Manag. Data Science

Cluster Analysis

Matthias Templ

Institute for Competitiveness and Communication

University of Applied Sciences and Arts Northwestern Switzerland
School of Business

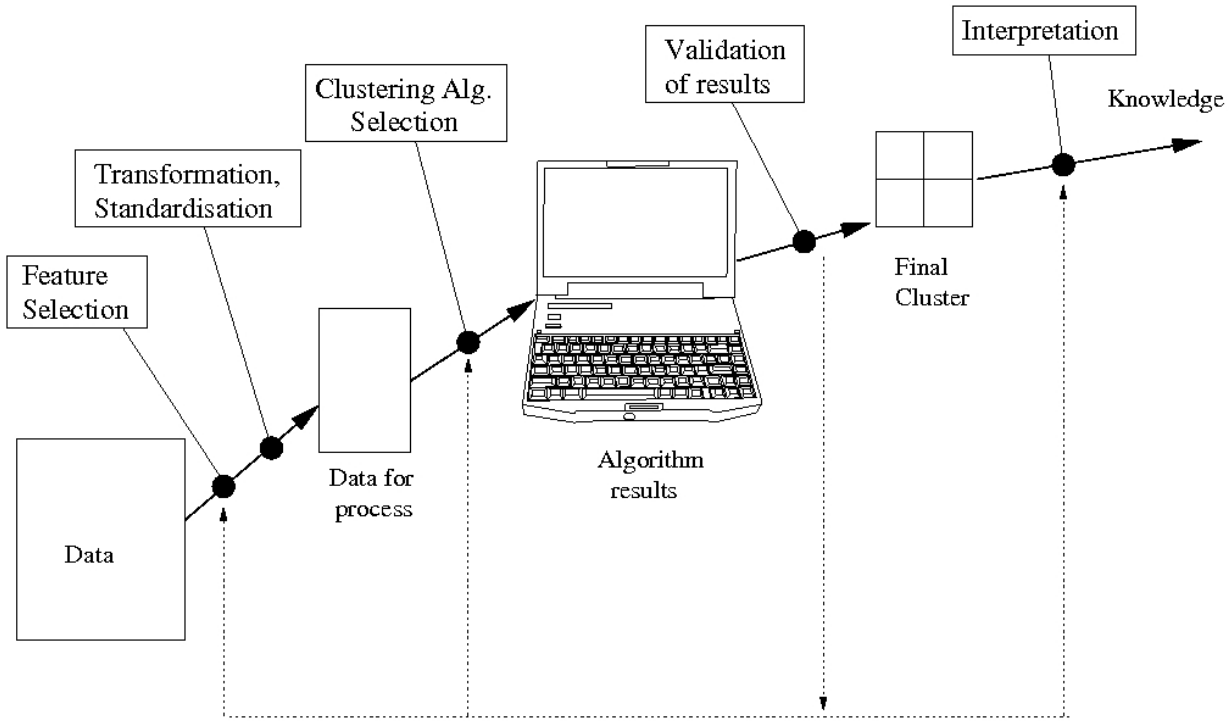
Wichtig von vorhin

- Distanzmasse (die Grundlage für die meisten Clusterverfahren)
- Transformation, Standardisierung
- Einleitung Clusteranalyse inklusive Mussel-Beispiel
- Kommentare zu Variablenclustering (auch genannt Q-Mode Clustering)
- Kommentare zu fehlenden Werten und Variablenselektion

Wiederholung: Was ist Clusteranalyse?

- **Exploratives** Tool um ähnliche Beobachtungen in Daten zu finden.

Etwas vereinfachter Ablauf einer Clusteranalyse



Unterschiedliche Ansätze

- **Visuell**: mit parallelen Koordinatenplot, heatmap der Distanzen, MDS und Tours
- **Hierarchisch**: in jedem Schritt werden Cluster grössser (aglomerativ) oder kleiner (divisiv).
- **Partitionierend**: jede Beobachtung wird genau einem Cluster zugeteilt
- **Fuzzyclustering** und **modellbasiertes Clustern**: jede Beobachtungen fällt in jedes Cluster mit einer bestimmten Höhe an Zugehörigkeiten.
- **Dichtebasierte Verfahren**: kann man sich als ausbreitende Epidemie vorstellen
- uva.

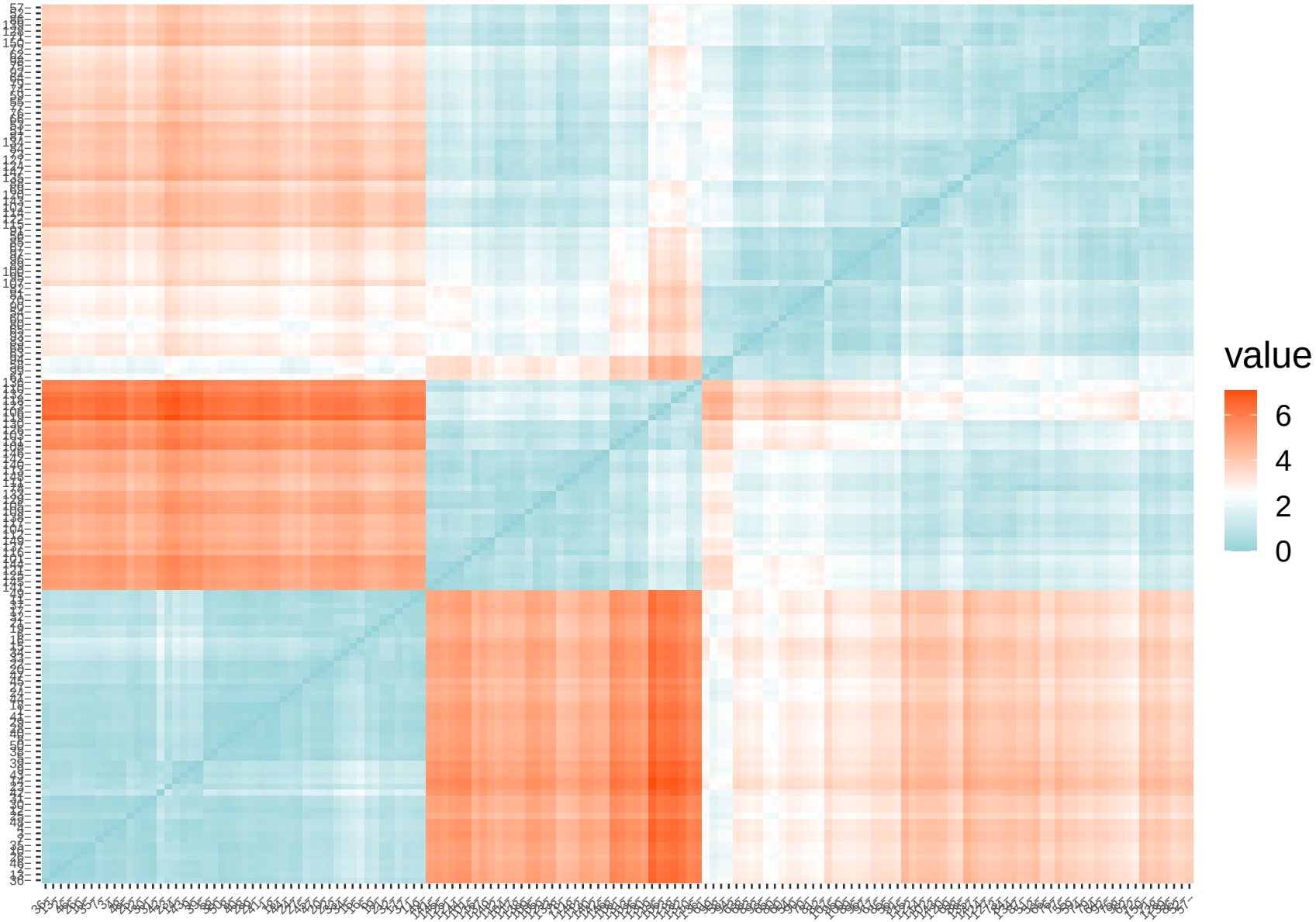
Nochmals Distanzen: Heatmap

Anhand der (sortierten) Distanzmatrix lässt sich bereits erahnen, ob eine gute Clusterstruktur vorliegt. Anhand der Iris-Daten sieht dies so aus (plot: nächste Seite)

```
library(factoextra)
distance <- get_dist(iris[, 2:5]) # oder dist()
fviz_dist(distance, order = TRUE,
          lab_size = 1,
          gradient = list(low = "#00AFBB",
                          mid = "white",
                          high = "#FC4E07"))
```

Falls Strukturen auffallen ist dies ein Indiz dafür, dass eine gute Clusterstruktur vorhanden ist.

Nochmals Distanzen: Heatmap



Nochmals Distanzen

Zum live Ausprobieren (da interaktiv)

```
library(heatmaply)
heatmaply(as.matrix(distance))
```

Die Dendrogramme rechts und oben werden wir als nächstes besprechen.

Ihre Mitarbeit ist gefragt

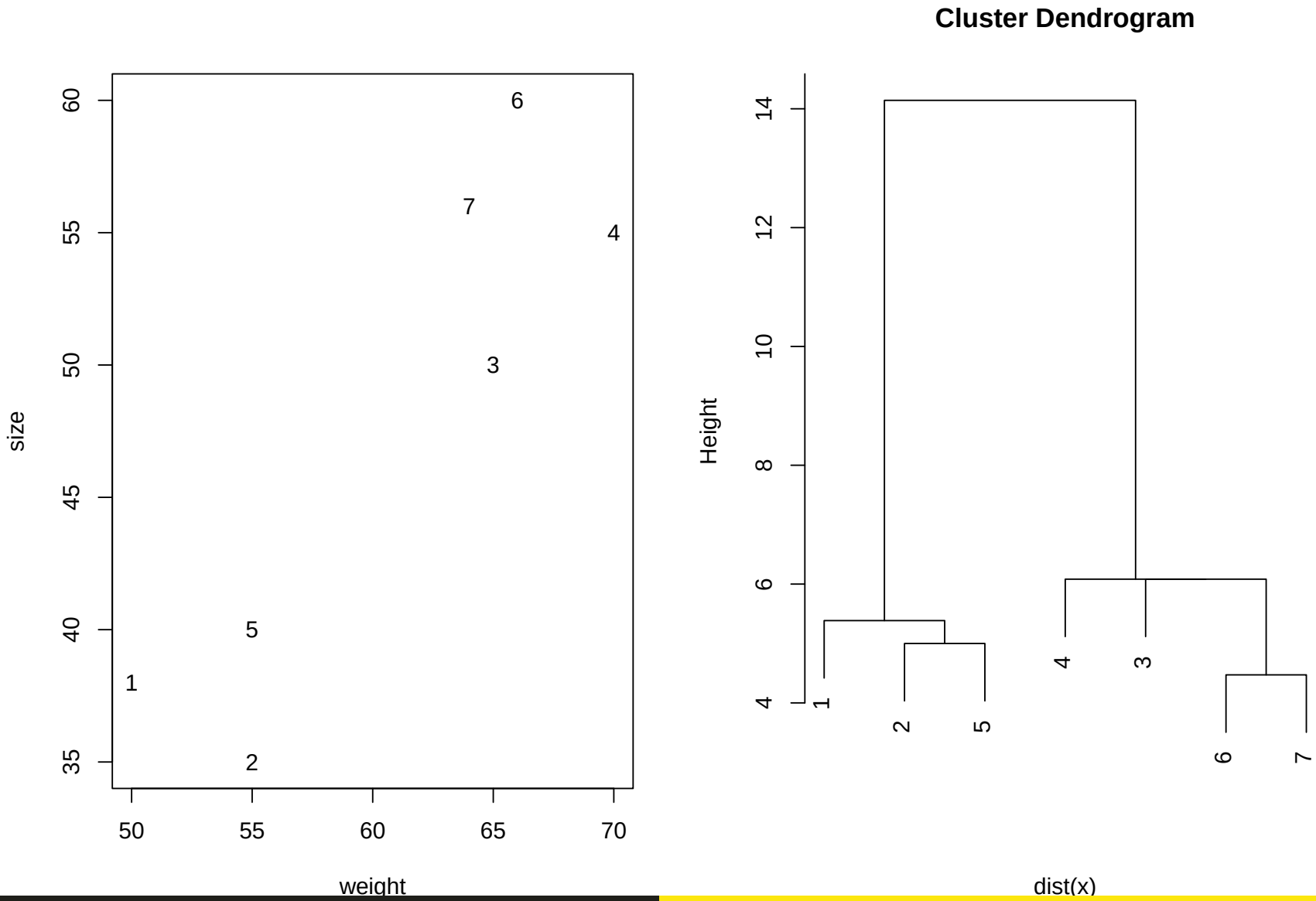
`all_exercises-cluster_nolsg.pdf`

Clustering

- Aufgabe 1: Visualisieren der Distanzen zwischen Beobachtungen

Hierarchisches Clustern

Dendrogramm: Beispiel Mussels, *Height* drückt die *Dissimilarity* aus



Hierarchisches Clustern

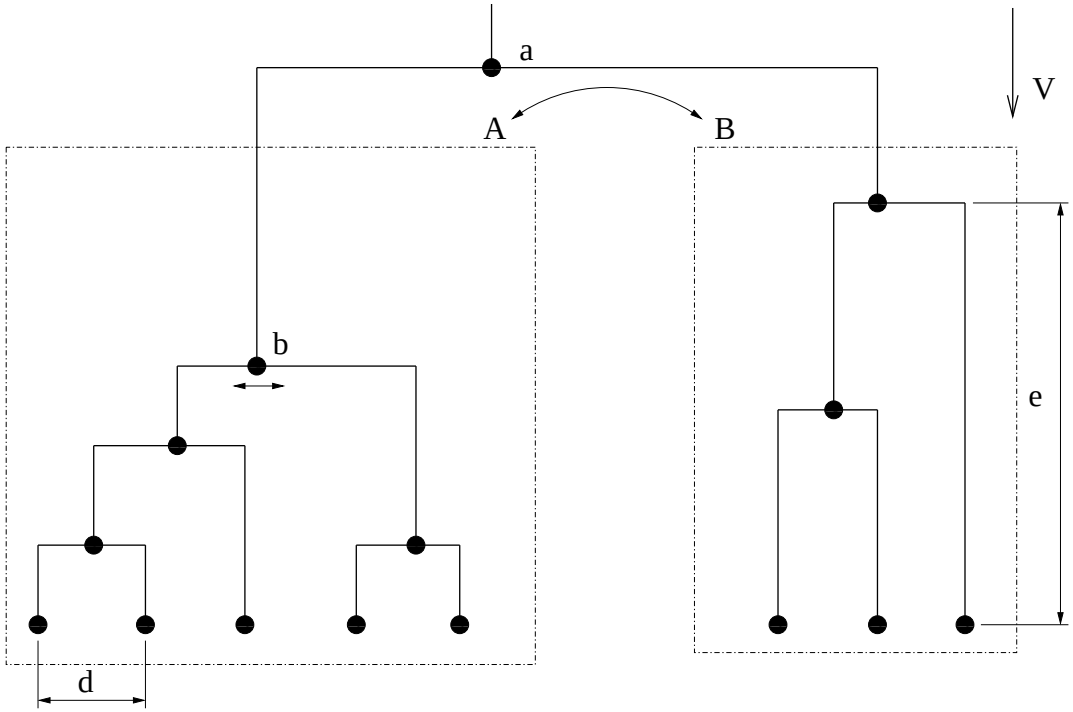
Sequenz von Clusterpartitionen welche mit einem Clustering Tree visualisiert wird, dem **Dendrogramm**.

Dendrogramm:

- Zunächst ist jeder Datenpunkt ein Cluster, diese werden dann sukzessive zusammengefügt.
- Die Fusion verschiedener Cluster wird im Dendrogramm mit einer waagrechten Linie gekennzeichnet.
- Auf der y-Achse des Dendrogramms wird die Heterogenität innerhalb der Cluster abgetragen, sie wächst mit zunehmender Clustergrösse an.
- Sind im Dendrogramm Bereiche mit langen senkrechten Linien, so zeigt dies einen grossen Anstieg an Heterogenität an.
- Die Messung der Heterogenität/Unähnlichkeit (dissimilarity) kann mit unterschiedlichen Methoden erfolgen.

