

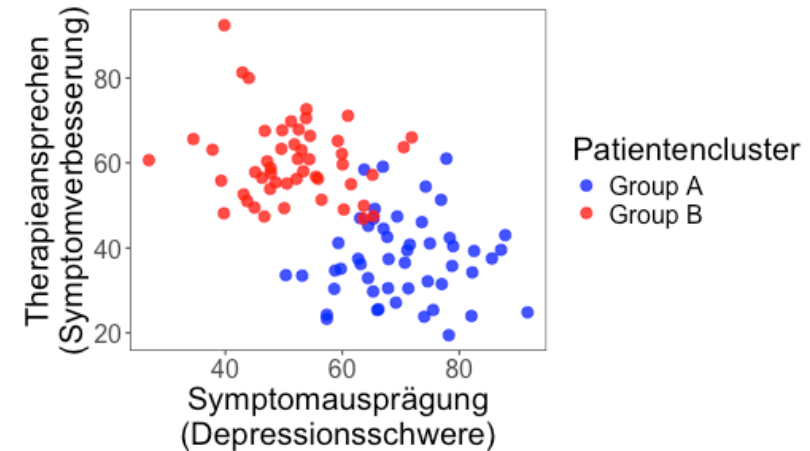
Multivariate Verfahren

Einheit 5: Gruppieren: Clusteranalyse

Wintersemester 2025 | Prof. Dr. Stephan Goerigk

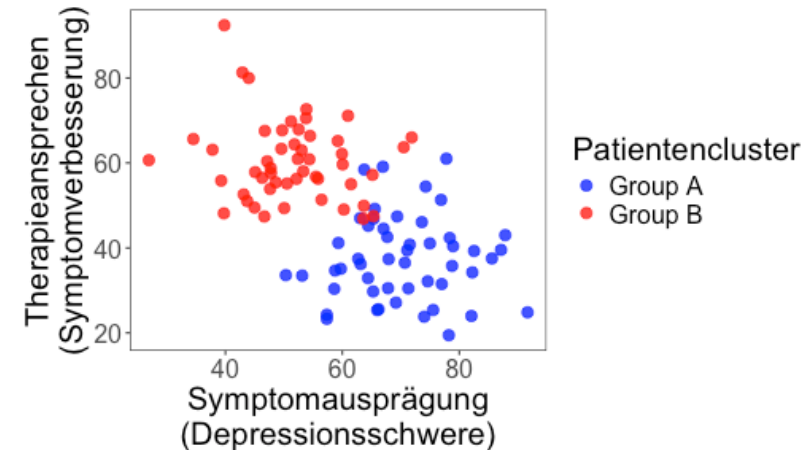
Einführung in die Clusteranalyse

- **Definition:** Clusteranalyse ist ein exploratives Verfahren, das verwendet wird, um **Objekte** (z. B. Personen, Symptome, Variablen) zu **Gruppen** (Cluster) zusammenzufassen, basierend auf deren Ähnlichkeit zueinander.
- **Ziel:** Finden von **natürlichen Gruppen** in den Daten, die ähnliche Merkmale oder Verhaltensmuster aufweisen (Mustererkennung).
- **Anwendungsbereiche in der Psychologie:**
 - **Kategorisierung** (z. B. Cluster von Symptomen).
 - **Identifikation von Risikogruppen** für psychische Störungen.
 - **Personalisierung** basierend auf Gruppenbildung (z. B. welche Therapie wirkt bei welcher Patientengruppe).



