



Les 14 – DEG's en clustering (3)

Emile Apol

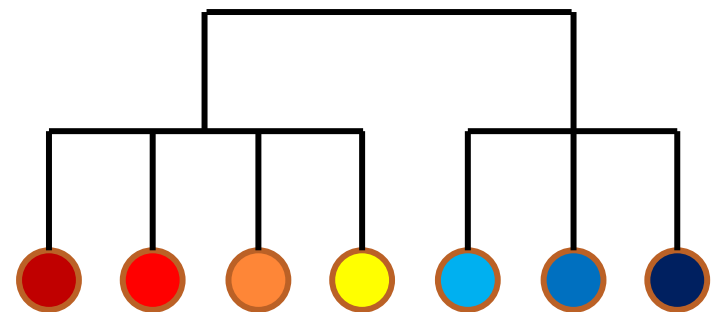
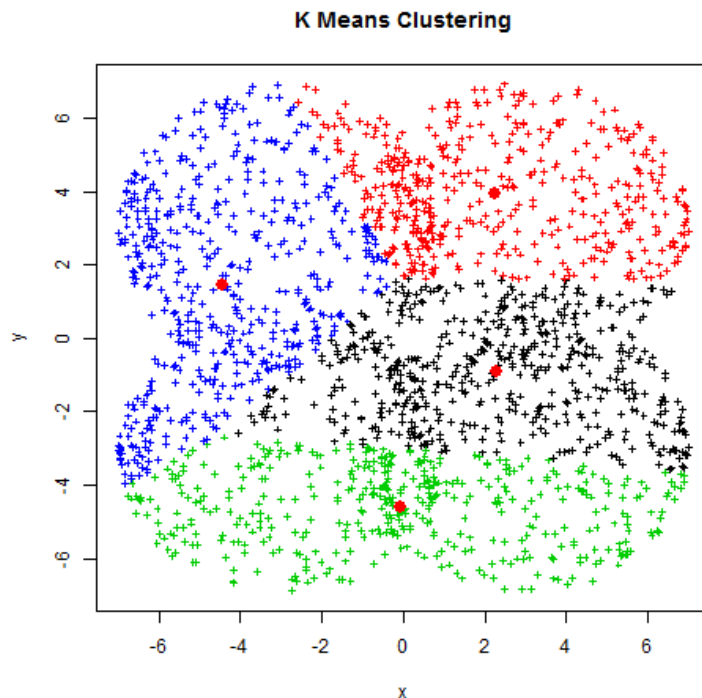


Hanze University Groningen
APPLIED SCIENCES

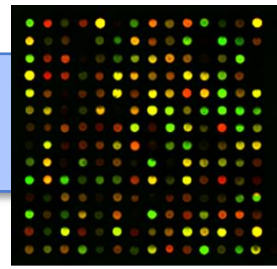
Institute for
Life Science & Technology

LES 12

- Verdeling van vals-positieven in MA data analyse
- Koppelen van MA data aan “externe” database
- *k*-means clusteren: **kmeans**



MICROARRAY ANALYSE: STAPPENPLAN



- Background correctie
- Log transformatie
- Normalisatie (bijv. loess)
- Toetsen op DEG's:
 - t -toets, 1-way ANOVA, ...
 - Wilcoxon's toets, Kruskal-Wallis toets, ...
- Aanpassen p -waarden voor multiple toetsing
- Clustering van DEG's:
 - Hiërarchisch clusteren
 - k -means
 - Principale Componenten Analyse (PCA)
- Grafische weergave: heatmap, vulcano plot, ...

MULTIPLE TOETSEN

- Per toets een kans α op vals positieve (FP) uitslag, d.w.z. H_0 waar maar toch verwerpen
- Bij G (= aantal genen) toetsen achter elkaar dus een verwacht (= gemiddeld) aantal vals positieven:

$$\overline{FP} = \alpha \cdot G$$

- Voorbeeld: 10 000 genen met $\alpha = 0.05$ geeft

$$\overline{FP} = 0.05 \cdot 10\,000 = 500$$

- De verdeling van $y = FP$ is ongeveer Poisson verdeeld, met

$$p(y) = \frac{\lambda^y \cdot e^{-\lambda}}{y!}$$

$$\lambda = \overline{FP} = \bar{y}$$

POISSON VERDELING (1)

- Kansverdeling: $p(y) = \frac{\lambda^y \cdot e^{-\lambda}}{y!}$
- R syntax:
 - Kansverdeling: `dpois(x, lambda)`

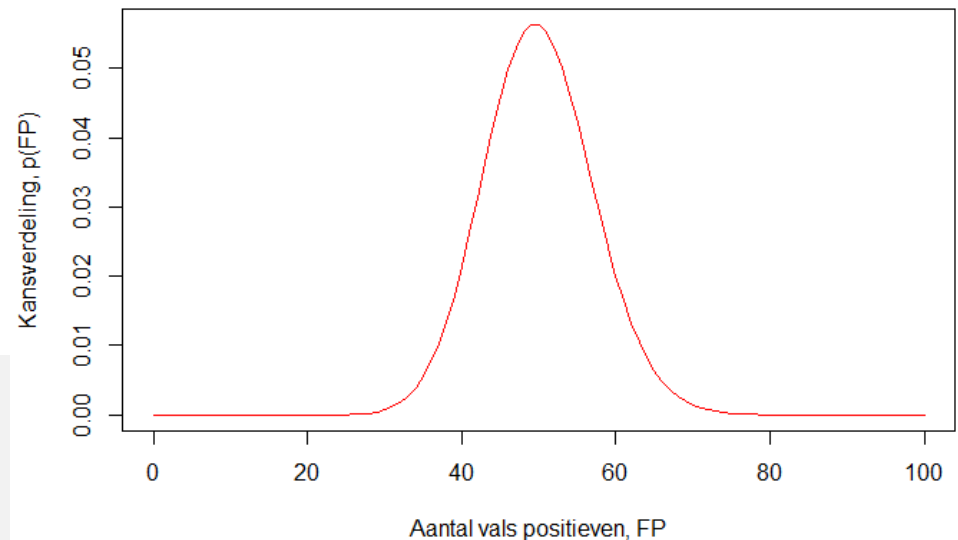
Stel: MA analyse met $G = 1000$ genen
en significantieniveau $\alpha = 0.05$:
Gemiddeld aantal FP = $\lambda = \alpha \cdot G = 50$

```
alpha <- 0.05
G <- 1000

n.FP.av <- G*alpha
cat("\nGemiddeld aantal FP genen: ", n.FP.av, "\n")

# Parameter van Poisson verdeling:
lambda <- n.FP.av
```

```
curve(dpois(x, lambda = lambda), from = 0, to = 2*lambda, n = 2*lambda+1, add = F, type = "l", col = "red",
      xlab = "Aantal vals positieven, FP", ylab = "Kansverdeling, p(FP)")
```