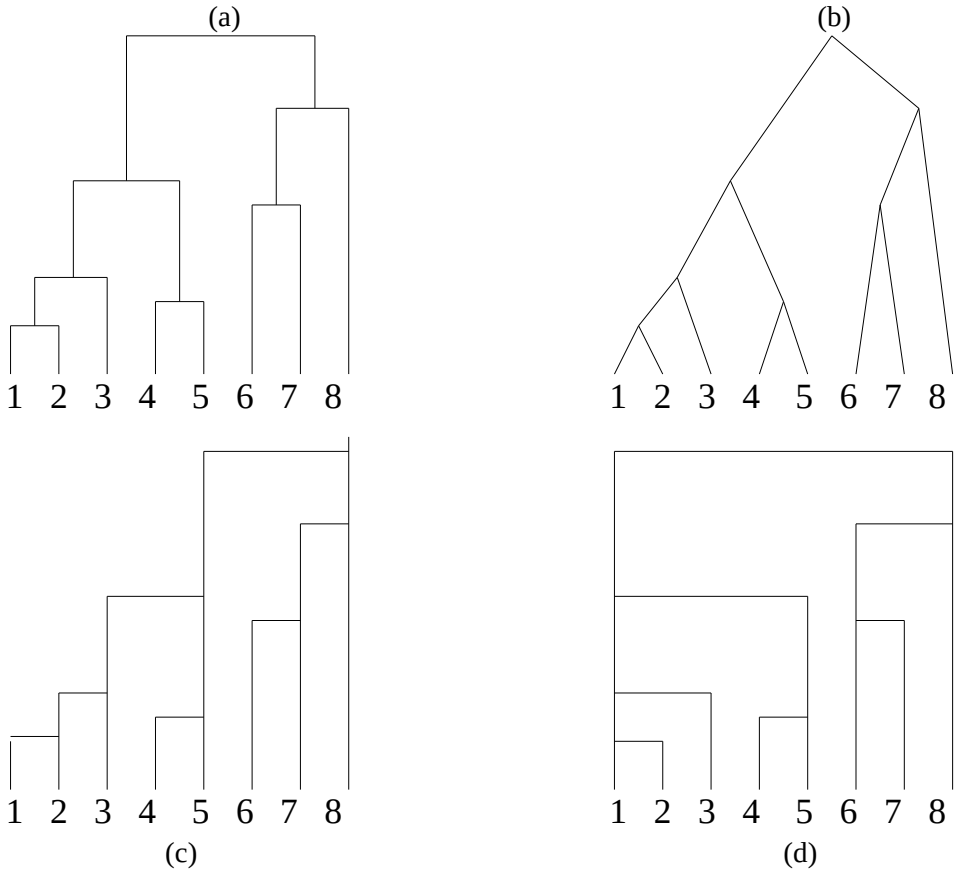
Hierarchisches Clustern

Einige freie Gestaltungsmöglichkeiten: A und B können vertauscht werden, b ist beliebig wählbar, ebenfalls d. c entspricht der Unähnlichkeit von Clustern.



Hierarchisches Clustern

Vier mögliche Dendrogramme, welche die selbe Info enthalten:



Hierarchisches Clustern (agglomerativ)

Nochmals genauer:

Start:

- Jedes Objekt ist ein eigener Cluster $\rightarrow$ n unterschiedliche Cluster.

Schrittweises Vorgehen:

- In jedem Schritt wird die Anzahl der Cluster um 1 reduziert, wobei die zwei ähnlichsten Klassen zusammengefasst werden.
  - Die *Disssimilarity* / Unähnlichkeit kann unterschiedlich gemessen werden (single linkage, complete linkage, average linkage, Ward, ...).
  - Eine Höhe wird dem neu erhaltenen Cluster zugeordnet
  - Am Ende ist nur mehr ein Cluster übrig, alle Objekte liegen in diesem Cluster.

Allgemeines Konzept (nur zur Übersicht!)

- Seien C_i und C_j zwei Cluster und das Unähnlichkeitsmasses (*dissimilarity*) zwischen diesen Clustern sei $d(C_i, C_j)$.
- Sobald diese beiden Cluster zusammengefügt werden (linked), gilt das folgende generalisierte Schema bzgl. der Unähnlichkeit zwischen $C_i \cup C_j$ und einem anderen Cluster C_k

$$\begin{aligned} d(C_i \cup C_j, C_k) &= \alpha_i d(C_i, C_k) + \alpha_j d(C_j, C_k) + \beta d(C_i, C_j) \\ &\quad + \gamma |d(C_i, C_k) - d(C_j, C_k)|, \end{aligned}$$

mit α_i, α_j, β, $\gamma \in \mathbb{R}$.

Allgemeines Konzept (nur zur Übersicht!)

Die gebräuchlichste Wahl der Parameter:

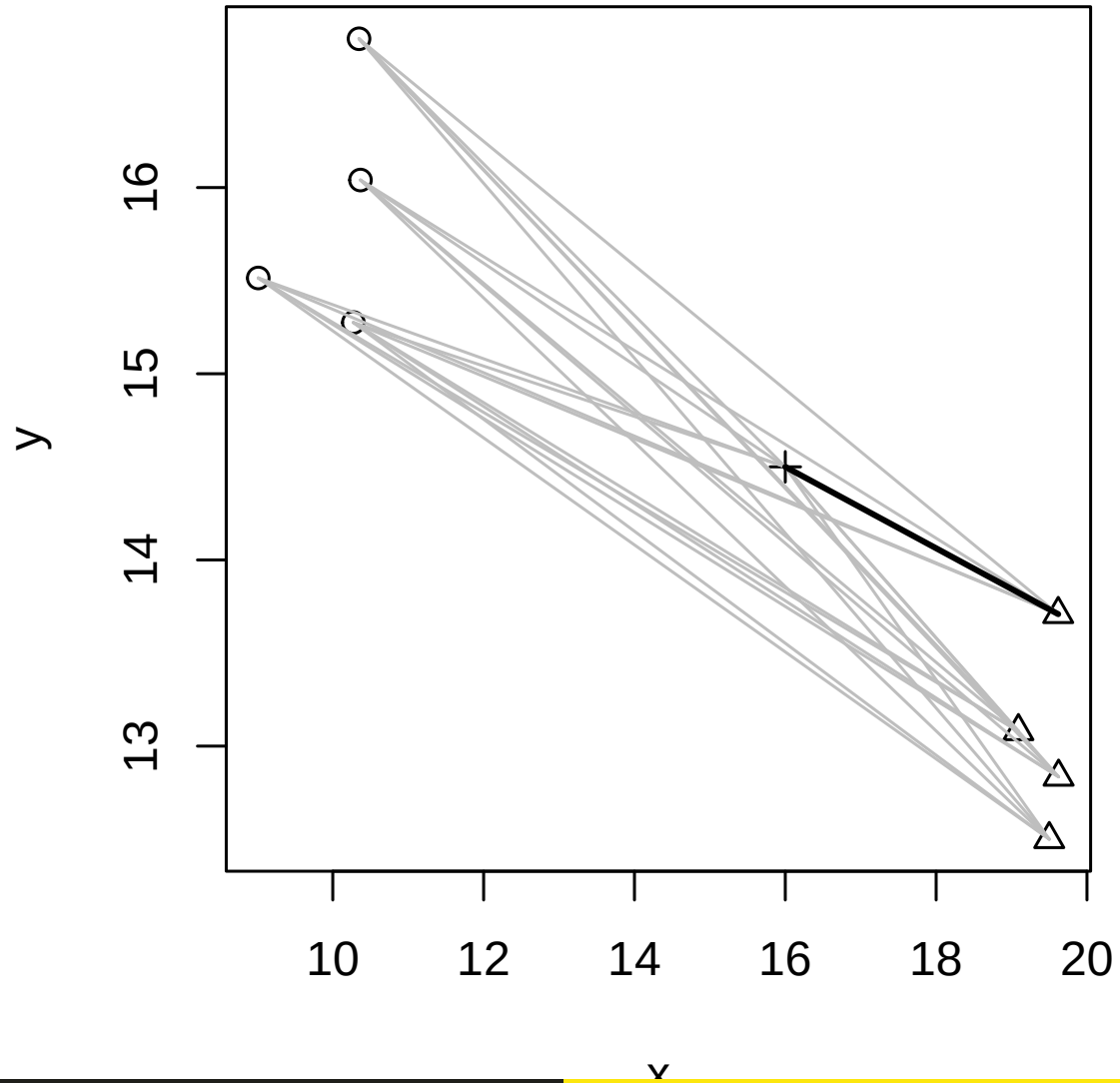
Clustering criterion	α_i, α_j	β	γ
Single linkage	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete linkage	$\frac{1}{2}$	0	$\frac{1}{2}$
Average linkage	$\frac{1}{2}$	0	0
Centroid linkage	$\frac{n_i}{n_i+n_j}$	$\frac{n_i n_j}{(n_i+n_j)^2}$	0
Ward's method	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{-n_k}{n_i+n_j+n_k}$	0

n_l ist die Anzahl an Beobachtungen in Cluster C_l $(l = i, j, k)$

Besser verständlich: folgende Abbildungen

Single Linkage (2-dim Beispiel)

Zu welchen Cluster (Beobachtungen mit $\circ$ oder mit $\triangle$) wird Beobachtung $+$ zugeordnet?



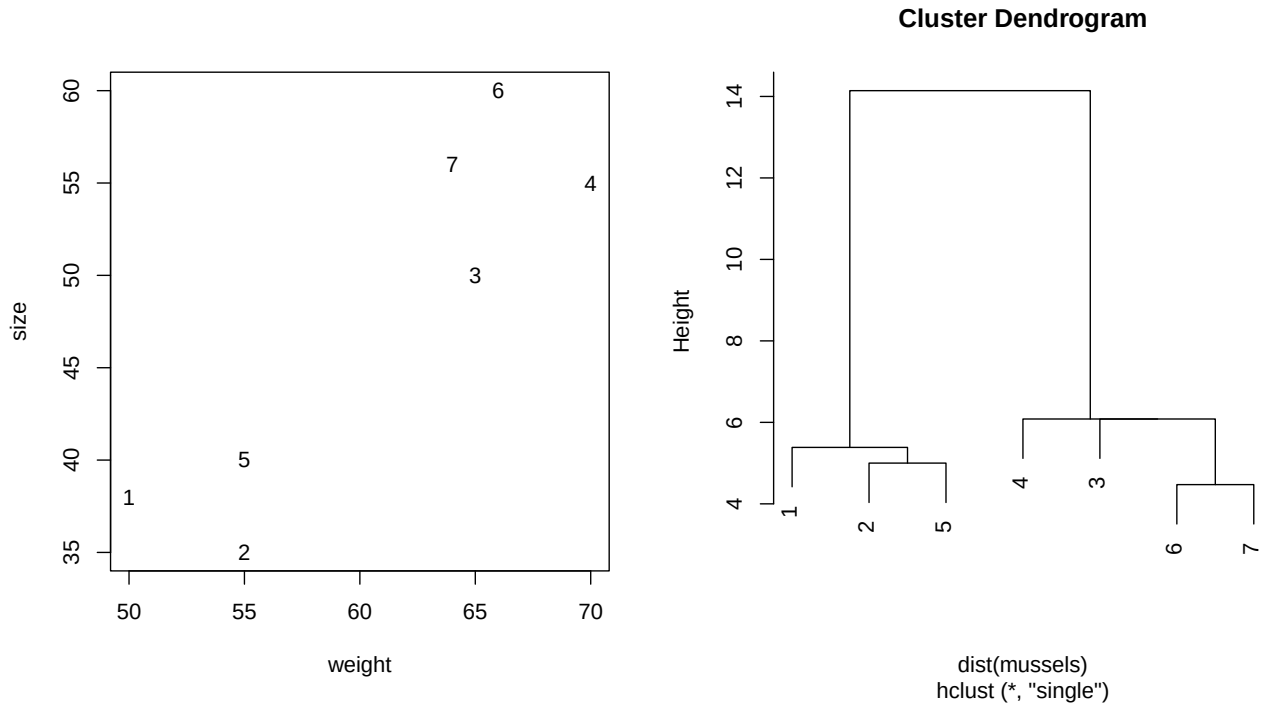
Single Linkage

- Tendiert zu unterschiedlich grossen Clustern, da grosse Cluster eher schnell zusammengefügt werden.
- Deshalb auch manchmal zur Ausreissererkennung verwendet.
- Kann sehr effizient berechnet werden.

Single Linkage (2-dim Beispiel, Daten: mussels)

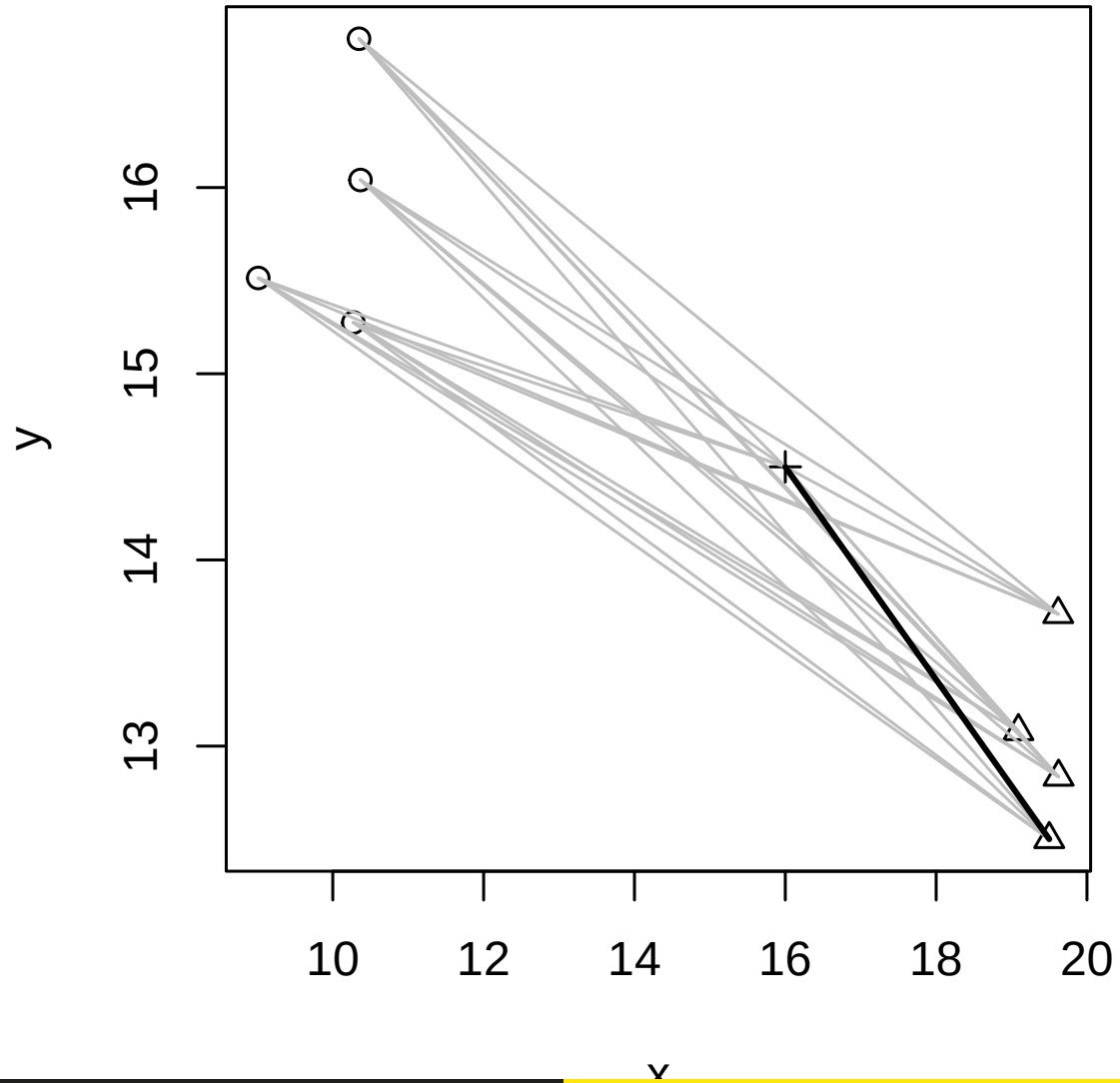
Wiederholung, diesmal mit Code:

```
par(mfrow = c(1,2), pty = "s")
plot(mussels, type = "n");text(mussels, rownames(mussels))
plot(hclust(dist(mussels), method = "single"))
```



Complete Linkage (2-dim Beispiel)

Von jedem Cluster die am weitest entfernte Beobachtung, zu welchen Cluster ist diese Distanz am geringsten . . .



Complete Linkage und weitere Linkage Kriterien

Complete Linkage:

- führt zu eher gleich grossen Clustern.
- Rechenzeit gering

Average Linkage:

- nimmt mittlere Distanzen statt minimale (single linkage) oder maximale (complete linkage), dafür steigt die Rechenzeit etwas.