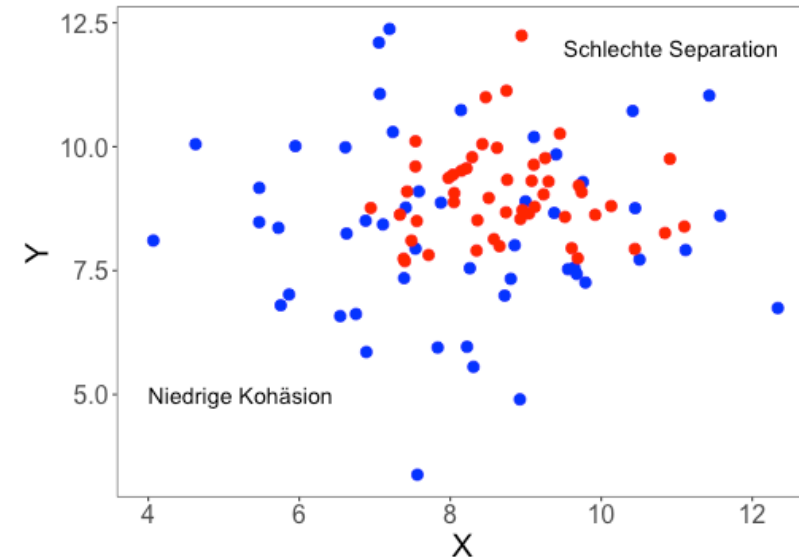
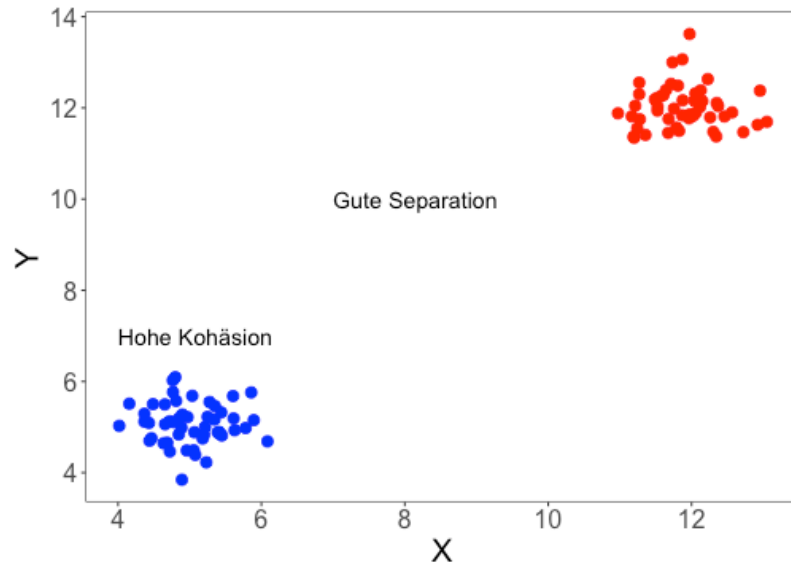
Grundkonzepte: Was ist Clustering?

- **Clustering** ist ein Verfahren der **strukturentdeckenden Verfahren**
- Datenpunkte in Gruppen zu unterteilen, **ohne** dass eine vorher festgelegte Zielvariable vorliegt.
- Im Machine Learning Kontext nennt man das auch "unsupervised Learning"
- Ähnliche Objekte sollen in einem Cluster zusammengefasst werden, während unähnliche Objekte unterschiedlichen Clustern zugeordnet werden.



Grundkonzepte: Wichtige Eigenschaften von Clustern

- **Kohäsion:** Objekte innerhalb eines Clusters sind einander ähnlich.
- **Separation:** Objekte in verschiedenen Clustern unterscheiden sich stark voneinander.

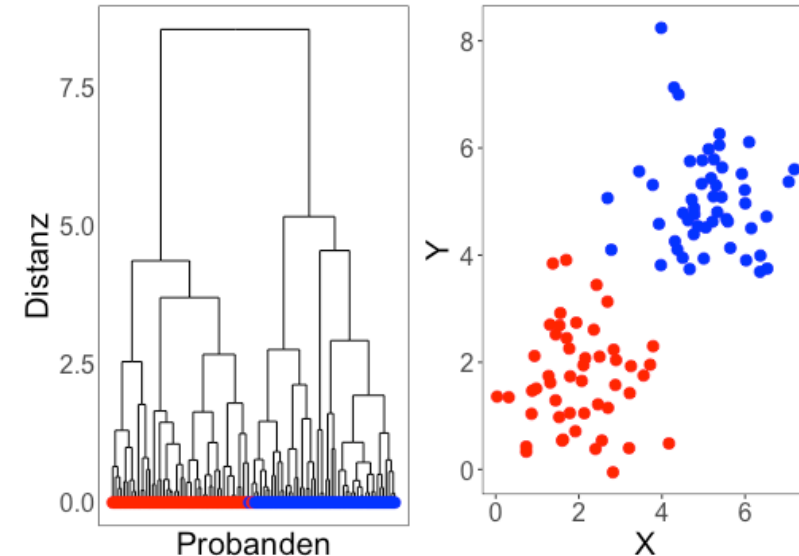


Arten der Clusteranalyse

- **Hierarchische Clusteranalyse:**
 - Organisiert Objekte in eine **hierarchische Struktur** von Clustern.
 - 2 Arten (**agglomerativ** vs. **divisiv**)
 - Ergebnis wird in einem **Dendrogramm** visualisiert, das die Clusterhierarchie zeigt.
- **Partitionierende Clusteranalyse (centroid-based):**
 - Teilt die Daten in eine **vorgegebene Anzahl von Clustern** ein.
 - Der bekannteste Ansatz ist der **K-Means-Algorithmus**:
 - Ziel: Minimierung der **Intra-Cluster-Varianz**.
- **Dichtebasierte Clusteranalyse:**
 - Gruppen von Objekten werden durch ihre **Dichte** in einem Raum definiert.
 - Findet Cluster von beliebiger Form.
 - Identifiziert auch **Ausreißer**, die nicht in Cluster passen.
- **Fuzzy Clustering:**
 - Fälle dürfen zu mehreren Clustern gehören (Zugehörigkeitsgrade)
 - z.B. Fuzzy C-Means (FCM) als beliebter Algorithmus