POISSON VERDELING (2)

- Kansverdeling: $p(y) = \frac{\lambda^y \cdot e^{-\lambda}}{y!}$
- R syntax:
 - Quantile functie: `qpois(p, lambda)`

Stel: MA analyse met $G = 1000$ genen en significantieniveau $\alpha = 0.05$:

Gemiddeld aantal FP = $\lambda = \alpha \cdot G = 50$.

Wat is het 95% BI van het gemiddeld aantal FP genen?

```
##{r}
BI <- qpois(c(0.025, 0.975), lambda = lambda)
cat("Het 95% BI voor het aantal FP genen bij ",G," onderzochte genen is: [",BI,"]\n")
```

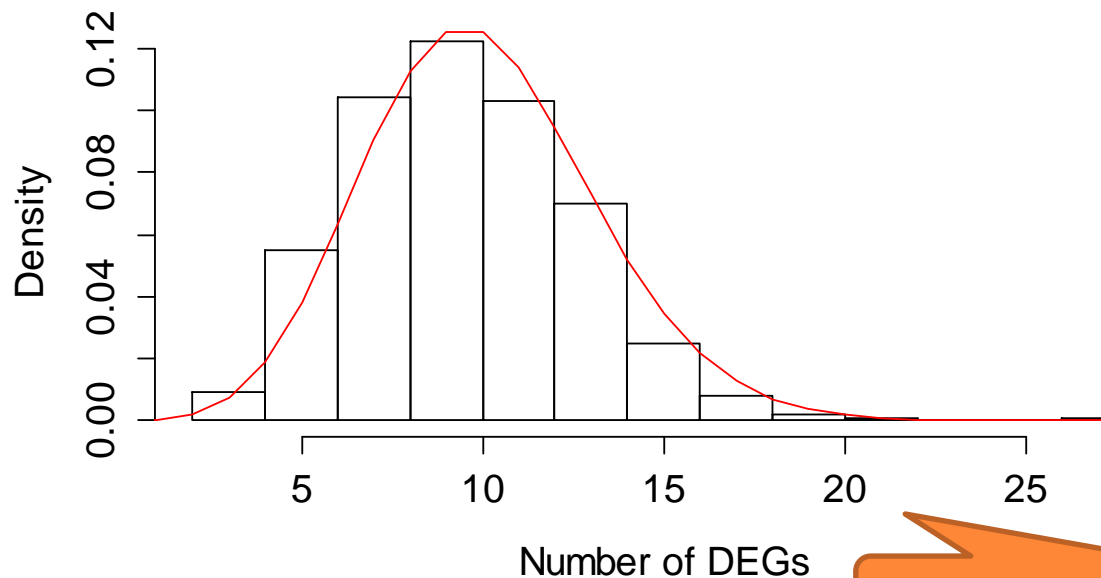
Het 95% BI voor het aantal FP genen bij 1000 onderzochte genen is: [37 64]

Dus met 95% zekerheid ligt het aantal vals positieve genen op basis van niet-gecorrigeerde p-waarden tussen de 37 en 64.

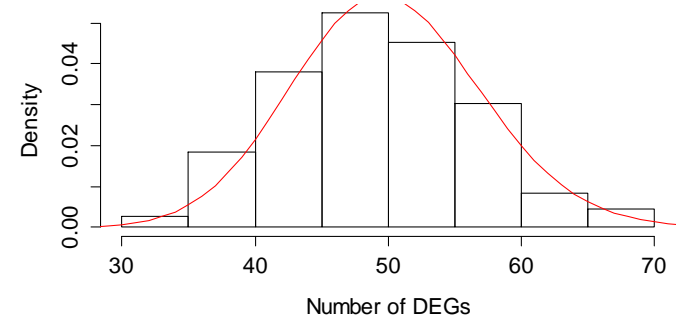
MULTIPLE TOETSEN

- Simulaties in R ($G = 200$, 1 000 of 10 000 genen, 500 x random MA met 5 replica's), met theoretische Poisson verdeling (in rood):