Ward Methode:

- in jedem Schritt wird jede mögliche Vereinigung mit einem Informationsverlustkriterium beurteilt, daher rechenintensiver
- das Informationsverlustkriterium ist zumeist der quadrierte Abstand zu den Clustermitteln
- Auswahl jener Vereinigung mit minimalen Anstieg des Informationskriteriums

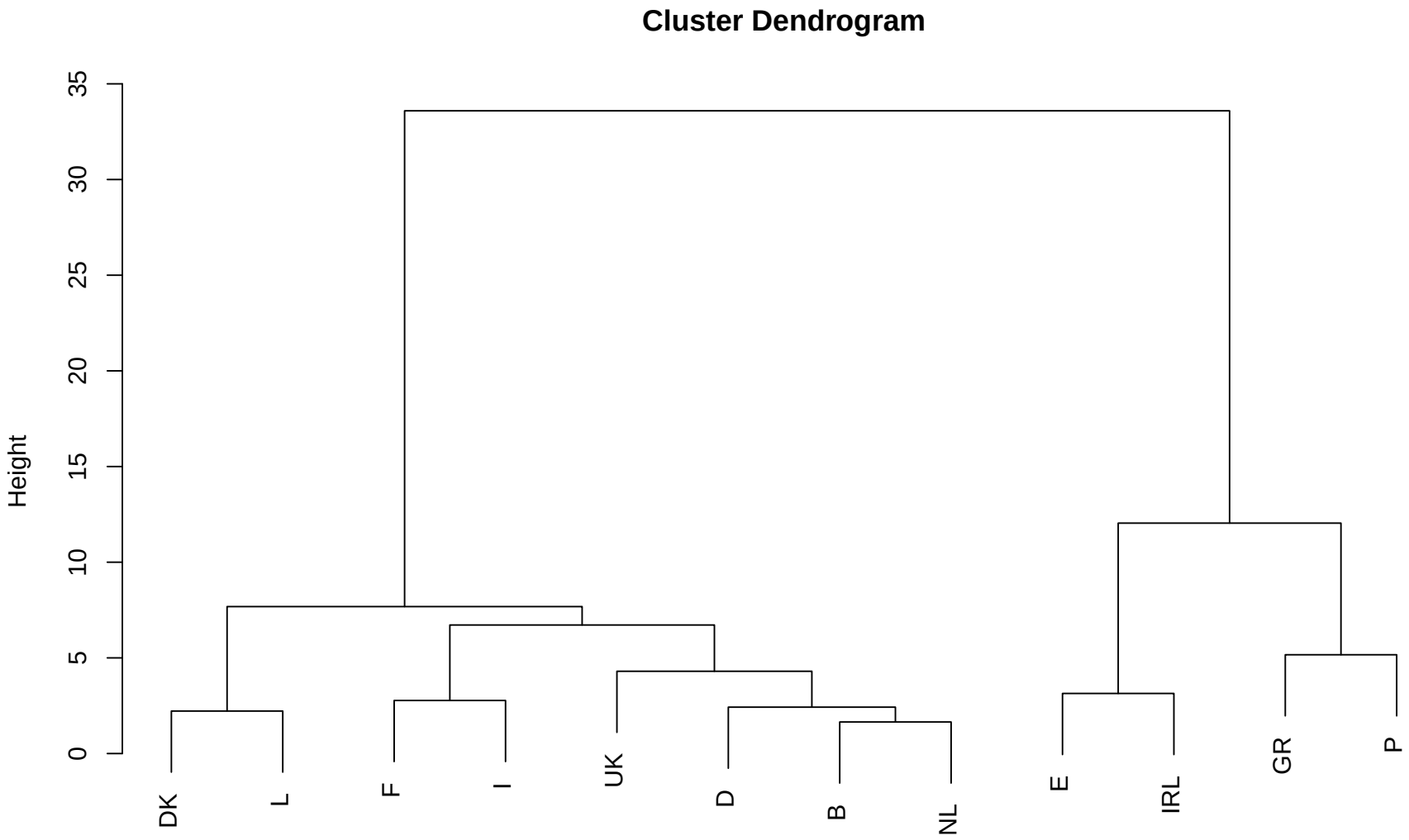
Beispiel hierarchisches Clustern

```
data(agriculture, package = "cluster")
# ?agriculture
colnames(agriculture) <- c("GDP", "Agriculture")
agriculture
```

```
##      GDP Agriculture
## B   16.8         2.7
## DK  21.3         5.7
## D   18.7         3.5
## GR   5.9        22.2
## E   11.4        10.9
## F   17.8         6.0
## IRL 10.9        14.0
## I   16.6         8.5
## L   21.0         3.5
## NL  16.4         4.3
## P    7.8        17.4
```

Beispiel hierarchisches Clustern (Ward)

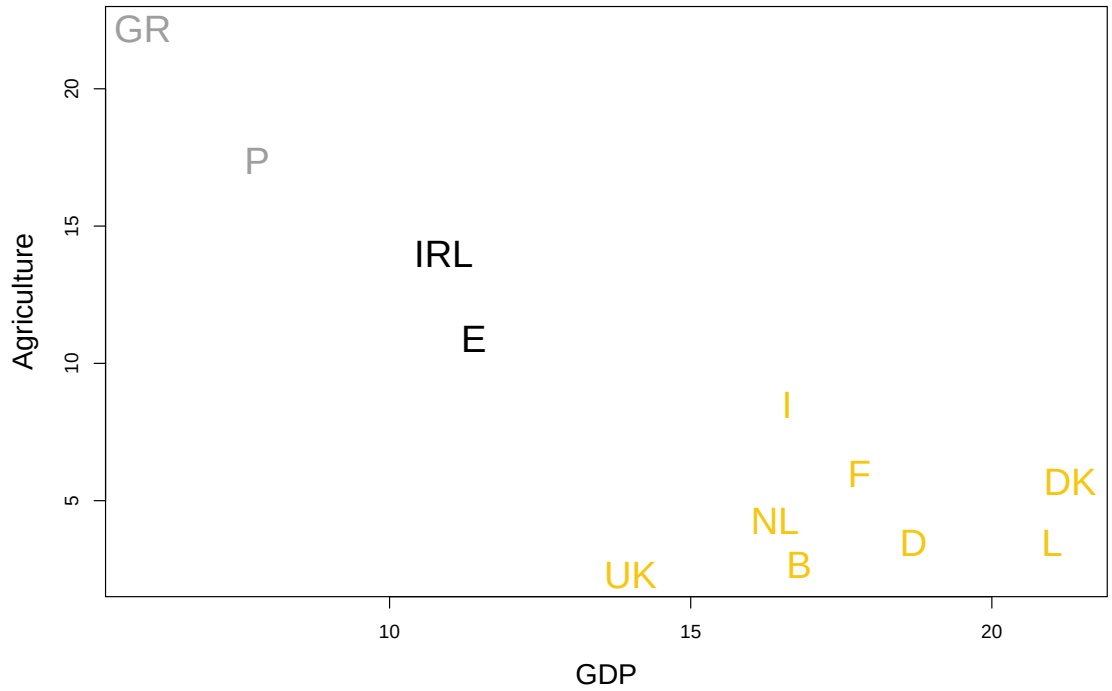
```
cl <- hclust(dist(agriculture), method = "ward.D2")
plot(cl)
```



Beispiel hierarchisches Clustern: cutree

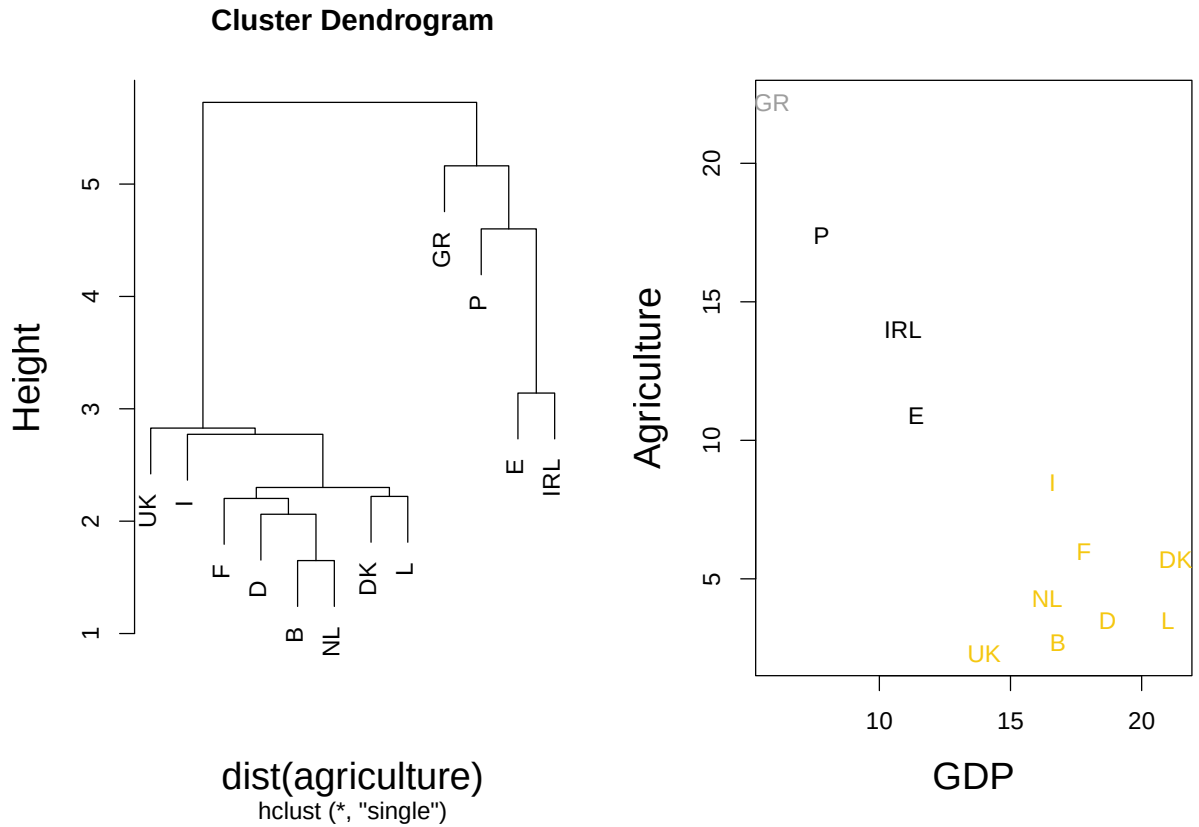
Durchschneiden des Histogramms mit Funktion `cutree` auf jener Höhe, sodass z.B. genau 3 Cluster entstehen.

```
ct <- cutree(cl, 3)
plot(agriculture, type = "n", cex.lab = 1.5)
text(agriculture, rownames(agriculture),
     col = ct+6, cex = 2) # plot mit agriculture
```



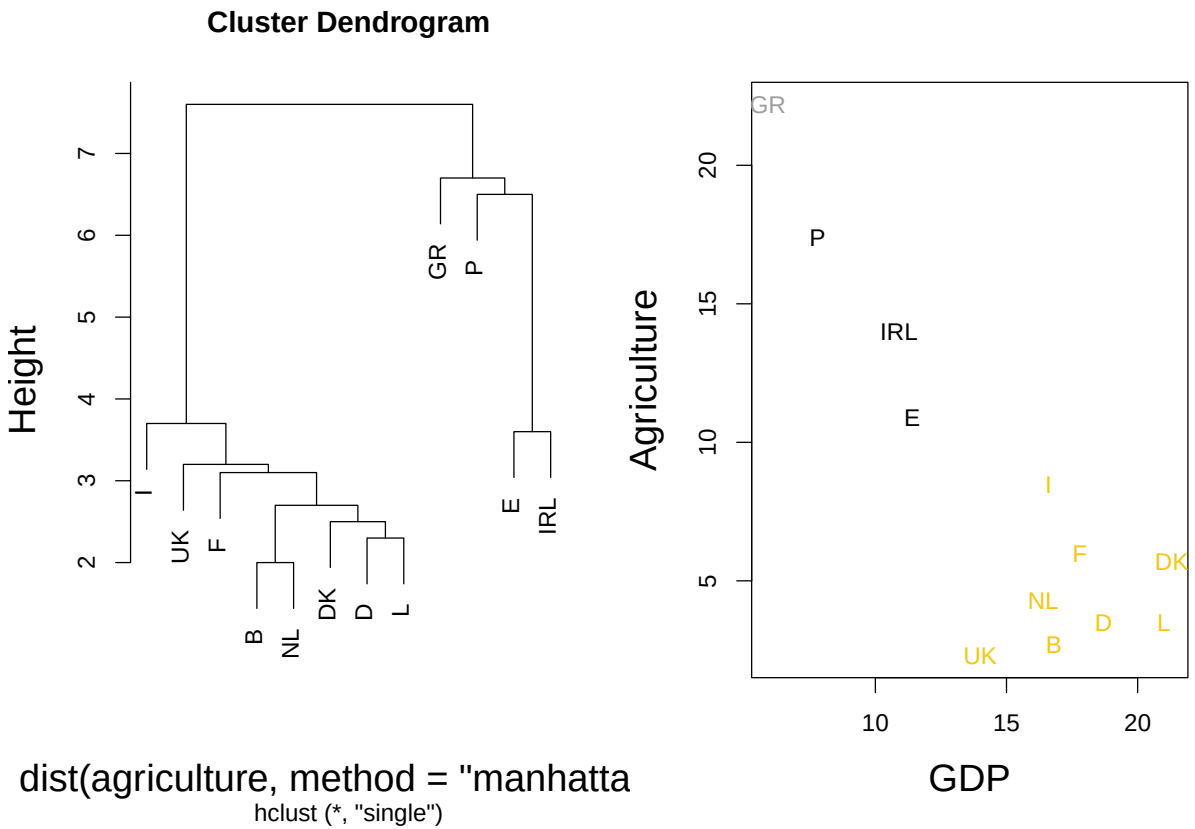
Beispiel hierarchisches Clustern

```
cl <- hclust(dist(agriculture), method = "single")
ct <- cutree(cl, 3)
par(mfrow = c(1, 2), cex.lab = 1.5, cex = 1.2)
plot(cl); plot(agriculture, type = "n", cex = 2)
text(agriculture, rownames(agriculture), col = ct+6)
```



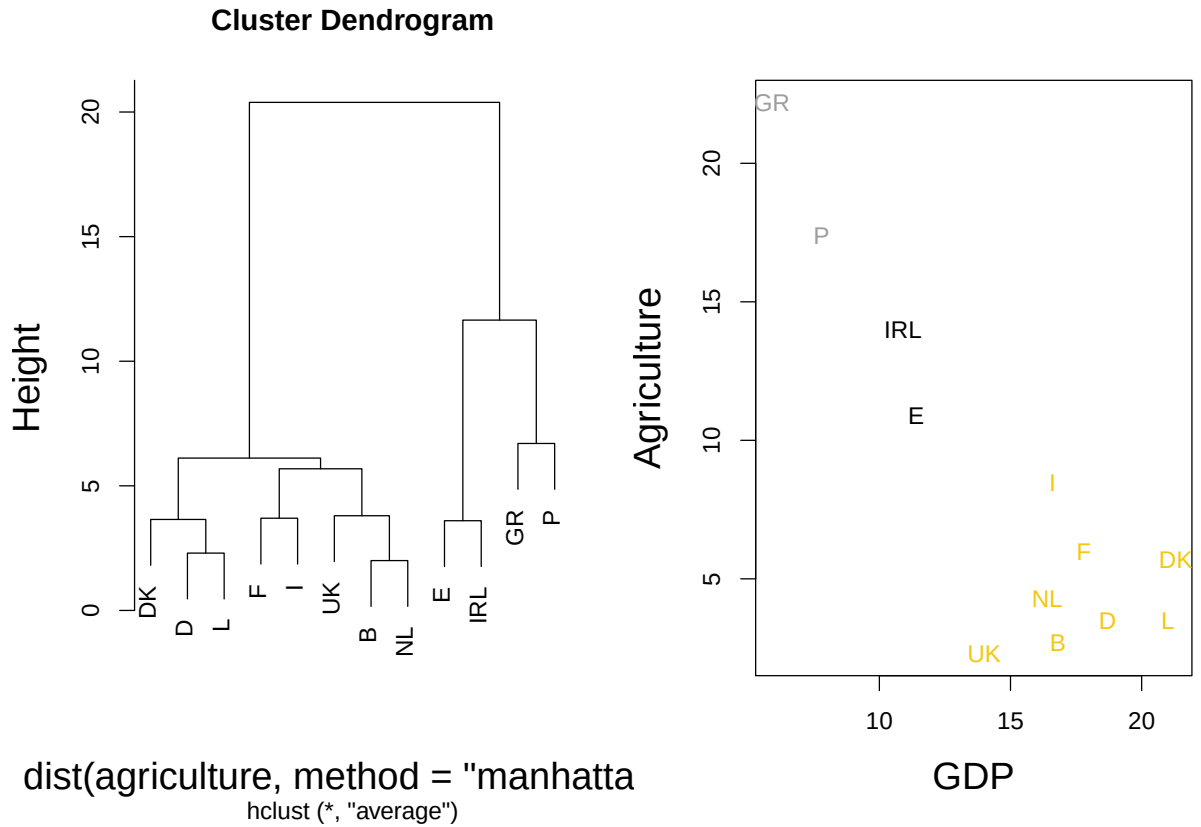
Beispiel hierarchisches Clustern

```
cl <- hclust(dist(agriculture, method = "manhattan"),
             method = "single"); ct <- cutree(cl, 3)
par(mfrow = c(1, 2), cex.lab = 1.5, cex = 1.2)
plot(cl); plot(agriculture, type = "n", cex = 2)
text(agriculture, rownames(agriculture), col = ct+6)
```