Hierarchisches Clustering

- Organisiert Objekte in eine **hierarchische Struktur** von Clustern.
 - 2 Arten (**agglomerativ** vs. **divisiv**)
 - Ergebnis wird in einem **Dendrogramm** visualisiert, das die Clusterhierarchie zeigt.
 - Zugehörigkeit von Personen auf Basis ihrer Werte in den "Input"-Variablen



Hierarchisches Clustering - agglomerativ vs. divisiv

Agglomeratives hierarchisches Clustering:

- Die agglomerative Clusteranalyse ist die häufigste Form der hierarchischen Clusteranalyse, die Objekte basierend auf ihrer Ähnlichkeit in Cluster gruppiert. Sie wird auch als AGNES (Agglomerative Nesting) bezeichnet.
- Die agglomerative Clusteranalyse arbeitet nach dem „bottom-up“-Prinzip:
- Jedes Objekt wird anfangs als eigenständiger Cluster (Blatt) betrachtet.
- In jedem Schritt des Algorithmus werden die beiden ähnlichsten Cluster zu einem größeren Cluster (Knoten) zusammengeführt.
- Dieser Vorgang wird wiederholt, bis alle Punkte Teil eines einzigen großen Clusters (Wurzel) sind.
- Das Ergebnis ist eine baumbasierte Darstellung der Objekte, die als Dendrogramm bezeichnet wird.

Hierarchisches Clustering - agglomerativ vs. divisiv

Divisives hierarchisches Clustering:

- Die divisive Clusteranalyse ist das Gegenteil der agglomerativen Clusteranalyse und wird auch als DIANA (Divisive Analysis) bezeichnet.
- Sie arbeitet nach dem „top-down“-Prinzip:
- Der Prozess beginnt bei der Wurzel, wobei alle Objekte in einem einzigen Cluster enthalten sind.
- In jedem Iterationsschritt wird der heterogenste Cluster in zwei Teilcluster aufgeteilt.
- Dieser Vorgang wird wiederholt, bis jedes Objekt seinen eigenen Cluster bildet.
- Während die agglomerative Clusteranalyse gut darin ist, kleine Cluster zu identifizieren, eignet sich die divisive Clusteranalyse typischerweise besser zur Identifikation von großen Clustern.

Hierarchisches Clustering - Linkage

- Ein entscheidender Faktor bei der hierarchischen Clusteranalyse ist die Messung der Unähnlichkeit zwischen zwei Clustern von Beobachtungen.
- Es wurden verschiedene Methoden zur Cluster-Aggregation (sogenannte Linkage-Methoden) entwickelt, um diese Frage zu beantworten. Die gängigsten Methoden sind:
 - Maximum- oder Complete-Linkage-Clustering
 - Minimum- oder Single-Linkage-Clustering
 - Mean- oder Average-Linkage-Clustering
 - Ward's Minimum-Variance-Methode

Hierarchisches Clustering - Linkage

- Maximum- oder Complete-Linkage-Clustering:
 - Berechnet alle paarweisen Unähnlichkeiten zwischen den Elementen von Cluster 1 und Cluster 2.
 - Nimmt den größten Wert (d. h. den Maximalwert) dieser Unähnlichkeiten als Abstand zwischen den Clustern.
 - Führt dazu, dass kompaktere Cluster entstehen.
- Minimum- oder Single-Linkage-Clustering:
 - Berechnet alle paarweisen Unähnlichkeiten zwischen den Elementen von Cluster 1 und Cluster 2.
 - Nimmt den kleinsten dieser Werte als Kriterium für die Verknüpfung.
 - Führt häufig zu langen, „losen“ Clustern.
- Mean- oder Average-Linkage-Clustering:
 - Berechnet alle paarweisen Unähnlichkeiten zwischen den Elementen von Cluster 1 und Cluster 2.
 - Nimmt den Durchschnitt dieser Unähnlichkeiten als Abstand zwischen den Clustern.
- Ward's Minimum-Variance-Methode:
 - Minimiert die totale Varianz innerhalb der Cluster.
 - In jedem Schritt werden die Cluster mit der minimalen zwischen-Cluster-Distanz zusammengeführt.