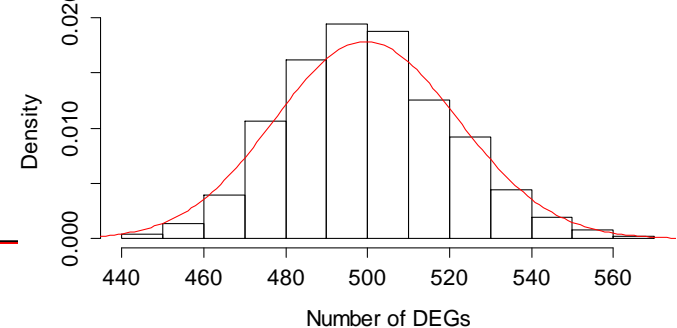
500 times 200 genes, alpha = 0.05



500 times 1000 genes, alpha = 0.05



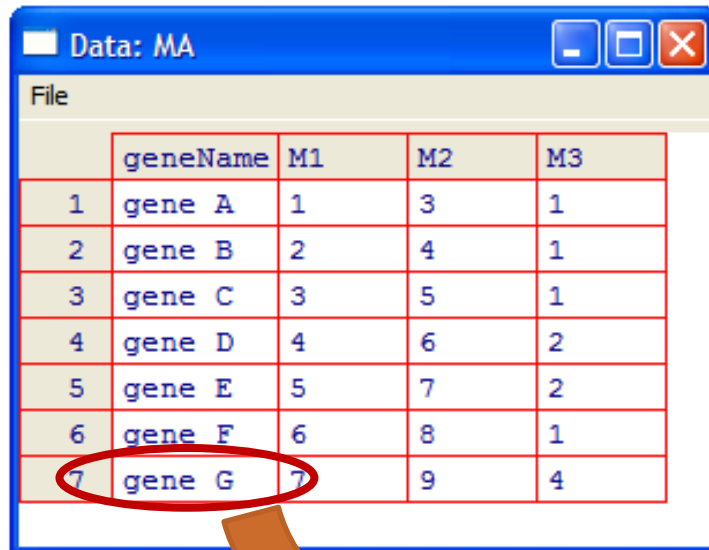
500 times 10000 genes, alpha = 0.05



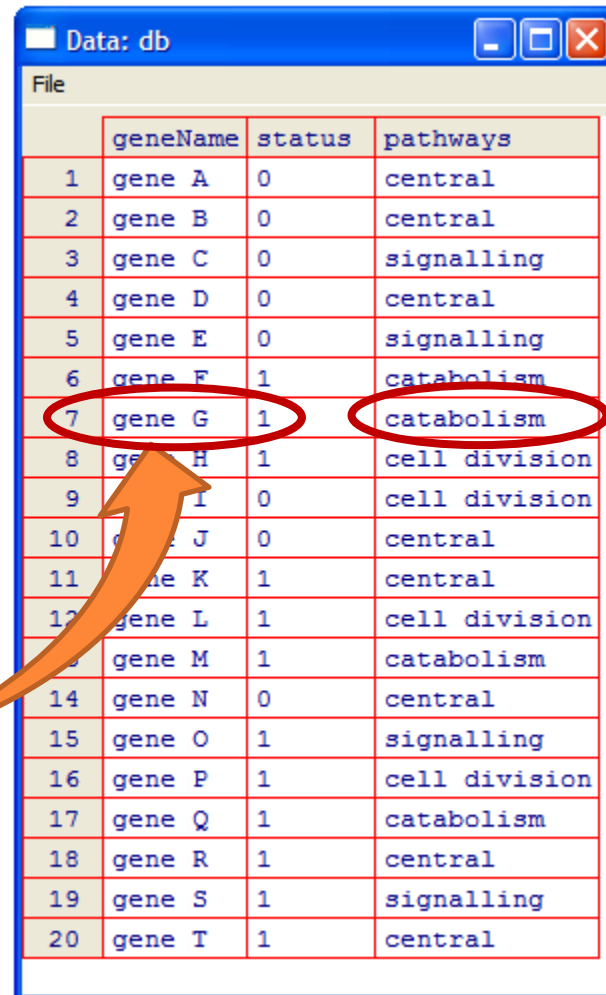
Dit zijn allemaal vals positieven!

KOPPELEN AAN “EXTERNE” DATABASE IN R

- Stel, we hebben dataframe **MA** met microarray data, en een dataframe **db** met extra info over genen:



	geneName	M1	M2	M3
1	gene A	1	3	1
2	gene B	2	4	1
3	gene C	3	5	1
4	gene D	4	6	2
5	gene E	5	7	2
6	gene F	6	8	1
7	gene G	7	9	4



	geneName	status	pathways
1	gene A	0	central
2	gene B	0	central
3	gene C	0	signalling
4	gene D	0	central
5	gene E	0	signalling
6	gene F	1	catabolism
7	gene G	1	catabolism
8	gene H	1	cell division
9	gene I	0	cell division
10	gene J	0	central
11	gene K	1	central
12	gene L	1	cell division
13	gene M	1	catabolism
14	gene N	0	central
15	gene O	1	signalling
16	gene P	1	cell division
17	gene Q	1	catabolism
18	gene R	1	central
19	gene S	1	signalling
20	gene T	1	central

KOPPELEN AAN “EXTERNE” DATABASE IN R

- Matchen van gen namen (of ORF, ...)
- Stringfunctie **%in%** geeft alleen **T** of **F**:
 - **MA\$geneName[7] %in% db\$geneName**
TRUE
- Stringfunctie **match** geeft positie van 1^e match:
- **match(MA\$geneName[7], db\$geneName)**
7 regel 7 in db
- **\$geneName** is een factor, dus resultaat is een level, GEEN string:

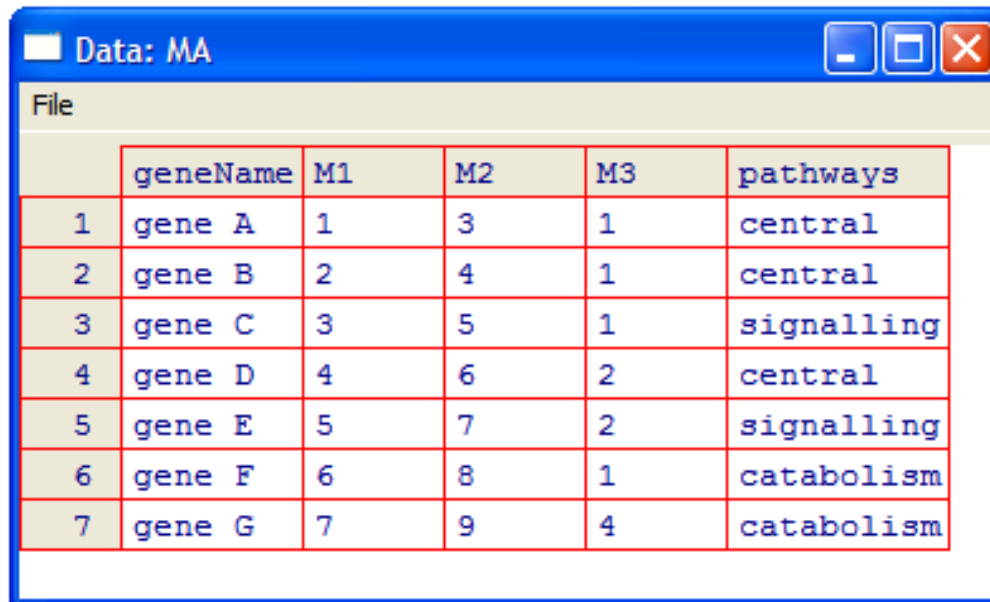
```
> db$pathways[match(MA$geneName[7], db$geneName)] # level, geen string!  
[1] catabolism  
Levels: catabolism cell division central signalling  
> as.character(db$pathways[match(MA$geneName[7], db$geneName)]) # string!  
[1] "catabolism"
```