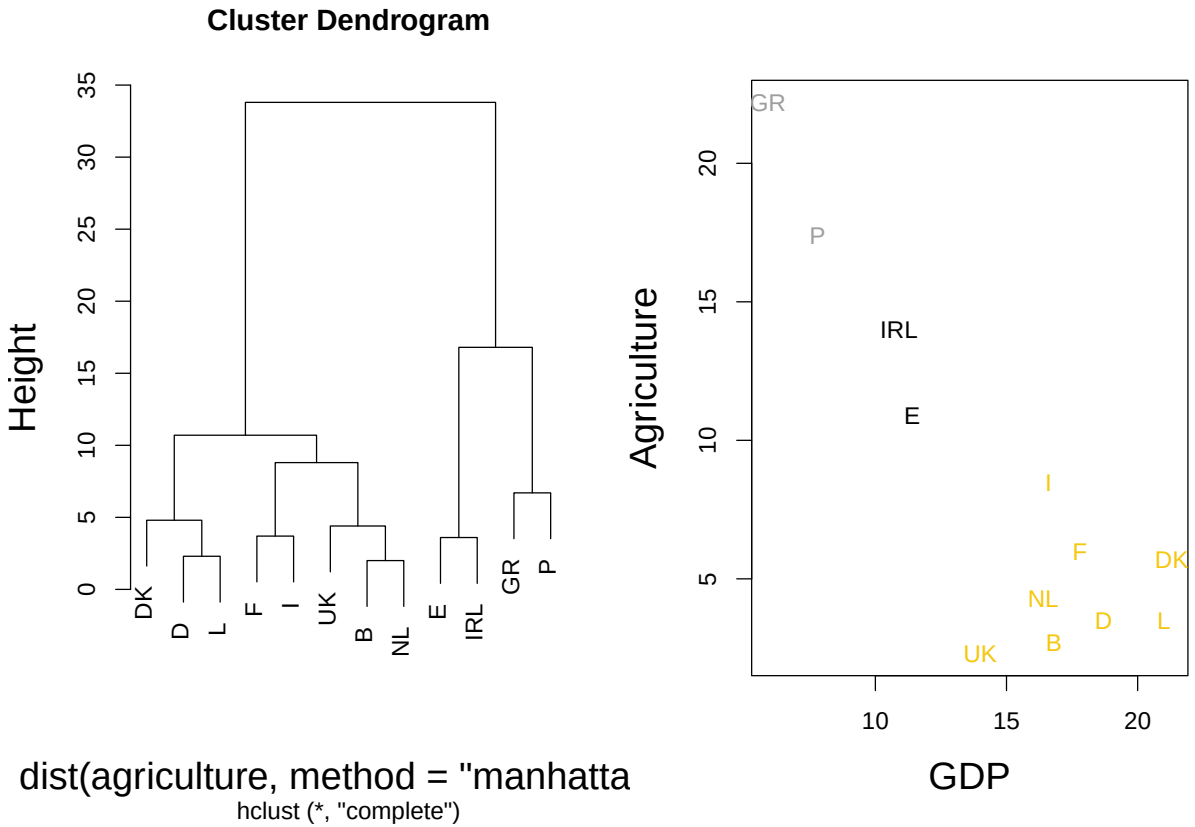
Beispiel hierarchisches Clustern

```
cl <- hclust(dist(agriculture, method = "manhattan"),
             method = "average"); ct <- cutree(cl, 3)
par(mfrow = c(1, 2), cex.lab = 1.5, cex = 1.2)
plot(cl); plot(agriculture, type = "n", cex = 2)
text(agriculture, rownames(agriculture), col = ct+6)
```



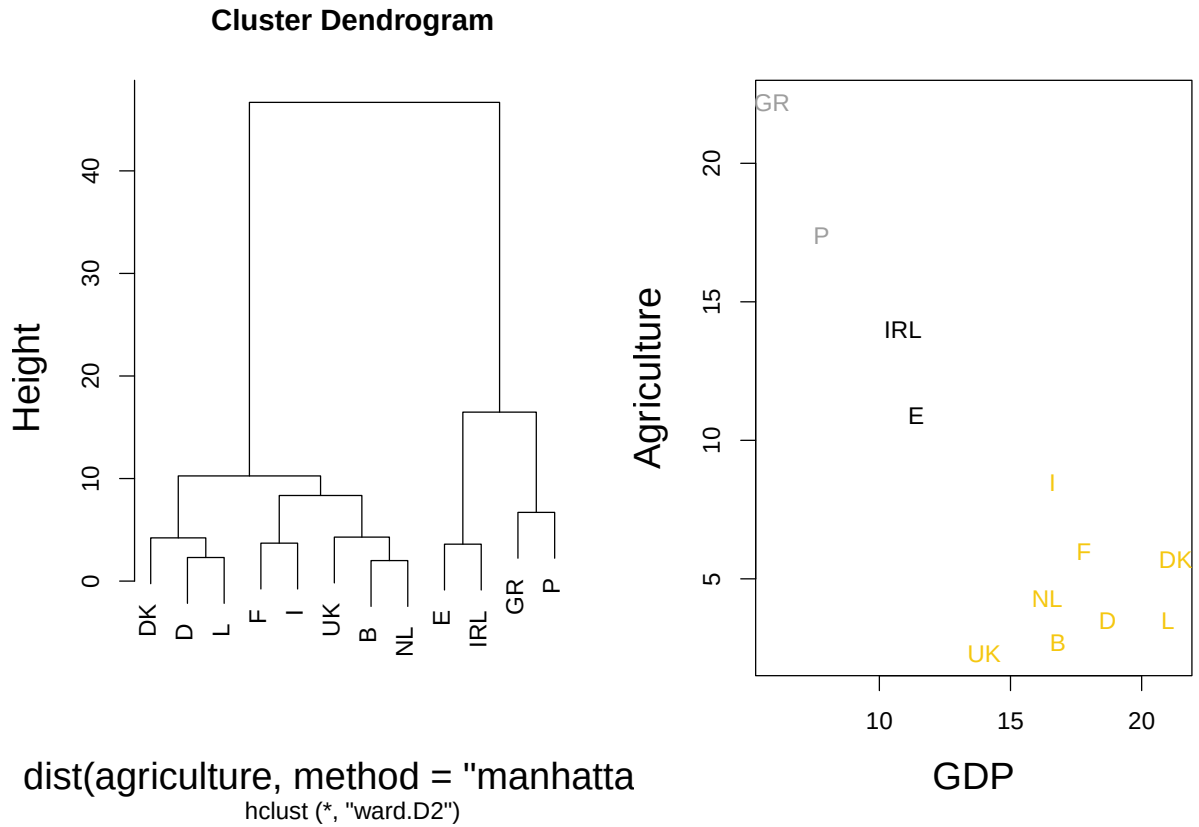
Beispiel hierarchisches Clustern

```
cl <- hclust(dist(agriculture, method = "manhattan"),
             method = "complete"); ct <- cutree(cl, 3)
par(mfrow = c(1, 2), cex.lab = 1.5, cex = 1.2)
plot(cl); plot(agriculture, type = "n", cex = 2)
text(agriculture, rownames(agriculture), col = ct+6)
```



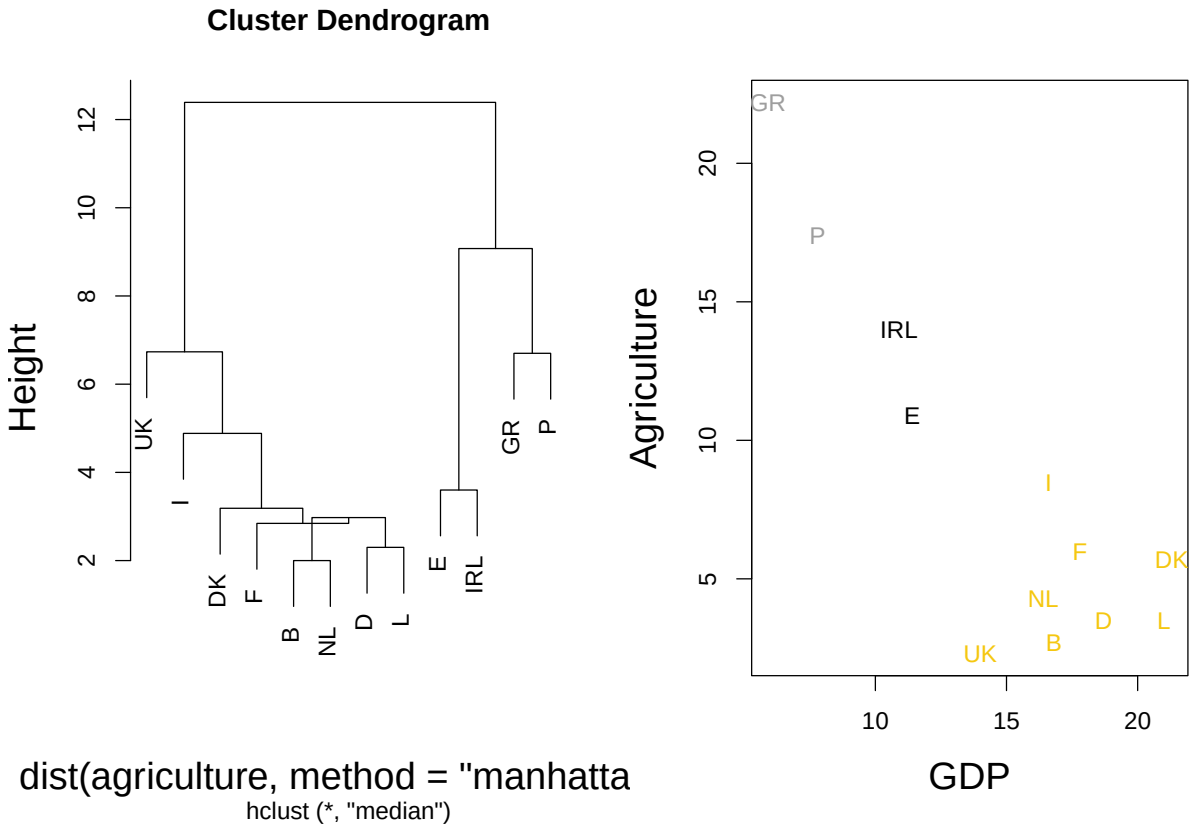
Beispiel hierarchisches Clustern

```
cl <- hclust(dist(agriculture, method = "manhattan"),
             method = "ward.D2"); ct <- cutree(cl, 3)
par(mfrow = c(1, 2), cex.lab = 1.5, cex = 1.2)
plot(cl); plot(agriculture, type = "n", cex = 2)
text(agriculture, rownames(agriculture), col = ct+6)
```



Beispiel hierarchisches Clustern

```
cl <- hclust(dist(agriculture, method = "manhattan"),
             method = "median"); ct <- cutree(cl, 3)
par(mfrow = c(1, 2), cex.lab = 1.5, cex = 1.2)
plot(cl); plot(agriculture, type = "n", cex = 2)
text(agriculture, rownames(agriculture), col = ct+6)
```



Ihre Mitarbeit ist gefragt

`all_exercises-cluster_nolsg.pdf`

Clustering

- Aufgabe 2: Hierarchisches Clustern und Dendrogramm
- Aufgabe 3: Q-mode Clustering (Variablenclustering)

Partitionierende Verfahren

1. Schritt: Fixiere Anzahl der Cluster
2. Schritt: Wende eine partitionierende Clustermethode an

Restriktionen:

- Jede Beobachtung fällt genau in einen Cluster
- Jeder Cluster enthält zumindest eine Beobachtung

> *Das bekannteste und mit Abstand am öftesten verwendete Verfahren ist der E(xpection)M(aximum)-Algorithmus **k-means**.*

Wir werden später noch (zumeist) bessere Verfahren kennen lernen.

k-means. Mittelwertsvektoren

- Ausgangspunkt ist Datenmatrix $\mathbf{X}$ mit n Beobachtungen und p Variablen.
- Ziel: Weise Beobachtungen zu den n_c Clustern $\{C_1, C_2, \ldots, C_{n_c}\}$ zu, sodass Cluster C_k insgesamt $n_{(k)}$ Mitglieder hat und jede Beobachtung genau einem Cluster zugeteilt ist.
- Die p Komponenten des Mittelwertsvektor (Centroid, Zentrum oder auch Prototype genannt) $\mathbf{v}_k$ eines Cluster C_k können folgendermassen berechnet werden.

$$\mathbf{v}_k (\in \mathbb{R}^p) = \left(\frac{1}{n_{(k)}} \sum_{i=1}^{n_{(k)}} x_{i1}^{(k)}, \ldots, \frac{1}{n_{(k)}} \sum_{i=1}^{n_{(k)}} x_{ip}^{(k)} \right)^T,$$

wobei $\mathbf{x}_i^{(k)} = (x_{i1}^{(k)}, \ldots, x_{ip}^{(k)})'$ die i-te Beobachtung bezeichnet die dem Cluster C_k zugewiesen ist. Für jedes Cluster $C_1, \ldots, C_{n_c}$ werden die Centroide $\mathbf{v}_1, \ldots, \mathbf{v}_{n_c}$ berechnet.

k-means. EM-Schritte

Bei voriger Initialisierung der Anzahl der Cluster n_c wird ebenfalls die Startlokation (die Centroide) der Cluster n_c initialisiert.

Der Algorithmus iteriert nun indem er immer die Beobachtungen zu den nächsten Centroiden zuweist:

1) Fixiere eine initiale Partition mit n_c Clusters.
2) E-step: (re)compute die Centroide mit den derzeitigen Clusterzugehörigkeiten (Memberships).
3) M-step: Weise jedes Objekt zum nähersten Clustercentroid zu $\rightarrow$ neue Zugehörigkeiten.
4) Gehe zu 2) bis die Memberships und daher die Centroide sich nicht mehr als eine sehr kleine Konstante ändern.

k-means. Zielfunktion

k-means Clustering optimiert somit die Zielfunktion

$$J(\mathbf{X}, \mathbf{V}, \mathbf{U}) = \sum_{k=1}^{n_c} \sum_{i=1}^{n} u_{ik} d^2(\mathbf{x}_i, \mathbf{v}_k), \tag{1}$$

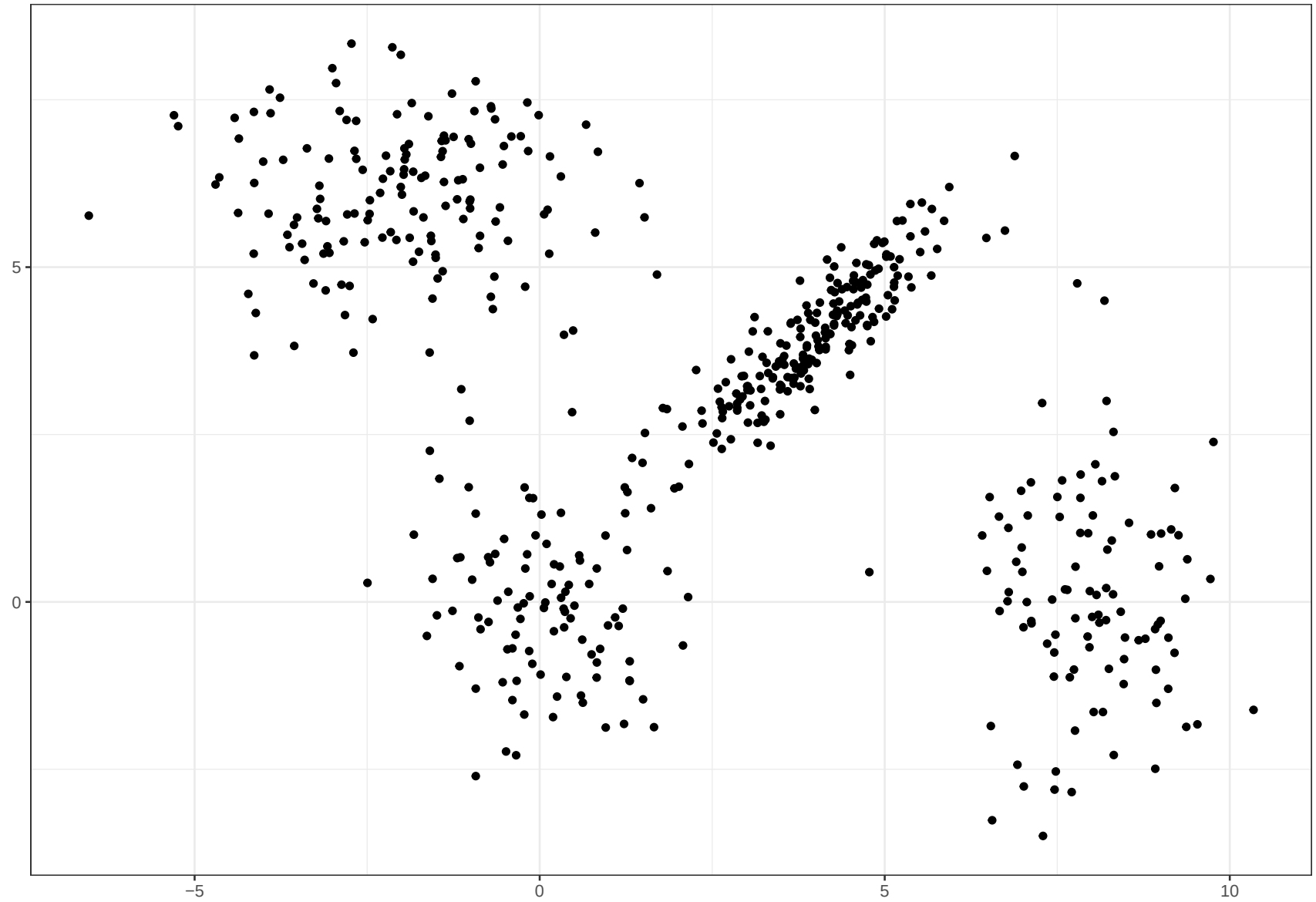
mit

- $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_{n_c})$ die Matrix der Clusterzentren der Dimension $p \times n_c$ und
- $\mathbf{U} = (u_{ik})$ ist eine $n \times n_c$ Matrix der Membership Koeffizienten u_{ik} für Beobachtung $\mathbf{x}_i$ zu einem Cluster C_k.
- Die Euklidische Distanz d misst die Distanz zwischen Beobachtungen und Clusterzentren.