Agglomeratives hierarchisches Clustering

Im Allgemeinen funktioniert die agglomerative hierarchische Clusteranalyse wie folgt:

1. Berechnung der (Un-)Ähnlichkeit ([Dis]similarities) zwischen jedem Paar von Objekten im Datensatz.
2. Verwendung einer Verknüpfungsfunktion (Linkage-Funktion)
 - Objekte werden basierend auf Basis der Ähnlichkeiten zu einer hierarchischen Clusterstruktur zusammengefasst.
 - Objekte/Cluster, die nahe beieinander liegen, werden zusammengeführt/verknüpft.
3. Pruning: Bestimmung, an welcher Stelle der hierarchische Baum (Dendrogram) in Cluster aufgeteilt werden soll

Agglomeratives hierarchisches Clustering

Beispiel: Identifizierung von Depressionssubtypen anhand von Entzündungsmarkern und neuronaler Aktivität

- Depression ist eine hochgradig heterogene Erkrankung, bei der Patienten unterschiedliche biologische Grundlagen aufweisen.
- Durch das Clustern von Patienten basierend auf Biomarkern können biologisch unterschiedliche Subtypen identifiziert werden
- Subtypen → maßgeschneiderte Behandlungen

Agglomeratives hierarchisches Clustering

Beispiel: Identifizierung von Depressionssubtypen anhand von Entzündungsmarkern und neuronaler Aktivität

Dimensionen für das Clustering:

1. Entzündungsmarker:

- Beispiele: C-reaktives Protein (CRP), Interleukin-6 (IL-6) oder Tumor-Nekrose-Faktor-Alpha (TNF- α).
- Höhere Werte können auf systemische Entzündungen hinweisen, die mit einer spezifischen Depressionsform verbunden sind, die häufig resistent gegenüber Standardbehandlungen ist.

2. Neuronale Aktivität:

- Beispiel: Funktionale Aktivität im präfrontalen Kortex, gemessen mit fMRT oder EEG.
- Reduzierte Aktivität im linken dorsolateralen präfrontalen Kortex (DLPFC) wird häufig bei Patienten mit therapieresistenter Depression beobachtet.

Agglomeratives hierarchisches Clustering

Beispiel: Identifizierung von Depressionssubtypen anhand von Entzündungsmarkern und neuronaler Aktivität

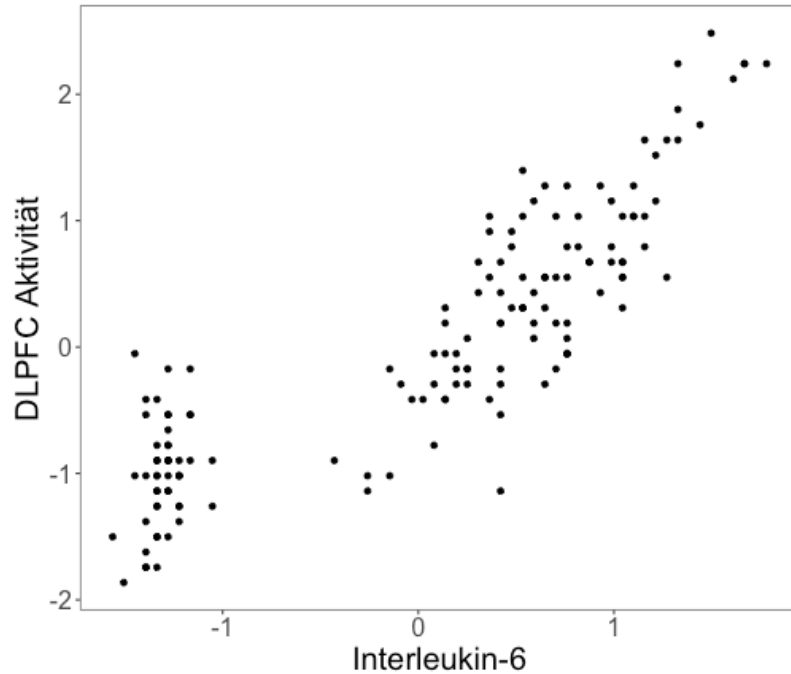
Dimensionen für das Clustering:

1. Entzündungsmarker: Interleukin-6 (IL-6)
2. Neuronale Aktivität: DLPFC Aktivität

Ausschnitt von $N = 15$ Personen aus Datensatz (Werte sind z-standardisiert)

##	DLPFC Aktivität	Interleukin-6
## 1	-0.89767388	-1.335752
## 2	-1.13920048	-1.335752
## 3	-1.38072709	-1.392399
## 4	-1.50149039	-1.279104
## 5	-1.01843718	-1.335752
## 6	-0.53538397	-1.165809
## 7	-1.50149039	-1.335752
## 8	-1.01843718	-1.279104
## 9	-1.74301699	-1.335752
## 10	-1.13920048	-1.279104
## 11	-0.53538397	-1.279104
## 12	-1.25996379	-1.222456
## 13	-1.25996379	-1.335752
## 14	-1.86378030	-1.505695
## 15	-0.05233076	-1.449047