KOPPELEN AAN “EXTERNE” DATABASE IN R

- Loop over alle genen (=rijen) in dataframe **MA**:

```
myPathways <- c()
for(i in 1 : nrow(MA)){
  myPathways[i] <- as.character(db$pathways[match(MA$geneName[i], db$geneName)])
}
MA <- data.frame(MA, pathways=myPathways)
```

- Resultaat:



	geneName	M1	M2	M3	pathways
1	gene A	1	3	1	central
2	gene B	2	4	1	central
3	gene C	3	5	1	signalling
4	gene D	4	6	2	central
5	gene E	5	7	2	signalling
6	gene F	6	8	1	catabolism
7	gene G	7	9	4	catabolism

KOPPELEN AAN “EXTERNE” DATABASE IN R

- Meestal blijken for-loops niet nodig in R:

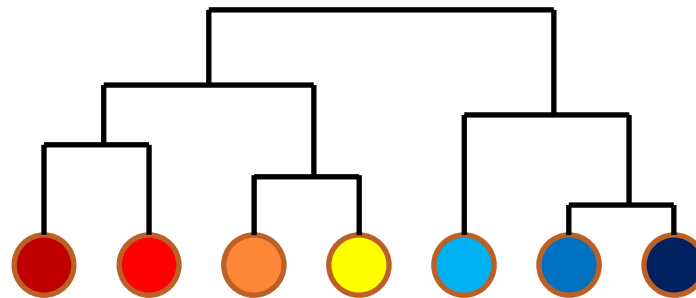
```
myPathways <- c()
for(i in 1 : nrow(MA)){
  myPathways[i] <- as.character(db$pathways[match(MA$geneName[i], db$geneName)])
}
MA <- data.frame(MA, pathways=myPathways)
```

- Dit kan ook in 1 regel, omdat de **match(x, y)** functie ook met vectoren **x** werkt (resultaat = vector met posities in **y**):

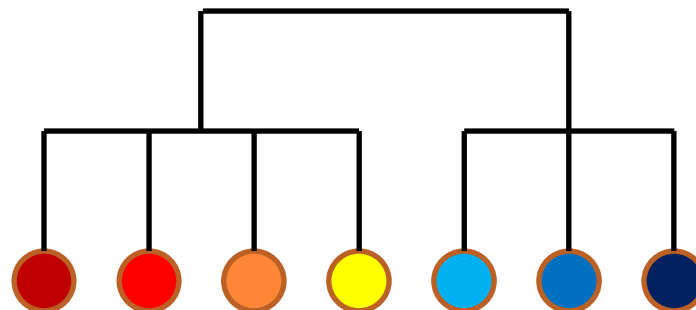
```
myPathways <- as.character(db$pathways[match(MA$geneName, db$geneName)])
MA <- data.frame(MA, pathways=myPathways)
```

K-MEANS CLUSTERING

- Hiërarchisch clusteren **hclust**: alles wordt één supercluster, evt. zelf in k subclusters opdelen **cutree**



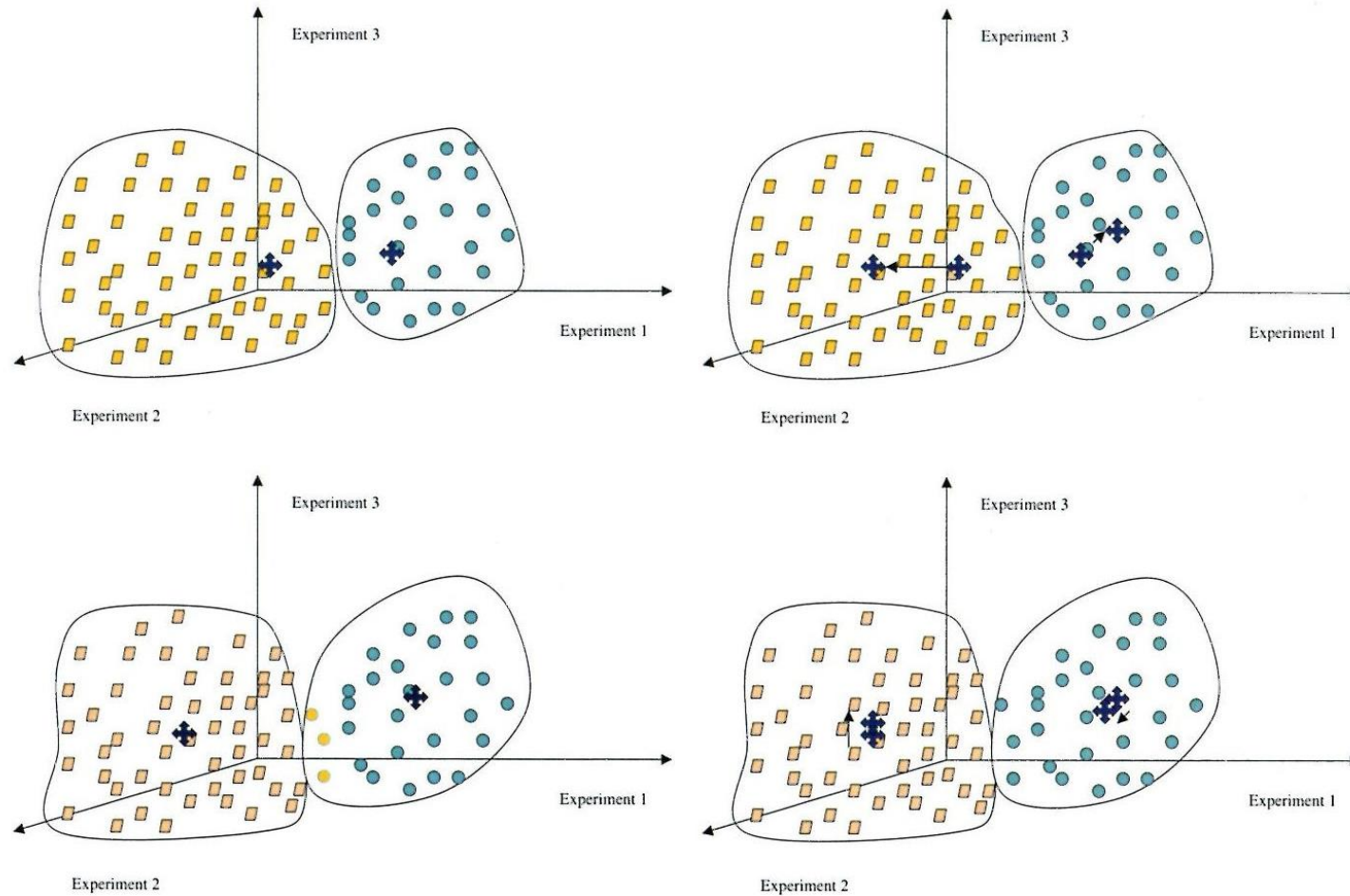
- k -means clustering **kmeans**: aantal clusters (k) van te voren bekend