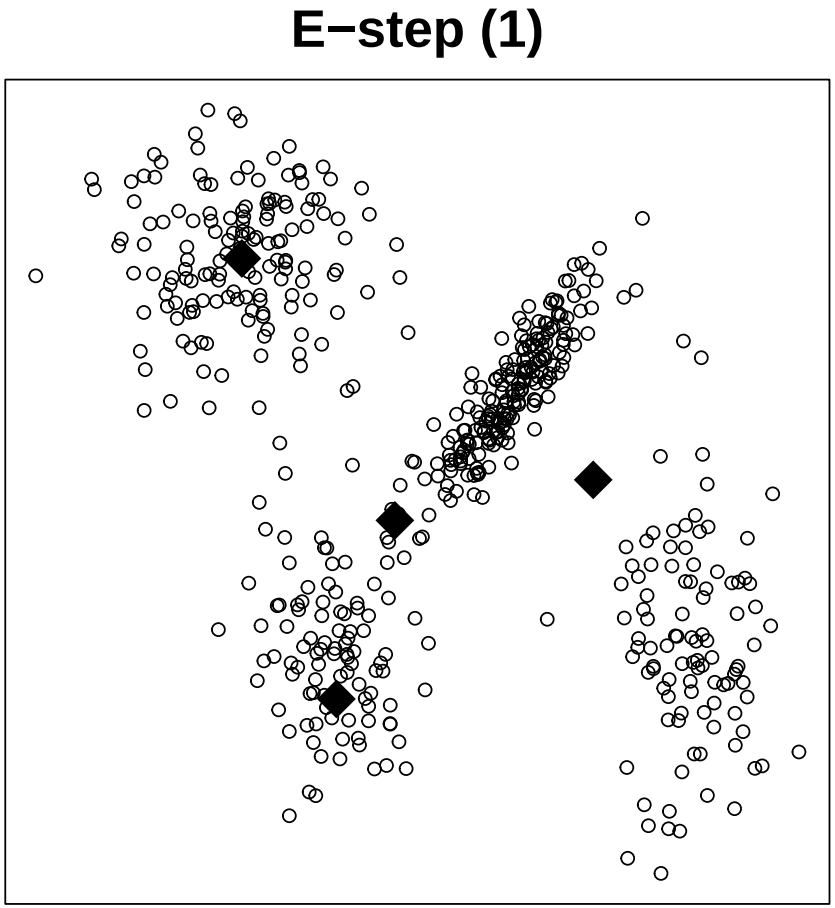
k-means. Bemerkungen

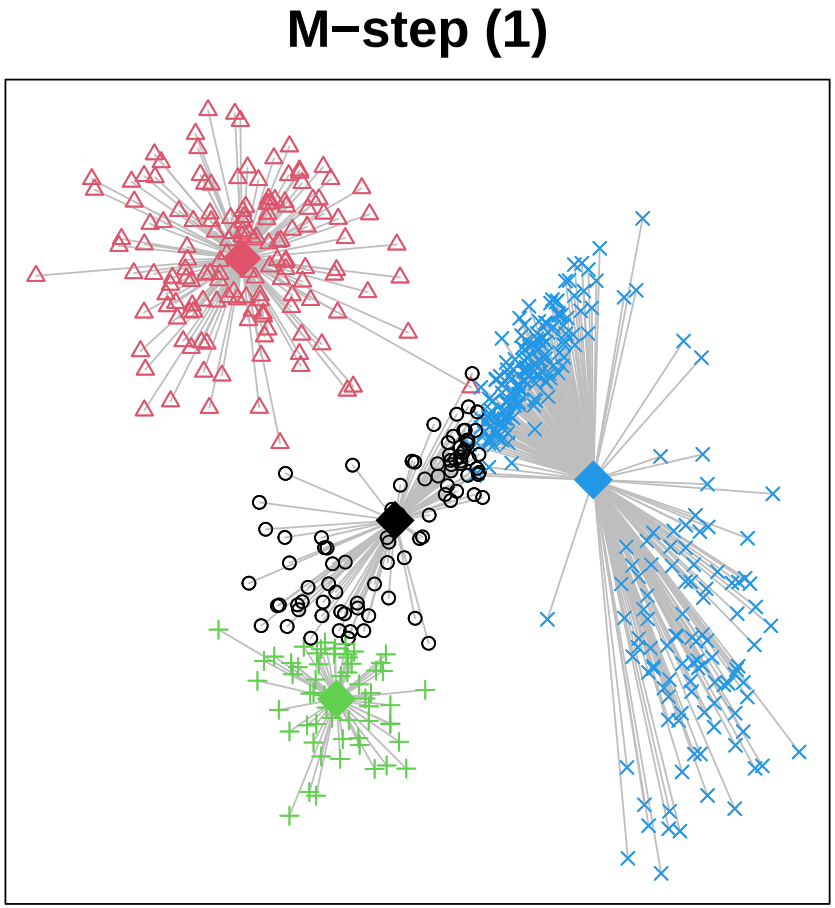
- der E-Step ist der Schätzschritt (Clusterzentren werden berechnet), der M-Step der Zuweisungsschritt.
- Iteration zwischen E- und M-Step verbessert die Lösung schrittweise, $J(\mathbf{X}, \mathbf{V}, \mathbf{U})$ wird mit jeder Iteration kleiner.
- der Algorithmus ist sehr schnell, kann auch parallelisiert werden und ist auch für relativ grosse Datensätze geeignet.

k-means. Funktionsweise visuell

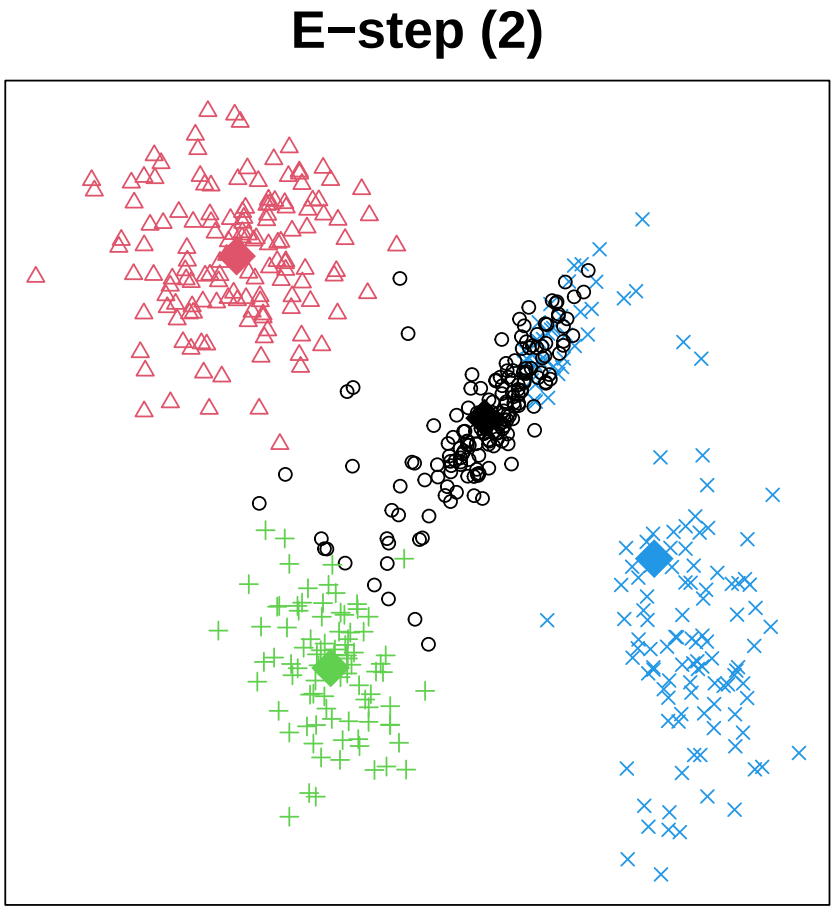


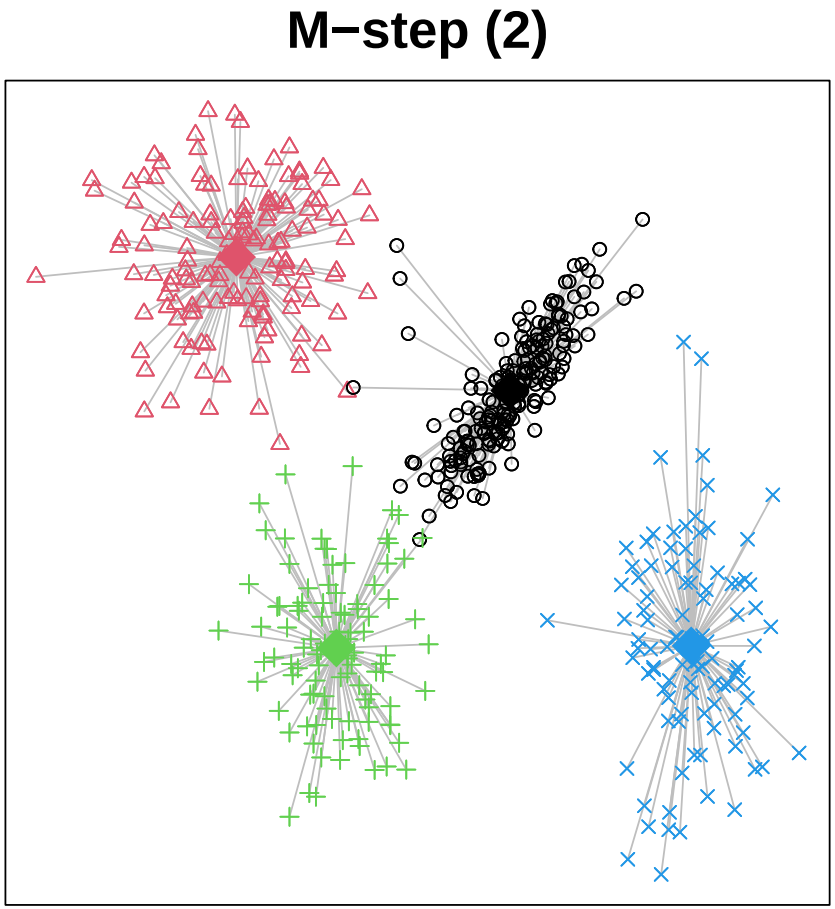
k-means. Funktionsweise visuell



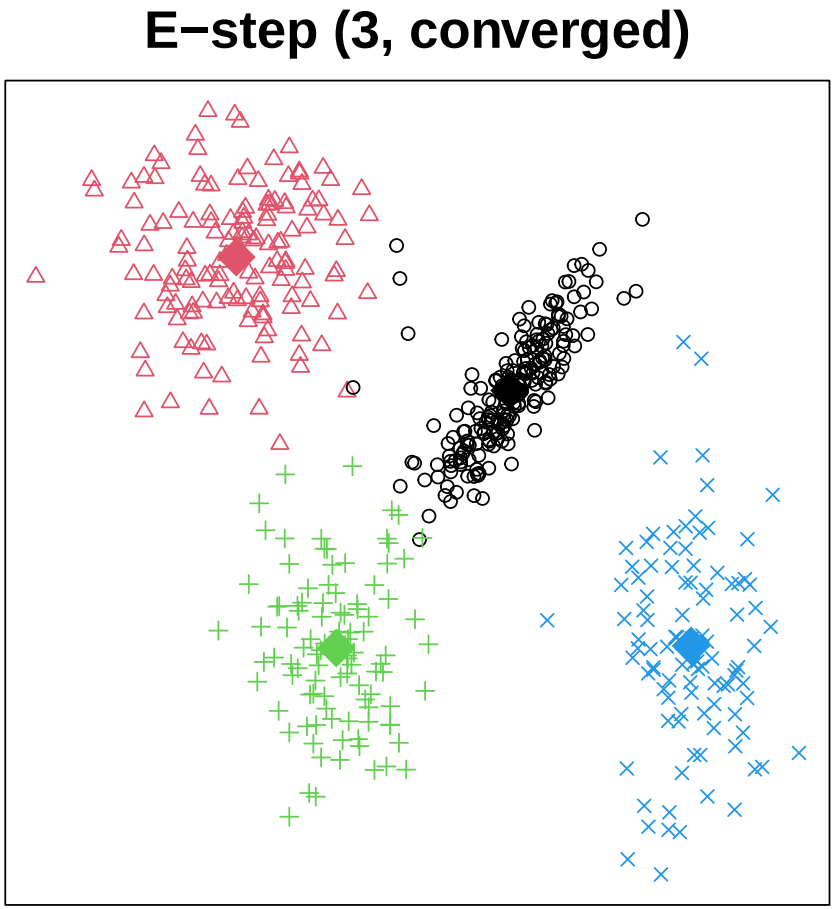


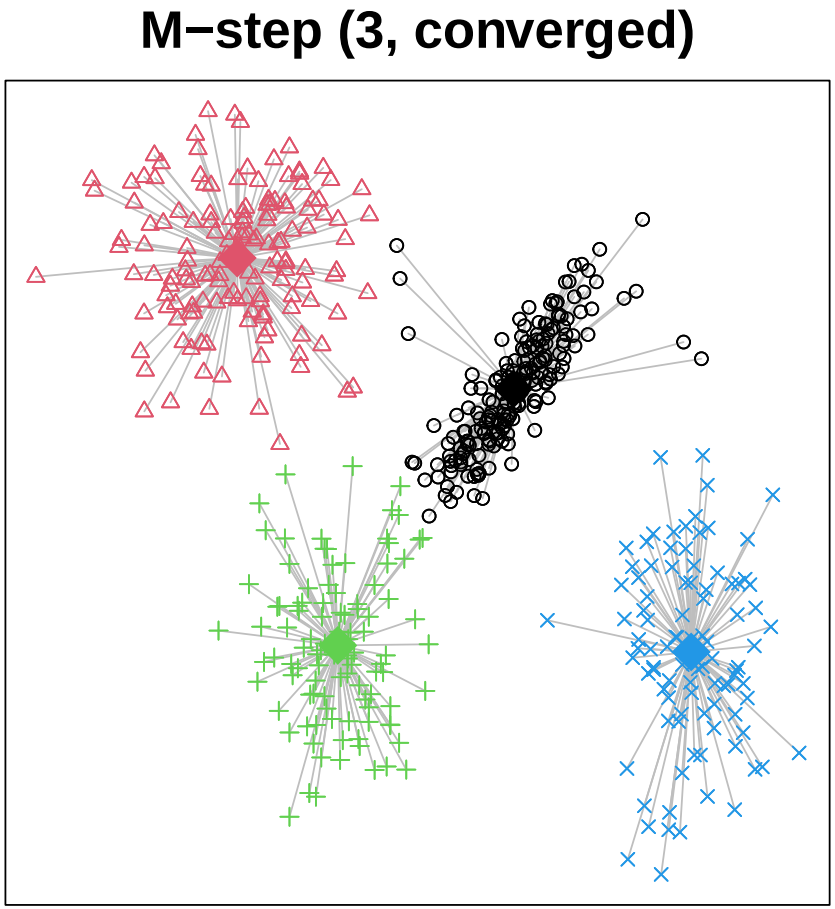
k-means. Funktionsweise visuell





k-means. Funktionsweise visuell





k-means. Modifikationen

- Statt Euclidischen Distanzen können andere Distanzen verwendet werden (z.B. Manhattan). Dann heisst er aber nicht mehr kmeans, sondern kmedians oder `cluster::pam` (partioning around medoids)
- Für sehr grosse Datensätze kann geschickt gesampled werden (Funktion `cluster::clara` - dies ist aber ein kmedians Algorithmus)
- Standardfunktion in R ist `kmeans`
- Sehr viele `R`-Pakete die kmeans anbieten

k-means. Nachteile

kmeans wird sehr oft angewandt, ist aber eine schlechte Methode.

- Arithmetische Mittel sind hoch nicht-robust. Besser: kmedians - statt artithm. Mittel, Mediane verwenden und statt Euklidischen Distanzen, Manhattan Distanzen, implementiert z.B. in `cluster::pam`
- Ausreisser werden ebenfalls zu Zentren zugeordnet.
  - Ausweg: trimmed kmeans (u.a. in R-Paketen `tclust` oder `trimcluster`)
- kmeans/kmedians vergibt nur 0/1 Memberships (im jeweiligen Cluster oder nicht).
  - Ausweg: Fuzzy-Clustering (oder Model-based Clustering)
- Basiert ausschliesslich auf Distanzen, somit entdeckt man eher (nur) spährische Strukturen!
  - Ausweg: Model-based Clustering

Ihre Mitarbeit ist gefragt

`all_exercises-cluster_nolsg.pdf`

Clustering

- Aufgabe 4: *k*means Clustering

Model-based Clustering

- Theorie zu model-based Clustering ist sehr ausufernd und in der Literatur ausführlich beschrieben. Wir werden uns nur auf praktische Aspekte konzentrieren.
- Ein statistisches Modell wird für die Beschreibung der Form der Cluster verwendet.
- Standardform: Multivariate Normalverteilung, d.h. Annahme: Verteilung eines Clusters j hat Dichte einer multivariaten Normalverteilung, wir kennen aber μ_j und σ_j nicht.

Gegeben n_c Cluster aus multiv. Normaldichten mit Erwartungsw. $\boldsymbol{\mu}_j$ und Kovarianzen $\boldsymbol{\Sigma}_j$ (für $j = 1, \ldots, n_c$)

- Die Clustergrösse in Anteilen werden als Mischungsfaktor (*mixing coefficients*) $\pi_1, \ldots, \pi_{n_c}$ bezeichnet, wobei $\pi_1 + \ldots + \pi_{n_c} = 1$.
- All diese Parameter (μ_j, σ_j, π_j) werden mit dem EM-Algorithmus geschätzt.