Agglomeratives hierarchisches Clustering



```
ggplot(df, aes(y = `DLPFC Aktivität`,  
               x = `Interleukin-6`)) +  
  geom_point()
```

- Visualisierung im Streudiagramm
- Leicht möglich, da Daten 2-dimensional
- Jeder Punkt entspricht einer Person

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities)

- Um zu entscheiden, welche Objekte/Cluster kombiniert werden sollen, müssen Methoden zur Messung der Ähnlichkeit angegeben werden.
- In R berechnen wir Ähnlichkeit über die Funktion `dist()`
- Standardmäßig berechnet die Funktion `dist()` die **euklidische Distanz** zwischen Objekten.
- Es ist jedoch möglich, andere Distanzmetriken anzugeben.

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities)

Euklidische Distanz:

- Die euklidische Distanz ist die geradlinige Entfernung zwischen zwei Punkten (Personen) in einem mehrdimensionalen Raum.
- Für zwei Punkte $A(x_1, y_1)$ und $B(x_2, y_2)$ im zweidimensionalen Raum:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Für n-dimensionale Punkte:

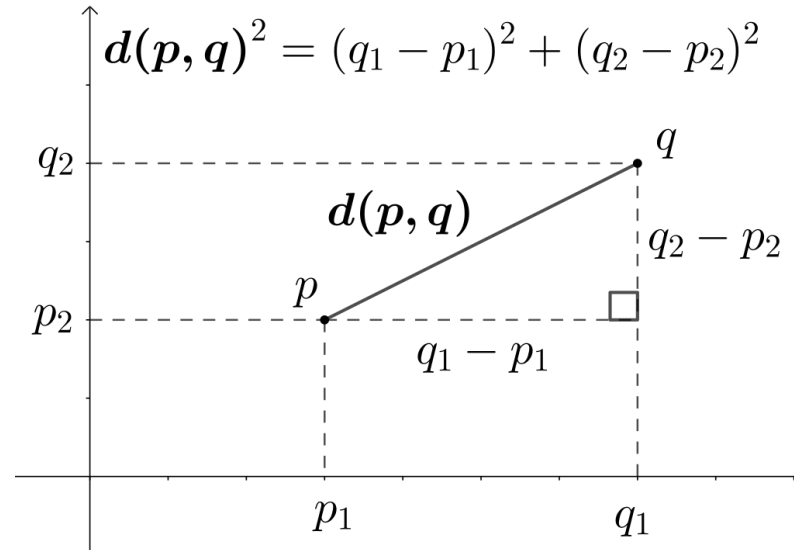
$$d(A, B) = \sqrt{\sum_{i=1}^n (x_{iB} - x_{iA})^2}$$

- Eigenschaften der euklidische Distanz
- Positive Werte: Die Distanz ist immer positiv (oder 0, wenn beide Punkte identisch sind).
- Symmetrie: $d(A, B) = d(B, A)$.
- Dreiecksungleichung: Der direkte Weg zwischen zwei Punkten ist immer kürzer oder gleich der Summe der indirekten Wege.

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities)

Euklidische Distanz:



Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities)

Exkurs: Alternative Distanzmetriken:

Manhattan-Distanz (City-Block-Distanz)

- Definition: Summiert die absoluten Differenzen der Koordinaten zwischen zwei Punkten.
- Formel:

$$d(A, B) = \sum_{i=1}^n |x_{iA} - x_{iB}|$$

- Beispiel: Geeignet für hochdimensionale Daten und bei Daten, die wie in einem Raster (z. B. Stadtstraßen) strukturiert sind.
- Eigenschaft: Führt oft zu „längeren“ Clustern als die euklidische Distanz.

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities)

Exkurs: Alternative Distanzmetriken:

Minkowski-Distanz

- Definition: Verallgemeinerung der euklidischen und Manhattan-Distanzen.
- Formel:

$$d(A, B) = \left(\sum_{i=1}^n |x_{iA} - x_{iB}|^p \right)^{\frac{1}{p}}$$

- Parameter p:
- p = 1: Manhattan-Distanz
- p = 2: Euklidische Distanz
- Beispiel: Ermöglicht Flexibilität in der Definition von Distanzen.

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities)

Exkurs: Alternative Distanzmetriken:

Cosinus-Distanz (Cosine Similarity)

- Definition: Misst den Winkel zwischen zwei Vektoren, nicht deren absolute Werte.
- Formel (Ähnlichkeit):

$$\text{Cosine Similarity} = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$$

Cosinus-Distanz = 1 - Cosine Similarity.

- Beispiel: Häufig in Textanalysen verwendet, z. B. bei der Analyse von Wortfrequenzen (Bag-of-Words).
- Eigenschaft: Unempfindlich gegenüber der Länge der Vektoren.