K-MEANS CLUSTERING

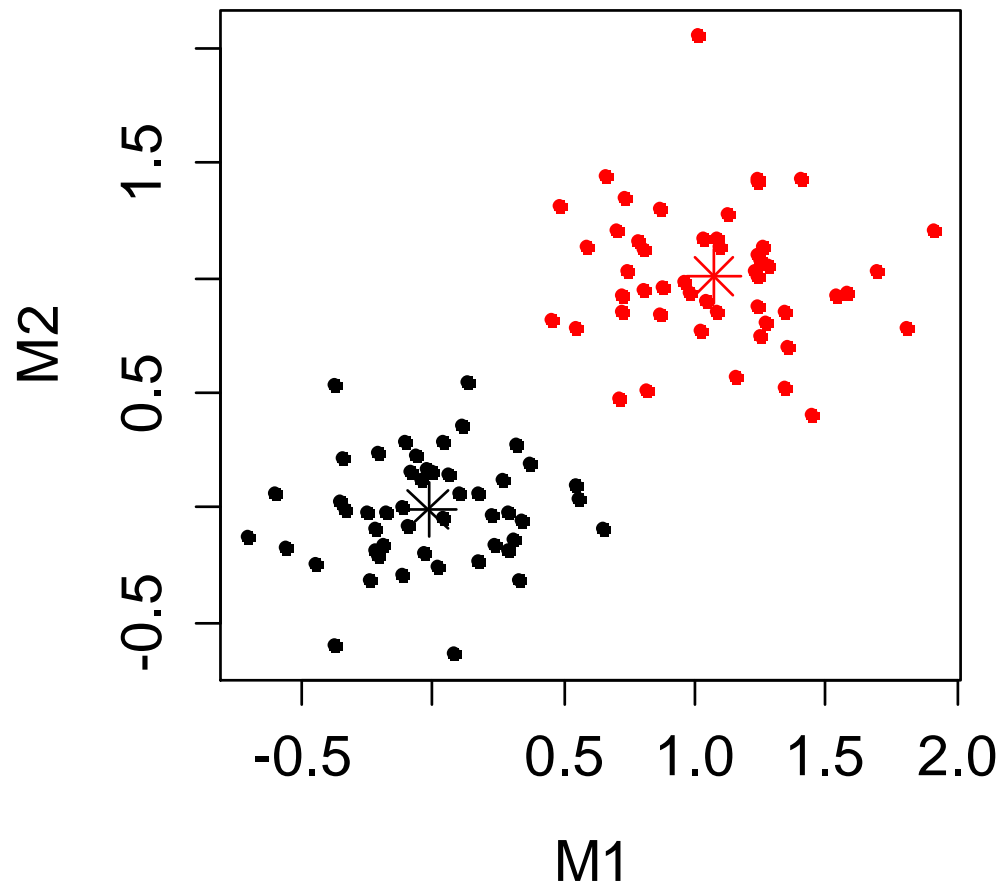
- **Distance**: Euclidisch (per definitie, anders bijv. `pam()`)
- Algoritme:
 1. Stel zelf aantal clusters k vast
 2. Bepaal **random** k cluster centers
 3. Bepaal voor elk punt op basis van distance tot center tot welke cluster hij behoort (least squares)
 4. Bereken nieuw cluster center (least squares)
 5. Bepaal opnieuw voor elk punt op basis van distance tot center tot welke cluster hij behoortenz.

K-MEANS CLUSTERING



K-MEANS CLUSTERING

- Voorbeeld: matrix **X** met 100 genen in 2 samples
 - **kmeans (X, centers=2)**
- Resultaat:



KMEANS()

- *k*-means clustering via **kmeans ()**:
 - **kclust <- kmeans(X, centers=2)**
- Resultaat: list

kclust\$ element	inhoud	kclust\$ element	inhoud
\$cluster	per gen cluster nr	\$totss	SS_{tot}
\$centers	coördinaten centers	\$withinss	SS_{within}
\$size	aantal genen per cluster	\$tot.withinss	sum SS_{within}
		\$betweeness	SS_{between}

```
> kclust$cluster
gene1 gene2 gene3 gene4 gene5 gene6
    2     2     1     1     2     2
```

cluster nummer

K-MEANS CLUSTERING: WITHIN & BETWEEN SS

- Opsplitsen van totale variantie (t.o.v. overall center) in variantie binnen clusters (t.o.v. cluster centers) en variantie tussen clusters:

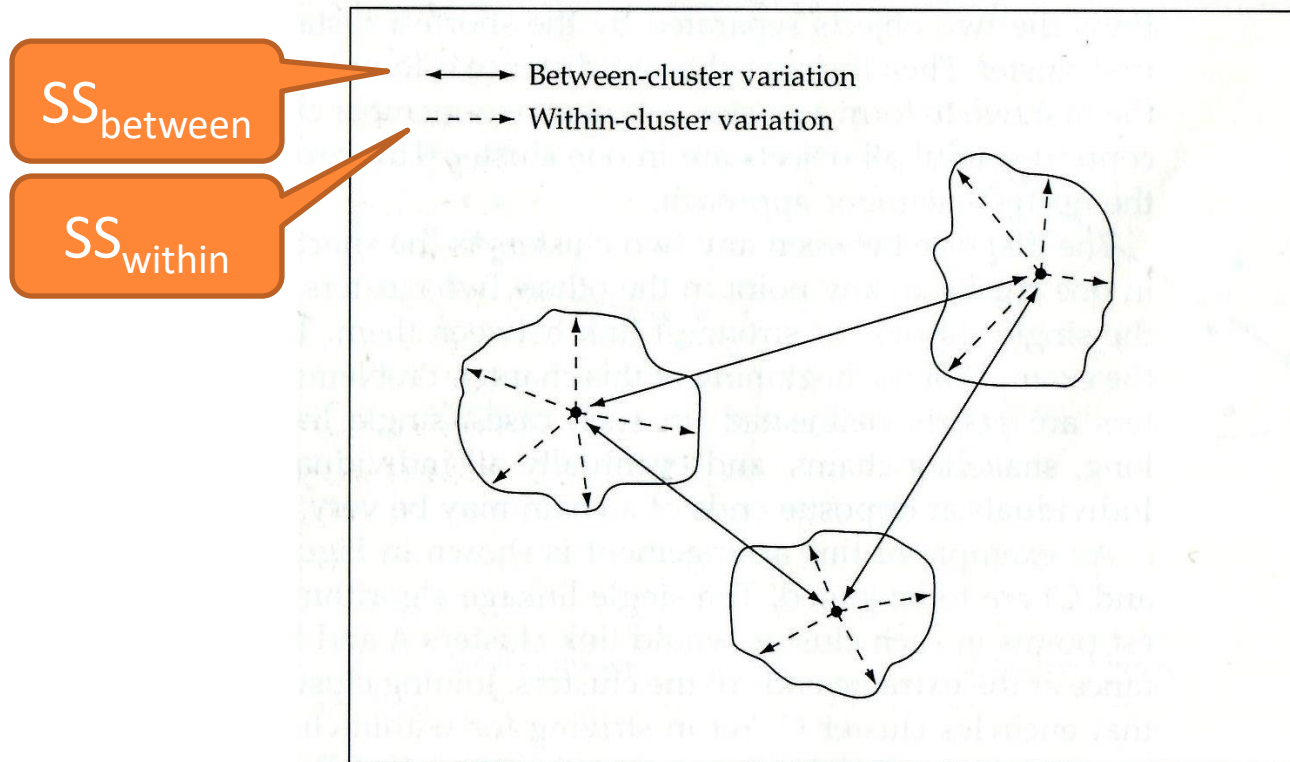
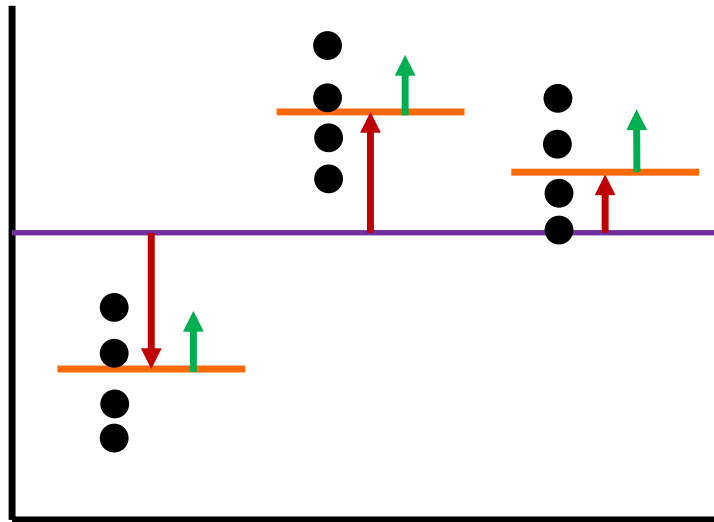