Model-based Clustering

Im Falle von p Variablen ist die Kovarianzmatrix jedes einzelnen Clusters von Dimension $p \times p$. Falls p gross (aber auch n_c) ist, müssen sehr viele Parameter geschätzt werden, was zu Instabilitäten führen kann. Deshalb werden oft die Clustermodelle durch Restriktionen vereinfacht.

Die einfachste Restriktion ist

- $\mathbf{\Sigma}_j = \sigma^2 \mathbf{I}$, für $j = 1, \ldots, n_c$, wobei $\mathbf{I}$ die Einheitsmatrix ist und σ^2 der Parameter der Varianz.
- $\rightarrow$ alle Cluster sind spährisch mit bestimmten Radii. Darum braucht nur mehr σ^2 geschätzt werden.

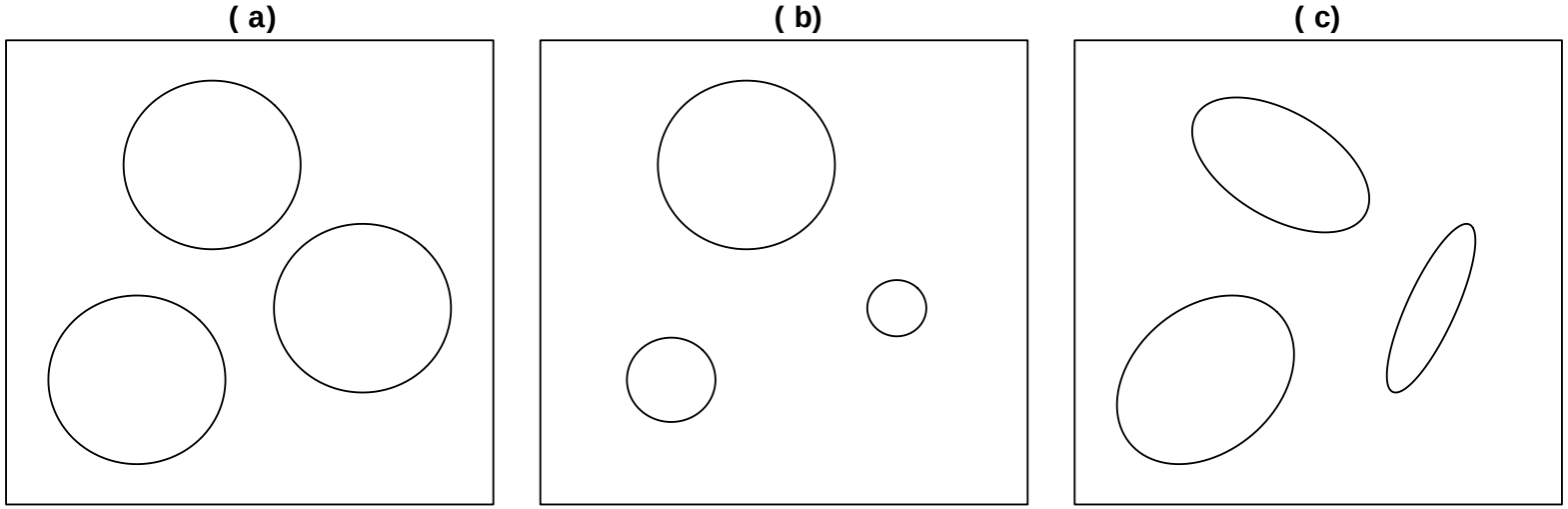
Eine weniger restriktierte Kovarianzstruktur ist

- $\mathbf{\Sigma}_j = \sigma_j^2 \mathbf{I}$, für $j = 1, \ldots, n_c$.
- In diesem Fall: Cluster sind weiterhin spährischer Natur, aber deren Grösse variiert bzgl deren Varianzen σ_j^2, welche geschätzt werden müssen.

Verschiedene Kovarianzen

Die folgende Abbildung illustrat verschiedene Kovarianzstrukturen der drei Cluster

a. $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}_3 = \sigma^2 \mathbf{I}$;
b. $\mathbf{\Sigma}_j = \sigma_j^2 \mathbf{I}$, für $j = 1, 2, 3$;
c. alle $\mathbf{\Sigma}_j$ sind unterschiedlich und keine spezielle sphärische Struktur



Optimales Modell

Ein optimales Modell kann über den BIC (Bayes'sches Informationskriterium) ermittelt werden.

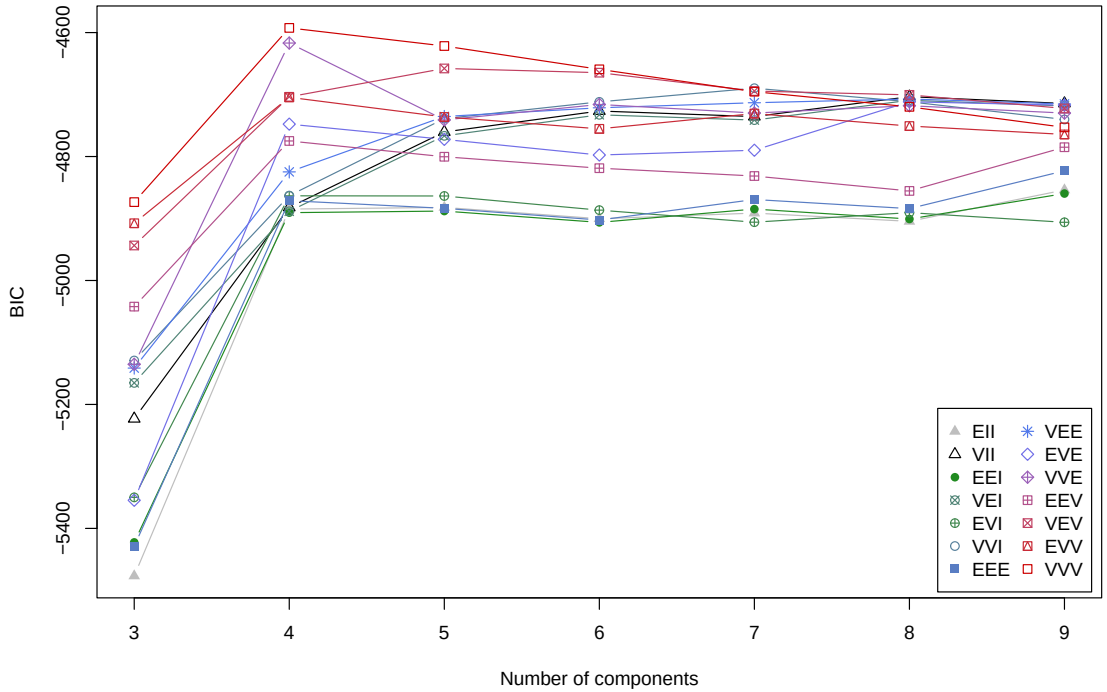
BIC:

- Theoretisch würde das zu sehr ausufern diesen genau zu erklären und herzuleiten
- $\approx$ 2 * log(durchschn. Likelihood der Cluster) minus Bestrafung von zuvielen Clustern.
  - es werden also recht kompakte Cluster gesucht
  - dies würde aber ohne Bestrafung zu kleine Cluster ergeben
  - Optimaler BIC: Balance zwischen Kompaktheit und nicht zu vielen Clustern
- Je grösser, desto besser.

Optimales Modell mit BIC in R

Standardmässig werden verschiedenste Clusterstrukturen gesucht, siehe `?mclust::mclustModelNames` (`mclust` muss natürlich einmalig installiert werden)

```
library("mclust"); data(Nclus, package = "flexclust")
# 3 to 9 mixture components:
res <- Mclust(Nclus, G = 3:9, verbose = FALSE)
plot(res, what = "BIC")
```



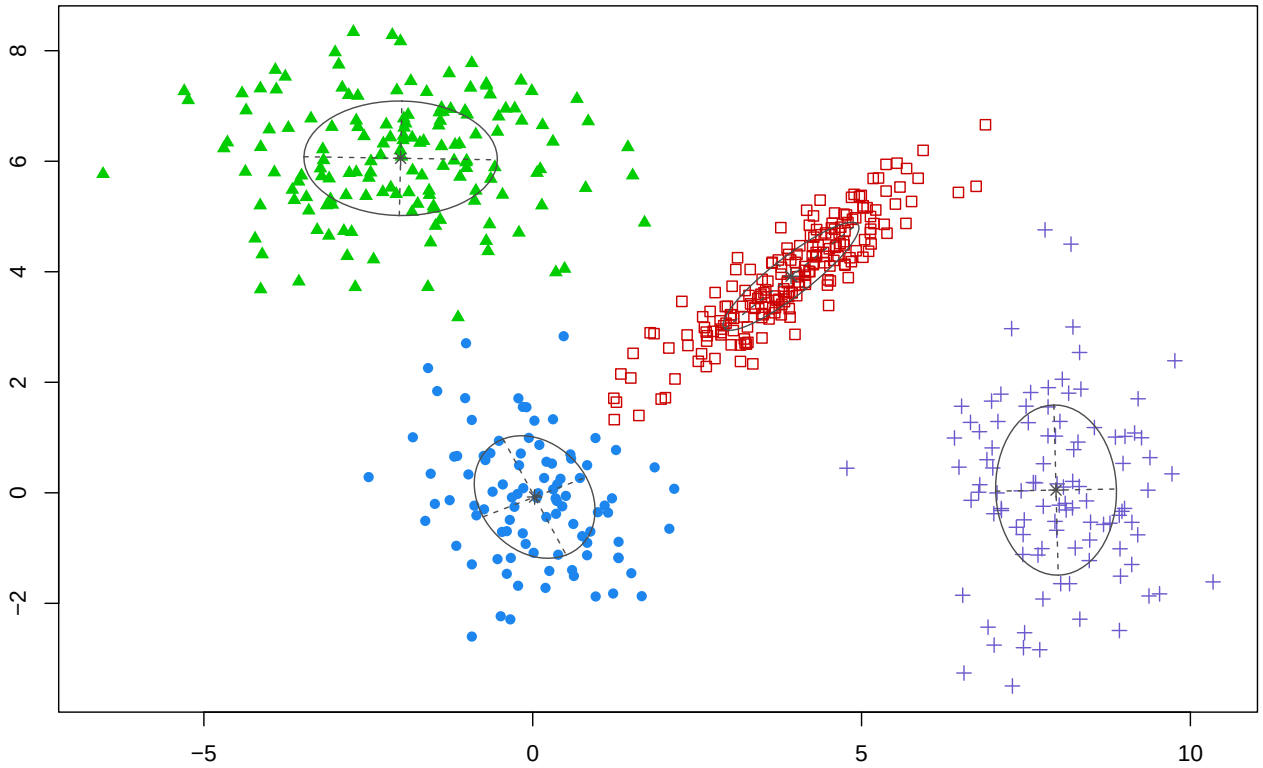
Optimales Modell

- Verschiedenste Kovarianzstrukturen und verschiedene Anzahl an Mixingkomponenten wurden getestet.
- Maximaler BIC indiziert optimales Modell.
- 4 Cluster scheinen optimal
- Das Modell VVV ist bestes Modell
- Erläuterungen in `?mclustModelNames` (Paket `mclust`)
- "VVV' ' heisst, dass die Kovarianzen der Cluster unterschiedlich sind (ellipsoidal, varying volume, shape, and orientation)

Optimales Modell

Ergebnis der Clustering inklusive Kovarianzstruktur der besten Lösung `VVV`:

```
plot(res, what = "classification")
```



Vergleich zu kmeans

- Der EM-Algorithmus muss bei model-based Clustering in jedem Schritt zusätzlich den Mixing Coefficient als auch die Kovarianz(en) schätzen. Bei kmeans nur die Clustermittel. Mclust ist also viel rechenintensiver.
- kmeans wies einige der Beobachtungen des *länglichen* Clusters augenscheinlich zu einem anderen Cluster zu. Mclust macht das besser.
- kmeans erkennt eher spährisch-symmetrische Cluster, Mclust ist hier flexibler.

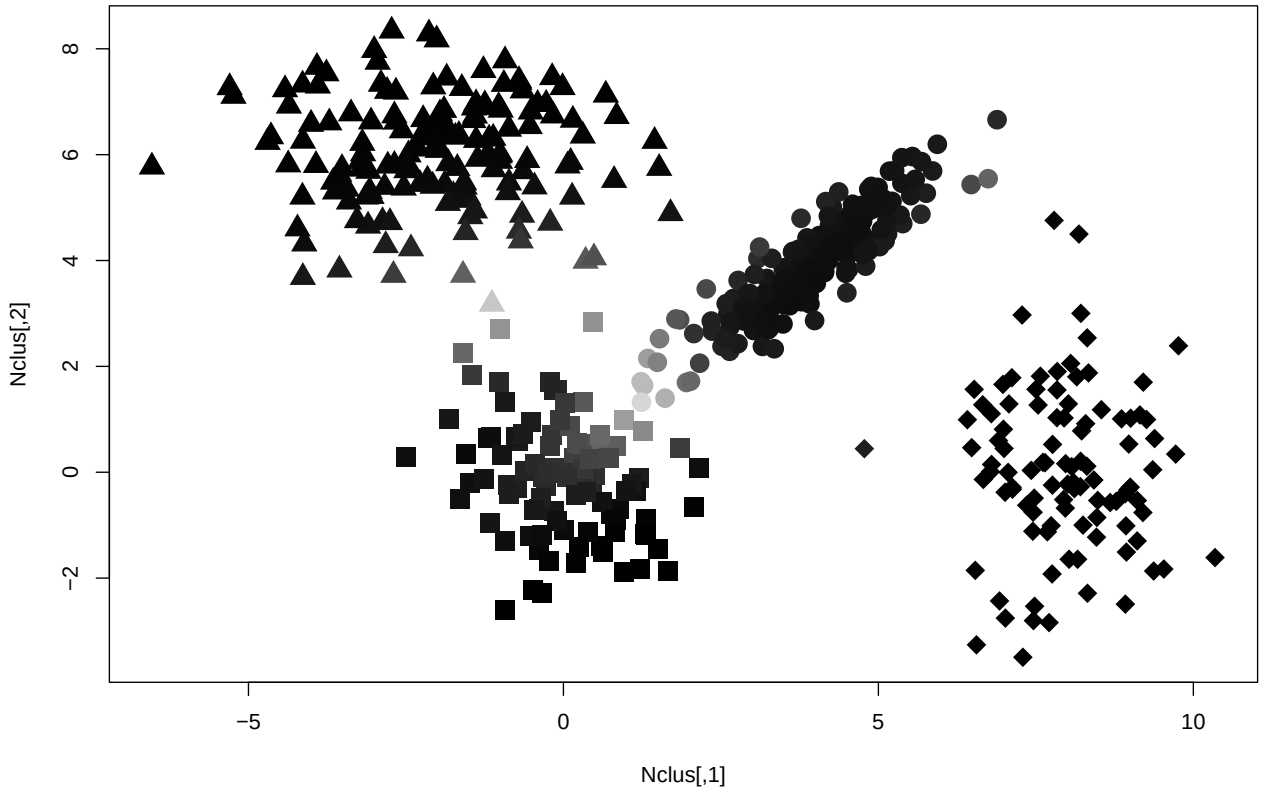
Auch wenn dies in der üblichen Praxis (anderswo) unbekannt scheint:

> *Falls man nicht sehr grosse Daten clustert ist auf jeden Fall das modellbasierte Clustern dem kmeans vorzuziehen.*

Kommentare Model-based Clustering, Unsicherheit

- Zusatzinformation: Zuordnung zu Clustern probabilistisch

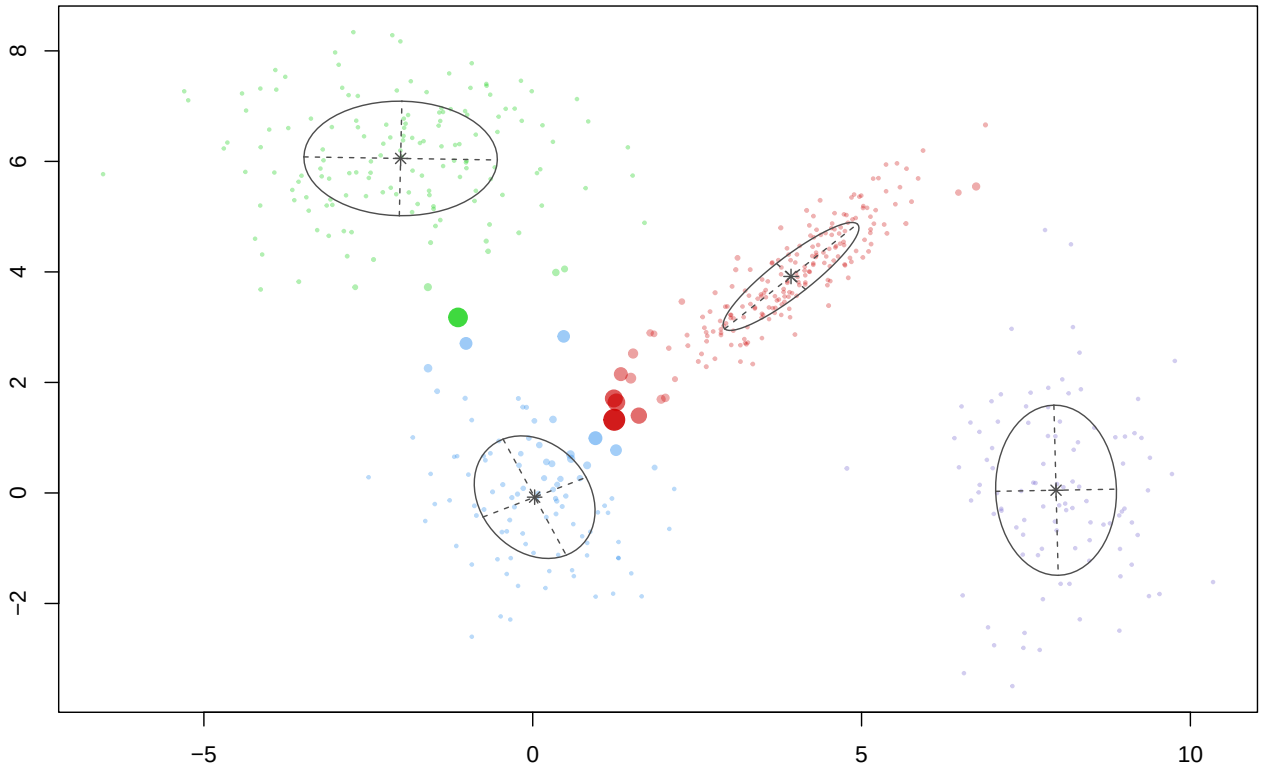
```
# see names(res)
plot(Nclus, col = grey((res$uncertainty)^(1/4)),
     pch = 14+res$classification, cex = 2)
```