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities)

Exkurs: Alternative Distanzmetriken:

Mahalanobis-Distanz

- Definition: Berücksichtigt die Korrelationen zwischen den Variablen und die Verteilung der Daten.
- Formel:

$$d(A, B) = \sqrt{(x_A - x_B)^T \Sigma^{-1} (x_A - x_B)}$$

wobei Σ^{-1} die inverse Kovarianzmatrix ist.

- Beispiel: Besonders nützlich bei korrigierten und skalierten Daten oder bei variierenden Skalen.
- Eigenschaft: Berücksichtigt Abhängigkeiten zwischen Variablen.

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities)

Exkurs: Alternative Distanzmetriken:

Jaccard-Distanz

- Definition: Misst die Ähnlichkeit zwischen zwei Mengen, insbesondere für binäre oder kategoriale Daten.
- Formel:

$$\text{Jaccard Similarity} = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard-Distanz = $1 - \text{Jaccard Similarity}$.

- Beispiel: Verwendung bei Clustering von Merkmalen wie „Ja/Nein“-Antworten.

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities) - Distanzmatrix

```
# Berechnung der (Un-)Ähnlichkeitsmatrix
```

```
d <- dist(df, method = "euclidean")
```

```
as.matrix(d)[1:8, 1:8] # Anzeigen für die ersten 8 Personen
```

##		1	2	3	4	5	6	7	8
## 1	0.0000000	0.2415266	0.4863634	0.60646792	0.12076330	0.4001682	0.60381651	0.13338940	
## 2	0.2415266	0.0000000	0.2480807	0.36669188	0.12076330	0.6272758	0.36228991	0.13338940	
## 3	0.4863634	0.2480807	0.0000000	0.16558865	0.36669188	0.8751847	0.13338940	0.37959163	
## 4	0.6064679	0.3666919	0.1655887	0.00000000	0.48636340	0.9727268	0.05664765	0.48305321	
## 5	0.1207633	0.1207633	0.3666919	0.48636340	0.00000000	0.5120752	0.48305321	0.05664765	
## 6	0.4001682	0.6272758	0.8751847	0.97272680	0.51207520	0.00000000	0.98093946	0.49616149	
## 7	0.6038165	0.3622899	0.1333894	0.05664765	0.48305321	0.9809395	0.00000000	0.48636340	
## 8	0.1333894	0.1333894	0.3795916	0.48305321	0.05664765	0.4961615	0.48636340	0.00000000	

Agglomeratives hierarchisches Clustering

Berechnung der (Un-)Ähnlichkeit ([Dis]similarities) - Distanzmatrix

##		1	2	3	4	5	6	7	8
## 1	0.0000000	0.2415266	0.4863634	0.60646792	0.12076330	0.4001682	0.60381651	0.13338940	
## 2	0.2415266	0.0000000	0.2480807	0.36669188	0.12076330	0.6272758	0.36228991	0.13338940	
## 3	0.4863634	0.2480807	0.0000000	0.16558865	0.36669188	0.8751847	0.13338940	0.37959163	
## 4	0.6064679	0.3666919	0.1655887	0.00000000	0.48636340	0.9727268	0.05664765	0.48305321	
## 5	0.1207633	0.1207633	0.3666919	0.48636340	0.00000000	0.5120752	0.48305321	0.05664765	
## 6	0.4001682	0.6272758	0.8751847	0.97272680	0.51207520	0.00000000	0.98093946	0.49616149	
## 7	0.6038165	0.3622899	0.1333894	0.05664765	0.48305321	0.9809395	0.00000000	0.48636340	
## 8	0.1333894	0.1333894	0.3795916	0.48305321	0.05664765	0.4961615	0.48636340	0.00000000	

- Große Werte = hohe Unähnlichkeit/Distanz zwischen Personen
- Kleine Werte = hohe Ähnlichkeit zwischen Personen (Kandidaten für gleiches Cluster)
- Diagonale 0 Werte = Distanz einer Person mit sich selbst (Distanz = 0 da Werte identisch)

Agglomeratives hierarchisches Clustering

Linkage in R

- Das hierarchische Clustering wird mit der `hclust()` Funktion durchgeführt
- Typischerweise werden hierfür die Complete-Linkage-Methode oder die Ward-Methode bevorzugt.

```
# Linkage berechnen (Distanzmatrix geben)
hc <- hclust(d = d, method = "ward.D2")
```