FIGURE 9.7 Cluster Diagram Showing Between- and Within-Cluster Variation

K-MEANS CLUSTERING: WITHIN & BETWEEN SS

1-way ANOVA



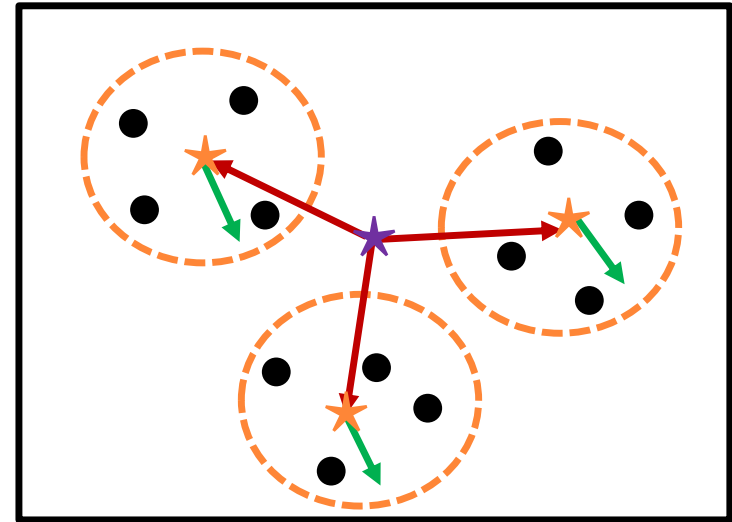
Factor A →

$$SS_{\text{tot}} = SS_A + SS_{\text{err}}$$

$$\text{Effectsterkte: } \eta^2 = \frac{SS_A}{SS_{\text{tot}}}$$

k-means clustering

Sample $M_2 \rightarrow$



Sample $M_1 \rightarrow$

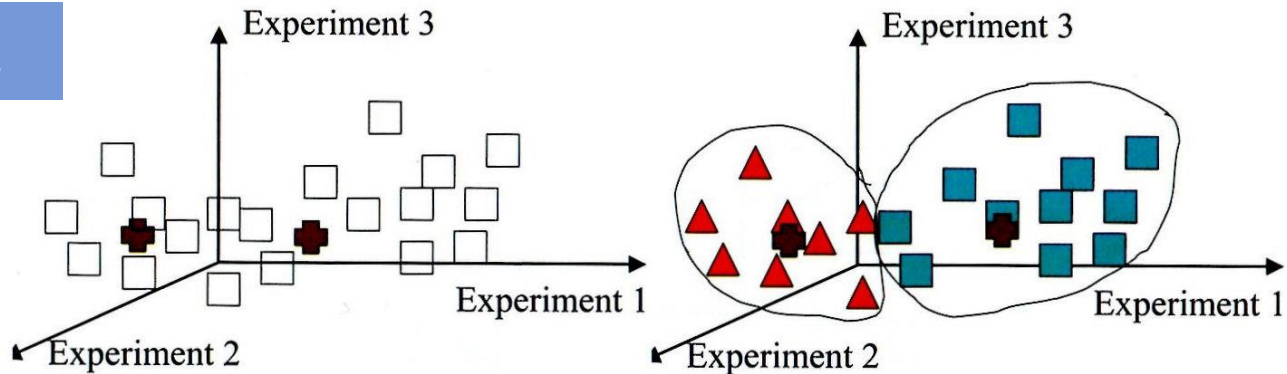
$$SS_{\text{tot}} = SS_{\text{clust}} + SS_{\text{within}}$$

$$\text{Effectsterkte: } \eta^2 = \frac{SS_{\text{clust}}}{SS_{\text{tot}}}$$

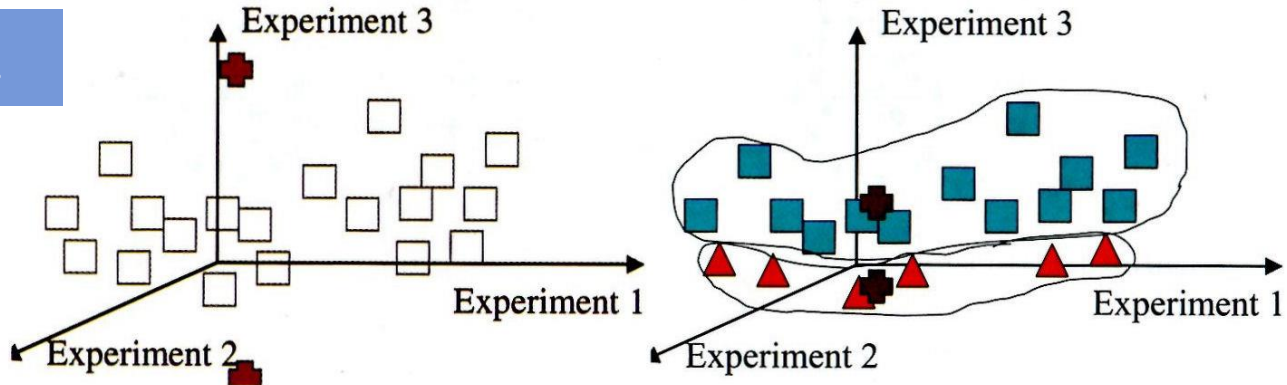
K-MEANS CLUSTERING: NIET DETERMINISTISCH

- Het resultaat van k -means clustering hangt af van de random keuze van begin centers!

run 1



run 2



KMEANS(): NSTART

- Bij de functie **kmeans** ook een optie voor meerdere random begin centers: **nstart**
 - **kmeans(X, centers=2, nstart=5)**
- Resultaat is oplossing met kleinste SS_{within}

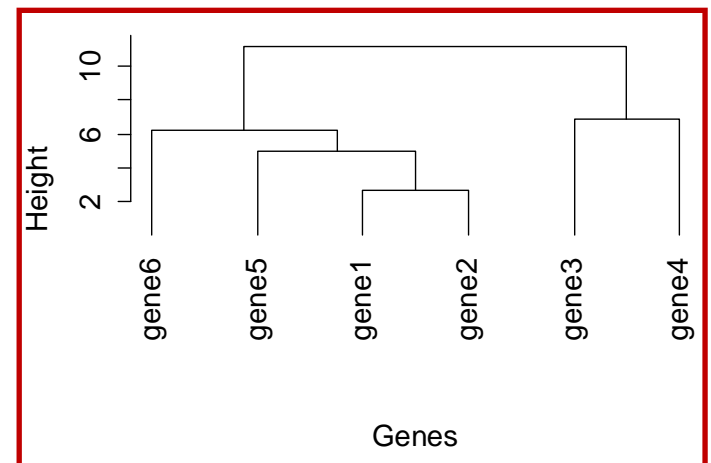
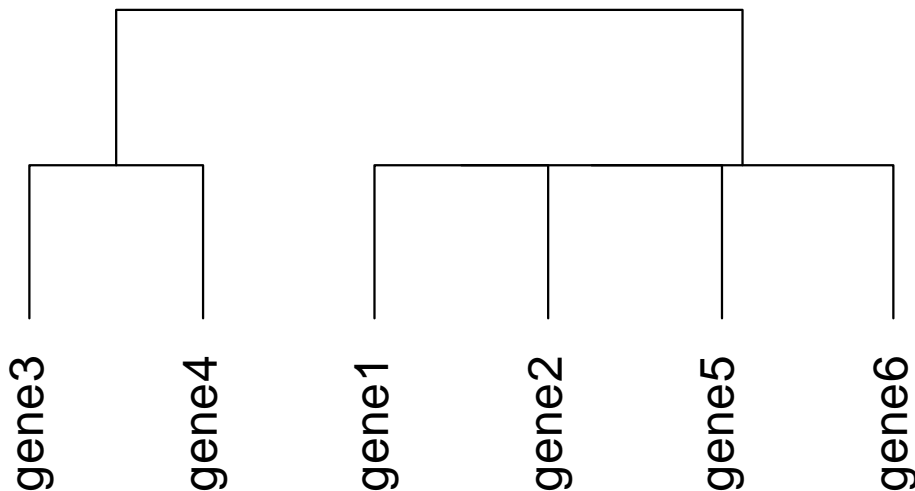
K-MEANS CLUSTERING: WEERGAVE RESULTAAT