Kommentare Model-based Clustering, Unsicherheit

Oder einfacher mit

```
plot(res, what = "uncertainty")
```



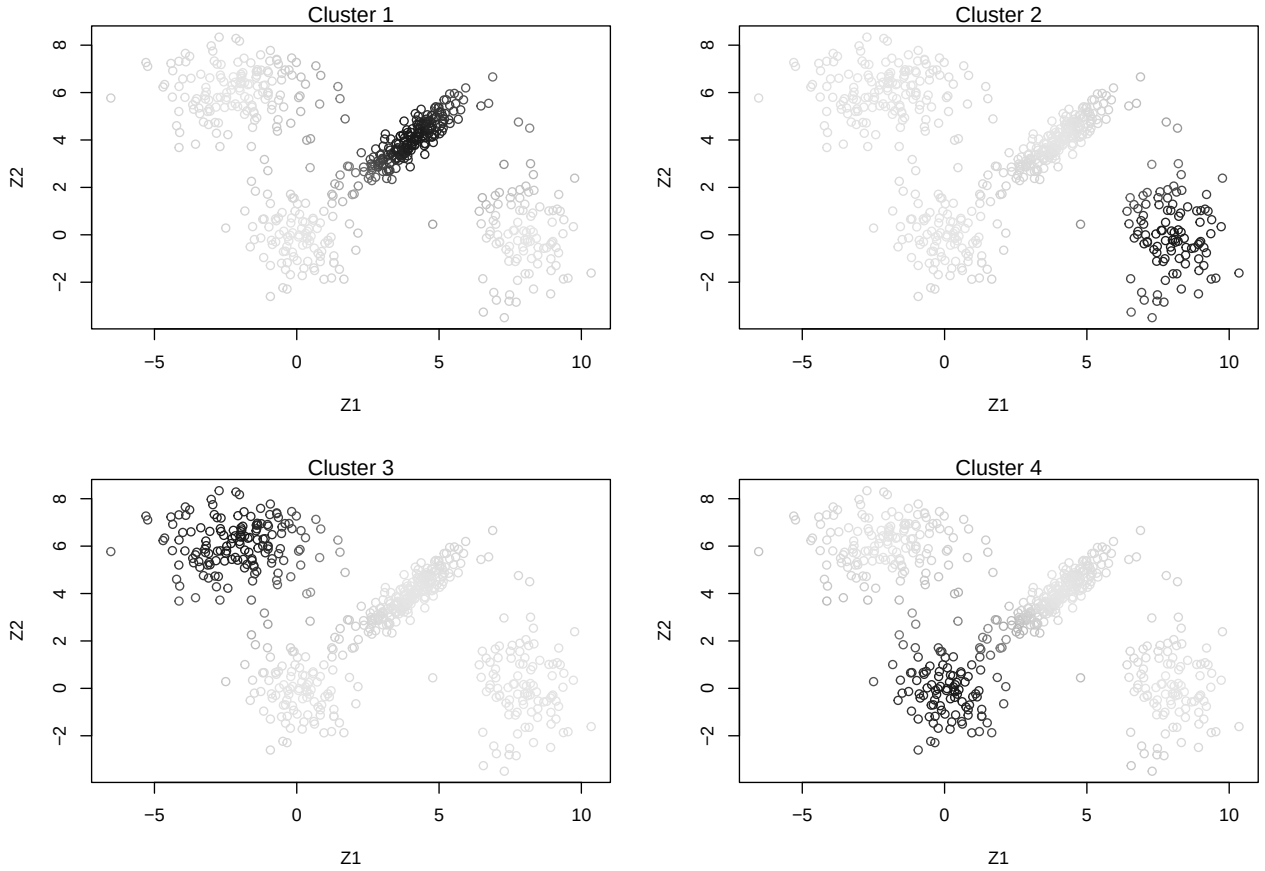
Fuzzy Clustering

Wir wollen hier auch nur die essentiellen Ideen wiedergeben und auf Theorie grossteils verzichten.

- Beobachtungen werden zu allen Clustern zugewiesen.
- Ein *membership coefficient* (Zugehörigkeitsgrad) u_{ik} drückt den Zugehörigkeitsgrad der Beobachtung i zum k-ten Cluster aus $(i = 1, \ldots, n;\ k = 1, \ldots, n_c)$, mit $u_{ik} \geq 0$ und $u_{i1} + \ldots + u_{in_c} = 1$, für alle i. Anmerkung: bei kmeans sind die u_{ik} nur 0 oder 1.
- Für eine fixe Anzahl von Clustern n_c wird die Lösung durch Minimierung einer Zielfunktion gefunden.
- Die Standardimplementierungen in R weisen numerische Probleme auf. Empfohlen ist daher die Funktion `cmeans` aus dem Paket e1017.
- Achtung: zufälliger Start des Algorithmus. Neuer Aufruf kann andere Ergebnisse liefern.

Fuzzy Clustering

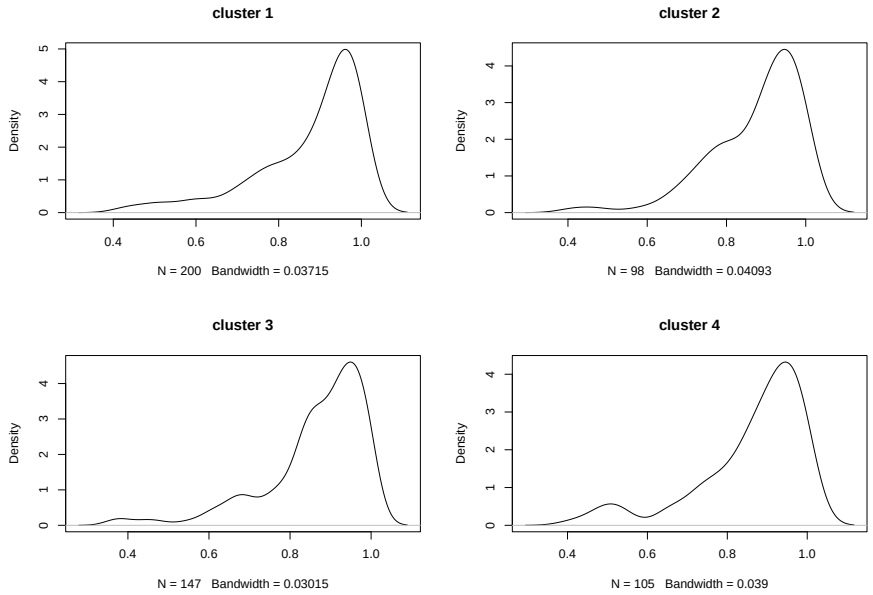
```
library("e1071"); groups <- 4
res <- cmeans(Nclus, groups)
for(i in seq_along(1:groups)){
  plot(Nclus, col = gray(1 - res$membership[, i]))}
```



Fuzzy Clustering

Wie sieht es aus mit der Güte der Cluster? Eine schnelle Möglichkeit: die Verteilung der Memberships zu visualisieren

```
par(mfrow = c(2,2))
for(i in 1:4){
  plot(density(res$membership[res$cluster == i,i]),
       main = paste("cluster", i))
}
```



Variablenclustern (Q-mode Clustering)

Statt die Beobachtungen zu clustern, clustert man die Variablen.

- entweder misst man wiederum Distanzen wie gehabt oder
- verwendet die Korrelation zwischen Variablen (gebräuchlicher)

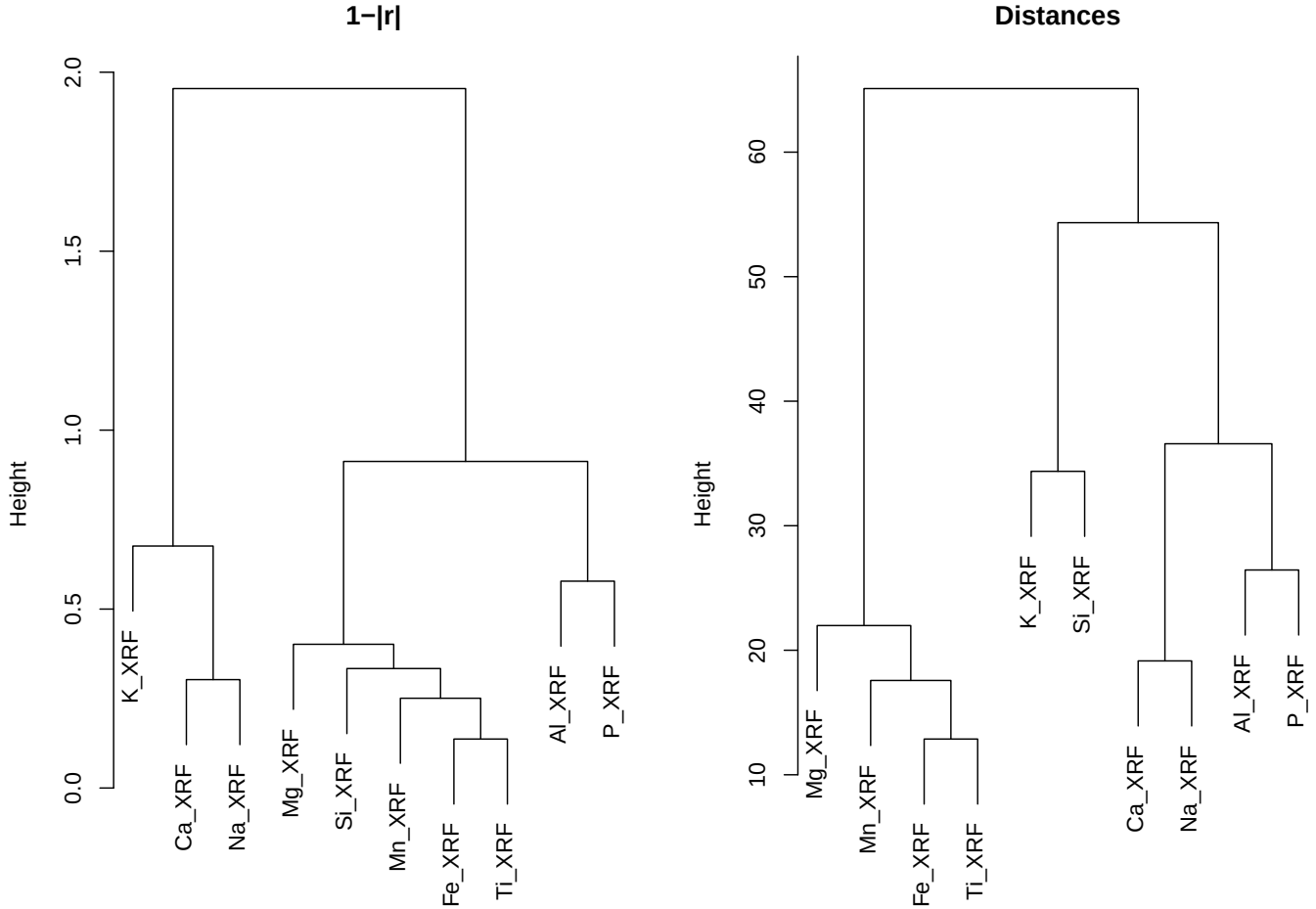
Variablenclustern (Q-mode Clustering)

Beispiel Kola-Daten (genaueres zu den Daten später in **Übung 04_cluster_xkola**)

```
library(mvoutlier)
data(chorizon)
## die (chemischen) Hauptelemente, log-scaled:
x <- scale(log(chorizon[, 101:110]))
# Q-mode Clustering (1- |Korrelationen|)
v.cor <- as.dist(1 - abs(cor(x)))
# ueber Distanzen:
v.d <- dist(t(as.matrix(x)))
```

Variablenclustern (Q-mode Clustering)

```
par(mfrow = c(1,2), mar = c(0,4,3,2))
plot(hclust(v.cor,method="ward.D"),xlab="",main="1-|r|")
plot(hclust(v.d,method="ward.D"),xlab="",main="Distances")
```



Ihre Mitarbeit ist gefragt

`all_exercises-cluster_nolsg.pdf`

Clustering

- Aufgabe 5: Fuzzy Clustering
- Aufgabe 6: Modelbasiertes Clustering
- Aufgabe 7: Weiteres Beispiel zu fuzzy- und modellbasiertem Clustern

Evaluierung der Cluster

- Wieviel Cluster sind optimal?
- Welche Clustermethode?
- Welche Transformation/Standardisierung der Daten?
- Welche Parametereinstellungen für eine Clustermethode?

Grundsätzlich sollte wohl gelten, dass ein Cluster möglichst homogen ist und die Cluster zueinander möglichst heterogen sind.

Unterschiedliche Arten von Gütekriterien

Wir unterscheiden grundsätzlich

- **interne** Clustervalidierungsmasse: man evaluiert das Clusterergebnis
- **externe** Clustervalidierungsmasse: man vergleicht ein Clusterresultat mit bekannter Gruppierung
- **relative** Clustervaldidierungsmasse: man vergleicht zwei Clusterresultate

Interne Clustervalidierungsmasse beinhalten eines oder mehrere dieser Kriterien:

1. Kompaktheit eines Clusters (Homogenität): wie nahe sind Beob. eines Clusters zueinander. Die *within-cluster* Masse.
2. Separiertheit (Heterogenität): wie separiert ist ein Cluster zu den anderen Clustern. Die *between-cluster* Masse.
3. Connectivity: Ist der nächste Nachbar einer Beobachtung im selben Cluster?

Evaluierung der Cluster. Heterogenität

Heterogenität kann man folgenderweise messen

$$B_{n_c} = \sum_{k=1}^{n_c} \|\mathbf{v}_k - \bar{\mathbf{v}}\|_2, \tag{2}$$

mit $\|\cdot\|$ der Euklidischen Norm, $\mathbf{v}_k$ das k-te Clusterzentrum ($k = 1, \ldots, n_c$), und

$$\bar{\mathbf{v}} = \frac{1}{n_c} \sum_{k=1}^{n_c} \mathbf{v}_k$$

das *Gesamtmittel* der Clusterzentren.

Dieses Mass ist als *between cluster sum of squares* bekannt.

Evaluierung der Cluster. Homogenität

Homogenität in Clustern ist definiert als

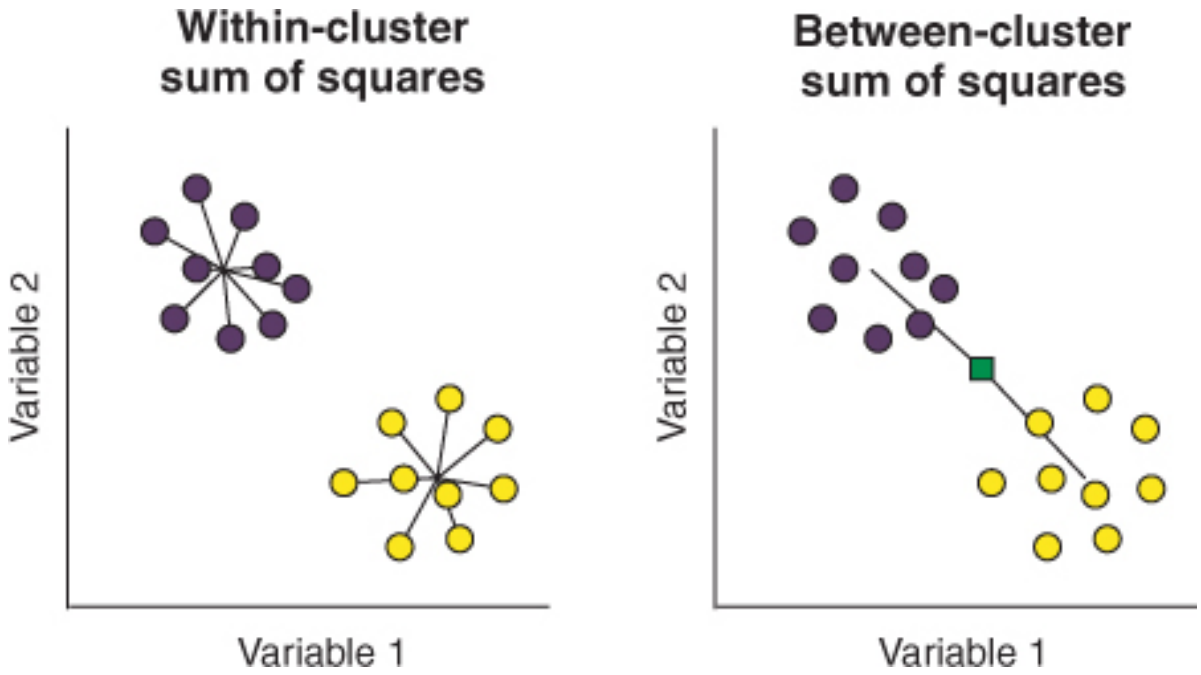
$$W_{n_c} = \sum_{k=1}^{n_c} \sum_{i \in C_k} \|\mathbf{x}_i - \mathbf{v}_k\|^2, \tag{3}$$

mit $\mathbf{z}_i$, $i = 1, \ldots, n$.