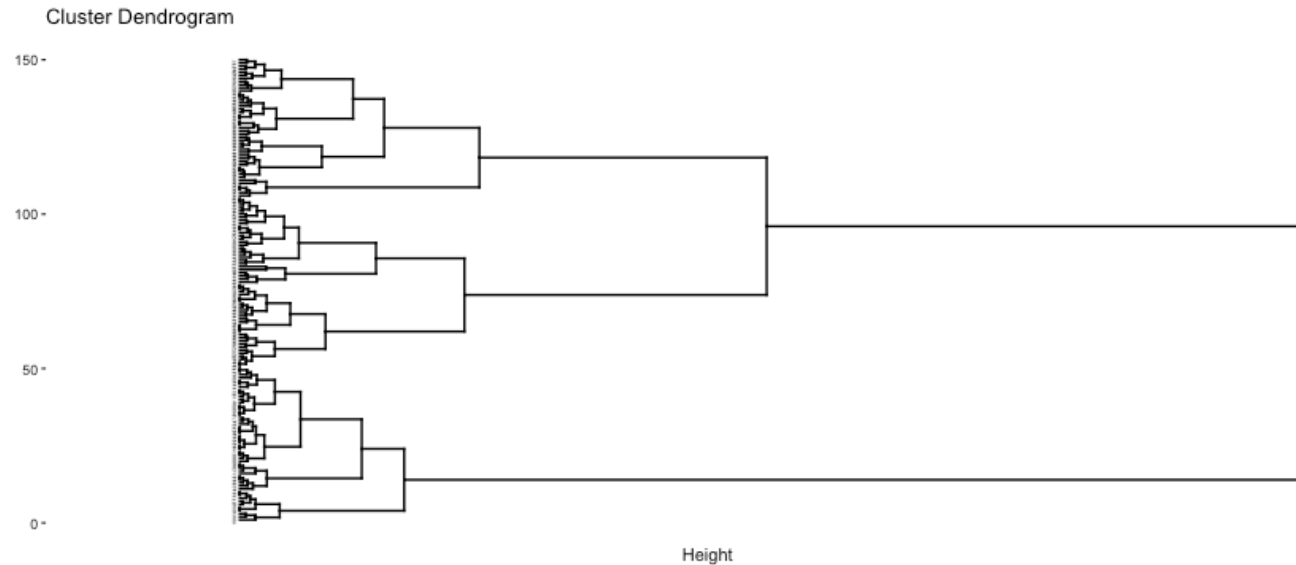
```
hc
```

```
##
## Call:
## hclust(d = d, method = "ward.D2")
##
## Cluster method      : ward.D2
## Distance            : euclidean
## Number of objects: 150
```

Agglomeratives hierarchisches Clustering

Dendrogram visualisieren

```
fviz_dend(hc, cex = 0.2) + coord_flip()
```



Agglomeratives hierarchisches Clustering

Dendrogram visualisieren

- In den Dendrogrammen entspricht jedes Blatt einem Datenpunkt oder Objekt.
- Objekte, die einander ähnlich sind, werden zu Ästen kombiniert, bis schließlich alles in einem einzigen Cluster zusammengeführt wird.
- Die Höhe der Verschmelzung auf der vertikalen Achse zeigt die (Un)Ähnlichkeit bzw. Distanz zwischen zwei Objekten oder Clustern an.
- Je höher die Höhe der Verschmelzung, desto weniger ähnlich sind die Objekte.
- Diese Höhe wird als cophenetische Distanz zwischen den beiden Objekten bezeichnet.

Agglomeratives hierarchisches Clustering

Dendrogram - Modellpassung

- Wir können überprüfen, wie gut der Baum die tatsächlichen Daten widerspiegelt, indem wir die Distanzmatrix betrachten, die mit der Funktion `dist()` berechnet wird.
- Diese Distanzmatrix dient im Wesentlichen als Möglichkeit, den Baum und die berechneten Distanzen zu validieren.
- Dies kann erreicht werden, indem die Korrelation zwischen den cophenetischen Distanzen und den ursprünglichen Distanzen berechnet wird.
- In der Theorie sollte diese Korrelation hoch sein. Idealerweise liegt die Korrelation über 0.75.
- Wenn die Korrelation nicht ausreichend ist → andere Linkage-Methoden ausprobieren

```
# cophenetischen Distanzen berechnen  
coph <- cophenetic(hc)
```

```
# Korrelation zwischen den cophenetischen Distanzen und den ursprünglichen Distanzen  
cor(d, coph)
```

```
## [1] 0.774994
```