- **hclust**: o.a. dendrogram
- **kmeans**: officieel géén dendrogram, alleen lijst met cluster nummers per gen
- Eigen functie **makeKMeansDendrogram** (zie ook BB):

```
makeKMeansDendrogram <- function(kc1){  
  clusters <- kc1$cluster  
  n.genes <- length(clusters)  
  # so there will be n.genes-1 hierarchical clusters!  
  gene.names <- names(clusters)  
  n.clusters <- length(kc1$size)  
  cat("kc1$size = ",kc1$size,"\n")  
  n.clusters.NONSINGLE <- length(which(kc1$size > 1))  
  n.clusters.SINGLE <- length(which(kc1$size == 1))  
  cat("Nr of non-singleton clusters = ",n.clusters.NONSINGLE,"\n")  
  cat("Nr of      singleton clusters = ",n.clusters.SINGLE,"\n")  
  NONSINGLE.g <- which(kc1$size > 1) # which cluster nrs are non-singleton?  
  cat("Non-singleton cluster nrs are ",NONSINGLE.g,"\n")  
  size.clusters <- kc1$size  
  ORDER <- order(clusters)
```

KMEANS(): “DENDROGRAM”

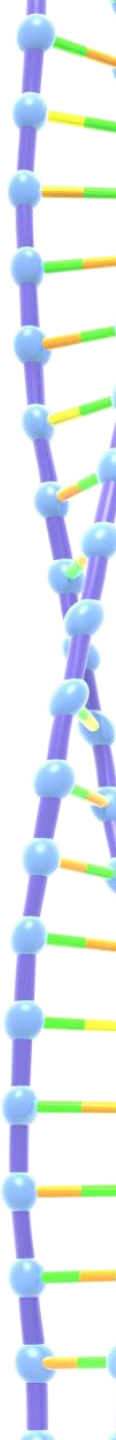
- Dataframe/matrix **M** met G rijen (genes) en n samples
 - `kcl <- kmeans(M, centers=2)`
 - `clust <- makeKMeansDendrogram(kcl)`
 - `plot(clust, hang=-1, ann=F, axes=T)`
- Resultaat:



hclust resultaat

K-MEANS CLUSTERING: k ?

- Vaak vermoed je – op basis van voorkennis – hoeveel clusters er in de data zullen zitten, bijv.:
 - aantal verschillende pathways (dus aantal clusters genen)
 - aantal verschillende biologische samples (dus aantal clusters samples)
- Soms meerdere clusterings k proberen...
 - Is er een “logische” cluster toekenning, bijv. op basis van functionaliteit, biologisch sample?
 - Statistisch: is het “nodig” om meer clusters te maken?

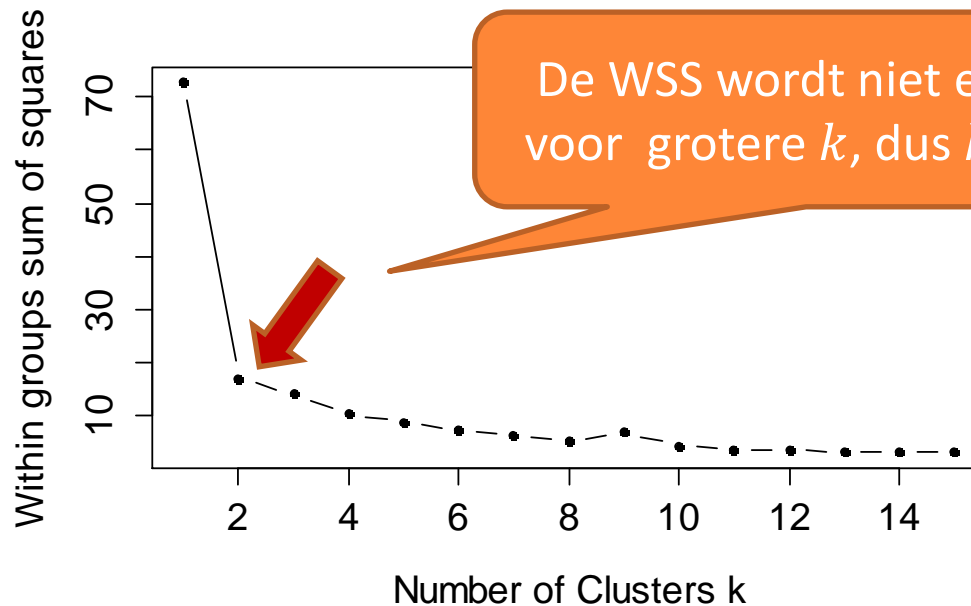


K-MEANS CLUSTERING: k ?

- Wat is een “goede” waarde voor k ?
- Plot de within-cluster sum of squares als functie van k :

```
# Determine number of clusters
wss <- (nrow(X)-1)*sum(apply(X,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(X, centers=i)$withinss)
plot(1:15, wss, type="b", pch=20,
     xlab="Number of Clusters k",
     ylab="Within groups sum of squares")
```

Robert I. Kabacoff, 2012



HCLUST() VS KMEANS()

- Hierarchisch clusteren:
 - **hclust**: clusteren tot 1 supercluster
 - **cutree**: opsplitsen in k subclusters
 - deterministisch (altijd zelfde resultaat)!
 - verschillende distance maten
- k -means clusteren:
 - **kmeans**: clusteren tot k clusters
 - niet deterministisch (i.h.a. bij herhaling verschillende resultaten)!
 - Euclidische distance
- k subclusters m.b.v. **hclust** zijn niet (altijd) k clusters m.b.v. **kmeans**

MEER “CLUSTER” FUNCTIES

- package **cluster**
 - `agnes ()`, `clara ()`, `clusplot ()`,
`daisy ()`, ...
- package **gplots**
 - `heatmap.2 ()`, `venn ()`, `redgreen (n)`,
`greenred (n)`, ...
- class **hclust**
 - `as.hclust ()`, ...
- class **dendrogram**
 - `as.dendrogram ()`, `dendrapply ()`, ...

Jullie kunnen nu de opdrachten van les 14 maken



Hanze University Groningen
APPLIED SCIENCES

Institute for
Life Science & Technology