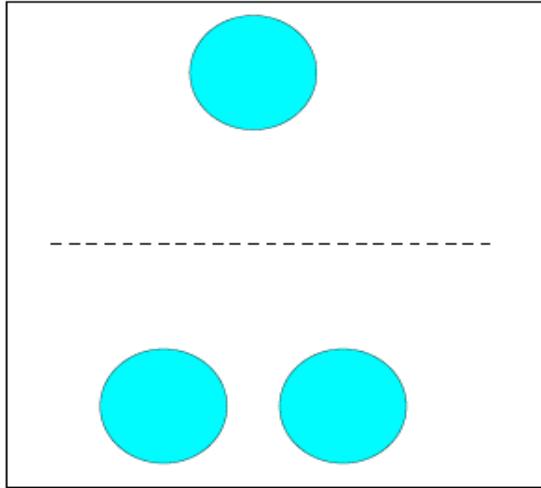




# Contraejemplos

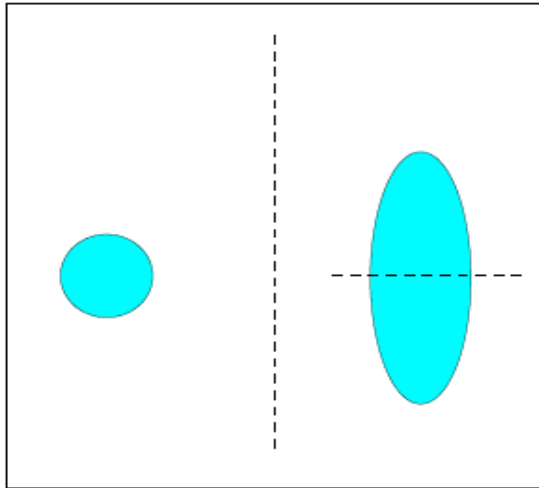
- Uno asume que las soluciones naturales tienen que ser estables
- Lo contrario no está garantizado. Hay soluciones estables que son “artificiales”

# Contraejemplos



- Distribución asimétrica
- $K=2$  es un poco mejor que  $k=3$ .
- Se suele superar seleccionando, entre todas las estables, la que tiene el mayor  $k$

# Contraejemplos



- Un cluster asimétrico
- $K=2$  es un poco mejor que  $k=3$ .
- Pero va en contra del caso anterior
- En este caso se pierde casi siempre

# Contraejemplos



- Los algoritmos que buscan “bolas” son estables en distribuciones alargadas para  $k$  crecientes
  - K-means en una distribución uniforme unidimensional



# Resumen

- Encontrar el número de clusters: problema abierto de interés actual
- Dos heurísticas muy usadas
  - Gap: Comparar la bondad de la solución con una distribución nula
  - Estabilidad: Las soluciones naturales tienen que ser estables
  - Las dos detectan cuando no hay clusters