Dieses Mass ist als *within cluster sum of squares* bekannt, da es Euklidische Distanzen der Beobachtungen mit ihren jeweiligen Clusterzentren berücksichtigt.

- B_{n_c} sollte möglichst gross sein
- W_{n_c} sollte möglichst klein sein
- Beide hängen von der Anzahl der Cluster (n_c) ab, auf welche somit Rücksicht genommen werden sollte.

B und W visualisiert



(aus Machine Learning with R, the tidyverse, and mlr (Ryhs, 2020))

Evaluierung der Cluster. Calinski-Harabasz Index

Calinski-Harabasz index: (Optimum: Max-Wert)

$$\mathrm{CH}_{n_c} = \frac{B_{n_c}/(n_c - 1)}{W_{n_c}/(n - n_c)}$$

Hartigan index: (Optimum: *knee*)

$$\mathrm{H}_{n_c} = \ln \frac{B_{n_c}}{W_{n_c}}.$$

Praxis:

1. Verschiedene Anzahl von Clustern probieren und Clustermethode anwenden
2. Gütemasse ausrechnen
3. Dort wo das Gütemasse *am besten* ist: optimale Anzahl an Cluster.

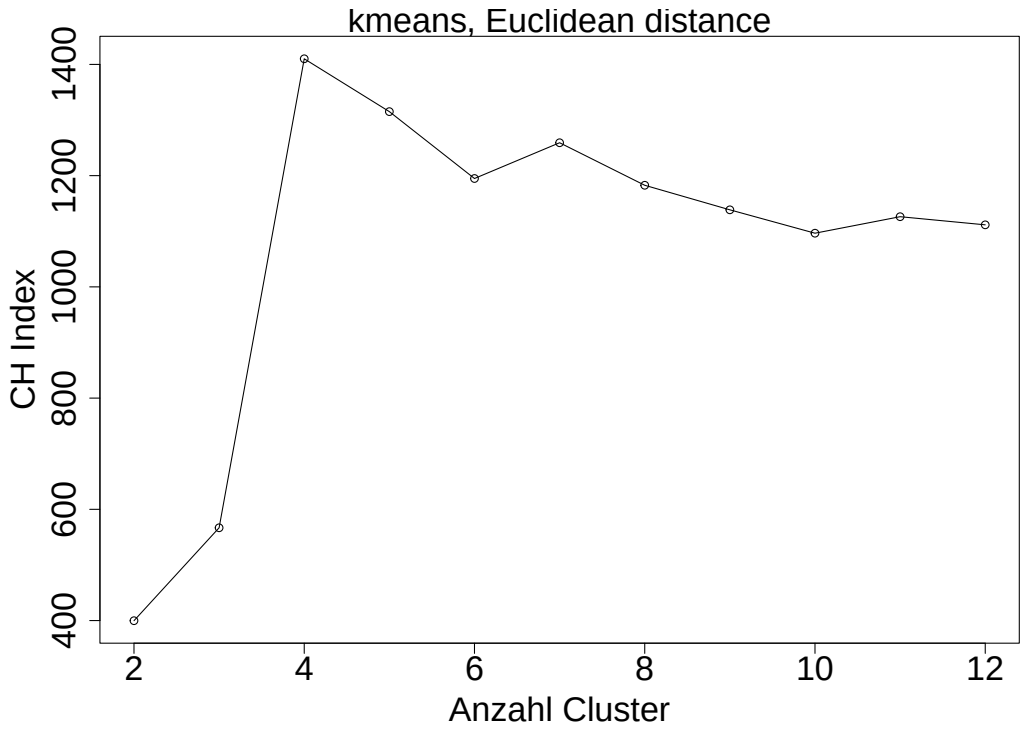
R Implementierungen Gütemasse

- Paket `NbClust` bietet (viele) Gütemasse, aber nur bzgl kmeans und einigen hierarchischen Clusterverfahren.
- Funktion `fpc::cluster.stats` hat ebenfalls (nicht ganz so viele) Gütemasse implementiert und es kann in Verbindung mit (fast) allen Clustermethoden angewandt werden.
- Paket `clValid` bietet einige Gütemasse zu vorgegebenen Clustermethoden.
- Paket `clusterSim`, `cclust`, `clv` bieten ebenfalls Gütekriterien an.

Beispiel Gütemasse Hartigan und Calinski-Harabasz

Optimum Anzahl Cluster ist 4 nach Calinski-Harabasz

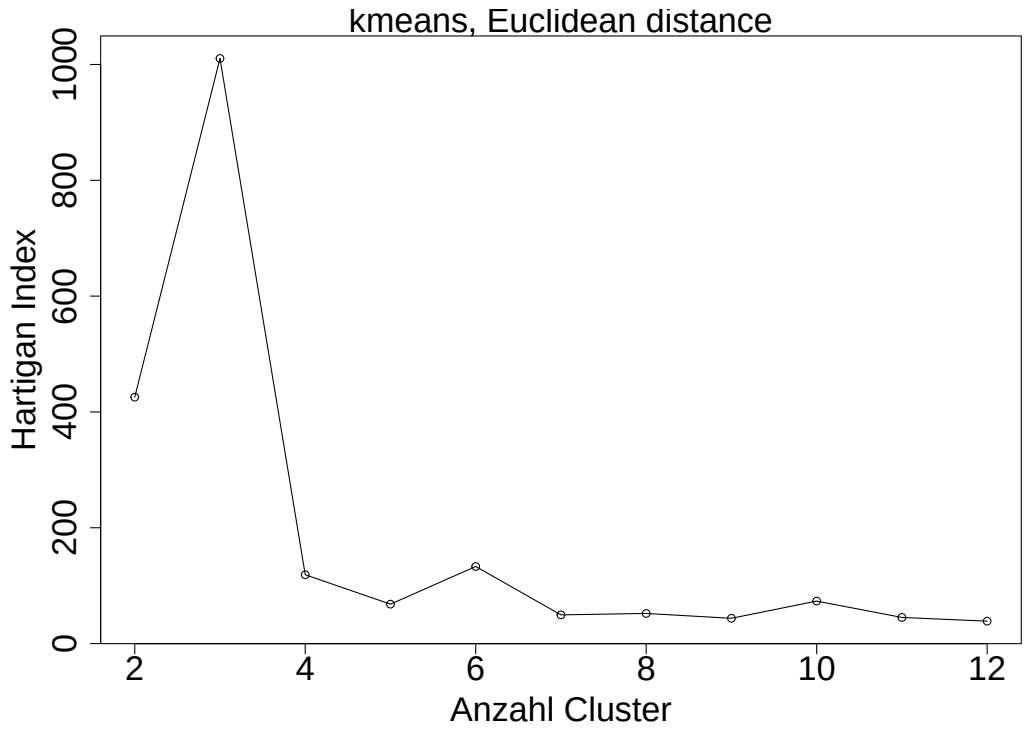
```
library(NbClust)
stats <- NbClust(data = Nclus, min.nc = 2, max.nc = 12,
                 method = "kmeans", index = "ch")
plot(2:12, stats$All.index, type = "o",
     xlab = "Anzahl Cluster", ylab = "CH Index")
```



Beispiel Gütemasse Hartigan und Calinski-Harabasz

Optimum Anzahl Cluster ist bei 4 nach Hartigan

```
library(NbClust)
stats <- NbClust(data = Nclus, min.nc = 2, max.nc = 12,
                 method = "kmeans", index = "hartigan")
plot(2:12, stats$All.index, type = "o",
     xlab = "Anzahl Cluster", ylab = "Hartigan Index")
```



Evaluierung der Cluster. Silhouette Widths

Average dissimilarity einer Beobachtung $\mathbf{x}_i$ zugehörig zum Cluster C_k zu allen anderen Beobachtungen **desselben** Clusters:

$$d_{i,C_k} = \frac{1}{n_{(k)} - 1} \sum_{i,j \in C_k, i \neq j} d^2(\mathbf{x}_i, \mathbf{x}_j),$$

wobei $n_{(k)}$ die Anzahl der Beobachtungen in Cluster C_k ist.

Average dissimilarity von $\mathbf{x}_i$ zu Beobachtungen eines **anderen** Clusters C_l:

$$d_{i,C_l} = \frac{1}{n_{(l)}} \sum_{j \in C_l} d^2(\mathbf{x}_i, \mathbf{x}_j).$$

Evaluierung der Cluster. Average Silhouette Width

Der Kleinste dieser Werte ist:

$$d_{i,C} = \min_l d_{i,C_l},$$

hat also kleinste Dissimilarity der i-ten Beobachtungen, nicht zu seinem eigenen, aber zu seinem **nähesten** der restlichen Cluster.

Der *silhouette value* ist definiert als:

$$s_i = \frac{d_{i,C} - d_{i,C_k}}{\max(d_{i,C_k}, d_{i,C})}.$$

- dieser ist normiert auf $[-1, 1]$
- Wenn s_i nahe 1: gut klassifiziert
- Wenn s_i nahe 0: Beobachtung liegt zwischen 2 Clustern
- Wenn s_i nahe -1: keine gute Klassifizierung der Beobachtung
- Negative s_i: Beobachtung ist womöglich im falschen Cluster zugeordnet

Evaluierung der Cluster. Average Silhouette Width

Die **Average Silhouette Width** ist:

$$\frac{1}{n} \sum_{i=1}^{n} s_i,$$

Je höher der Wert, desto besser die Clusterung.

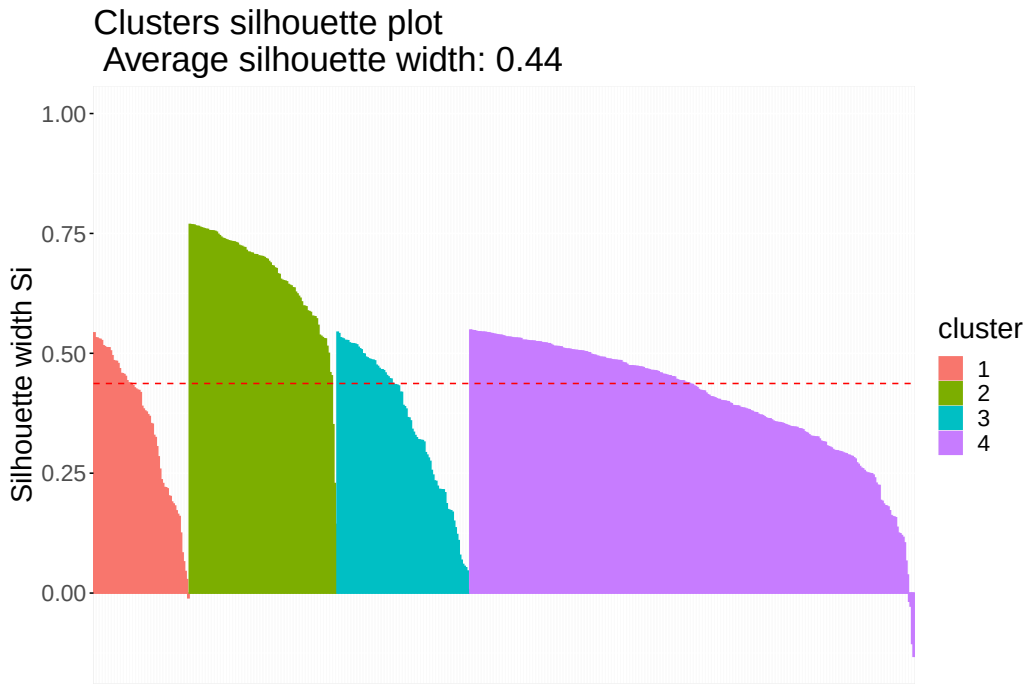
(sehr grobe) Faustregel:

- keine Clusterstruktur bei unter 0.25
- schwache Clusterstruktur zwischen 0.25 und 0.5
- gute Clusterstruktur zwischen 0.5 und 0.75
- sehr gute Clusterstruktur über 0.75

Beispiel zu Silhouette Widths

ZB mit Paket `factoextra` und `cluster::silhouette`. Gute Clusterung, bzgl. allen Clustern.

```
library(factoextra); library(cluster)
cl1 <- kmeans(Nclus, centers = 4)
sil <- silhouette(cl1$cluster, dist(Nclus))
fviz_silhouette(sil, print.summary = FALSE)
```



Kommentar Silhouette Widths

- Input (in `silhouette` bzw. Formel bevor) ist die Distanzmatrix und die Einteilung der Beobachtungen in Cluster.
- Es geht also um **Euklidische Distanzen** einer Beobachtung zu allen Beobachtunen innerhalb des selben Clusters und zu allen Beobachtungen andere Cluster.
- Das Mass ist somit nicht geeignet für modellbasiertes Clustern, falls hier ellipsenhafte Strukturen mittels unterschiedlicher Kovarianzen beurteilt werden.

Kommentare zu Gütemasse für die Praxis

Abseits geschönter (2-dim.) Lehrbeispiele (=Spielzeugdaten) ist es in der Praxis oft schwierig die optimale Anzahl von Clustern zu finden bzw. eine Clusterung zu bewerten.

- Daten mit Noise (das ist die Praxis):
  - globale Gütemasse sind sehr mit Vorsicht zu geniesen. Oft geben Sie zu wenige Cluster an
  - genauso sind hier relative Gütemasse betroffen
  - oft besser dann nur mehr mit lokalen Gütemasse einzelne Cluster bewerten
- Oft empfielt es sich visuell die Clusterergebnisse zu betrachten um zu bewerten, ob Cluster übersehen wurden ... Z.B. durch parallelen Koordinatenplot oder Grand Tours oder Guided Tours.

Deshalb werden wir uns ein komplexeres Beispiel ansehen, welches auch komplexer ist als von Ihnen (zur Prüfung) verlangt ($\rightarrow$ 04_cluster_kola.pdf)

Ihre Mitarbeit ist gefragt

`all_exercises-cluster_nolsg.pdf`

Clustering

- Aufgabe 8: Validity of clusters

Externe Gütemasse

Externe: Hier will man eine Clusterlösung mit einer bekannten Partition vergleichen. Anders formuliert, man hat Information über Gruppen und evaluiert, inwieweit diese Partition sich mit dem Clusterergebnis deckt. Nehmen wir an `iris[, 5]` enthält die **wahre Partition**.

```
set.seed(123)
cl <- kmeans(iris[, 1:4], 3, nstart = 10)
table(cl$cluster, iris$Species)
```

```
##
##      setosa versicolor virginica
##   1      50          0         0
##   2       0         48        14
##   3       0          2        36
```

Wir sehen einige Missclassifikationen (2 und 14).

Externe Gütemasse: Corrected Rand Index

- Corrected Rand Index misst eine Similarität zwischen 2 Partitionen.
- Die Korrektur ist relativ komplex, wir wollen sie hier nicht ausführen.
- Dieser Index ist zwischen -1 (keine Übereinstimmung) bis 1 (Clustereinteilung und Partition stimmen 100% überein)

```
library(fpc)
stats <- cluster.stats(dist(iris[, 1:4]),
  clustering = cl$cluster,
  alt.clustering = as.integer(iris$Species))
stats$corrected.rand
```

```
## [1] 0.7302383
```

Kommentar Clustering / Klassifikation

- Bei Methoden des Unsupervised Learnings (MDA, Clustering, Guided Tours, ...) gab es KEINE Zielvariable $\Rightarrow$ unsupervised learning, unsupervised wurden Gruppen gesucht.
- Bei Methoden der Klassifikation gibt es eine Zielvariable nach der klassifiziert wird $\Rightarrow$ supervised learning
- In der Praxis kann Clustering eine Vorstufe zur Klassifikation sein, falls die Zielvariable erst *gefunden* werden muss
  - Gruppen mittels Clusteranalyse finden
  - Auf diese Gruppen (=Zielvariable) ein Klassifikationsmethode trainierun um zukünftige Beobachtungen zu klassifizieren.

Summary (aller-)wichtigste Funktionen in R

- **dist** oder **factoextra::get_dist** für Distanzberechnung, **factoextra::fviz_dist** für Visualisierung der Distanzen
- **hclust**, **cutree** und **plot.hclust** für hierarchisches Clustering
- **kmeans**, **cluster::pam**, **cluster::clara** für k-means und k-medians Methoden.
- **e1071::cmeans** für Fuzzy-Clustering
- **mclust::Mclust** und **mclust::plot.Mclust** für modellbasiertes Clustern
- **NbClust::NbClust** iund **fpc::clusterstats** für Gütemasse
- **cluster::silhouette** und **factoextra::fviz_silhouette**