Agglomeratives hierarchisches Clustering

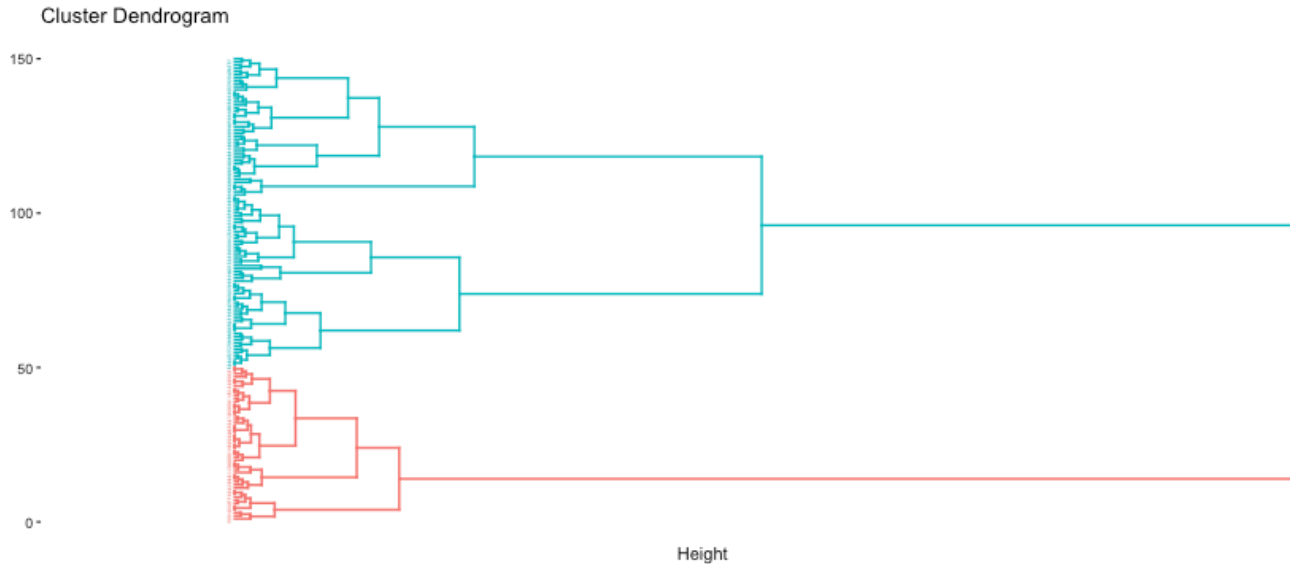
Zuweisung der Fälle zu den Clustern

- Als Nächstes können wir die Cluster tatsächlich benennen, da der Algorithmus zwar ein Dendrogramm erzeugt, wir dieses aber interpretieren müssen.
- Wir müssen die Anzahl der Cluster k bestimmen. In unserem Graph sah es so aus, als wäre $k = 2$.
- Es ist jedoch möglich, komplexere Methoden zu verwenden, um ein geeignetes k zu bestimmen, indem das Dendrogramm genauer betrachtet und analysiert wird (folgt später).
- Wir fahren mit $k = 2$ fort.

Agglomeratives hierarchisches Clustering

Zuweisung der Fälle zu den Clustern

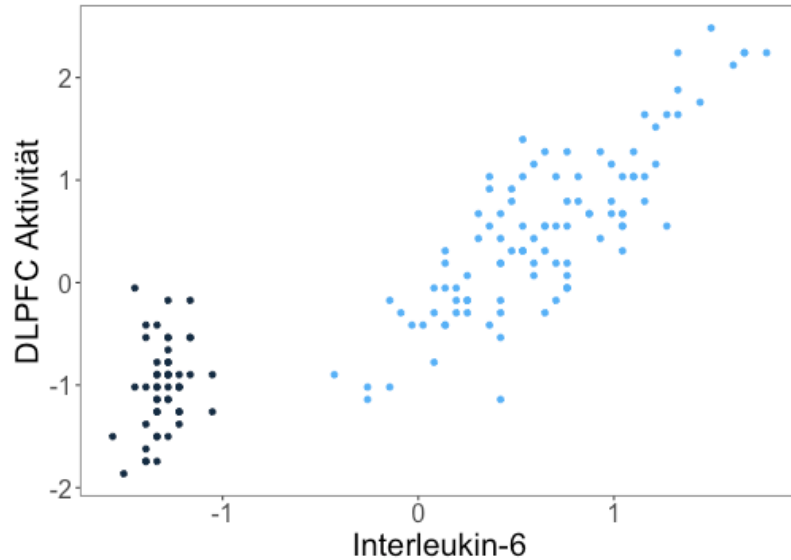
```
fviz_dend(hc, cex = 0.2, k = 2, color_labels_by_k = TRUE) + coord_flip() # Nach k einfärben
```



Agglomeratives hierarchisches Clustering

Zuweisung der Fälle zu den Clustern

```
df$cluster <- cutree(hc, k = 2)
```



```
ggplot(df, aes(y = `DLPFC Aktivität`,  
               x = `Interleukin-6`,  
               colour = cluster)) +  
  geom_point()
```

- Visualisierung im Streudiagramm
- Leicht möglich, da Daten 2-dimensional
- Jeder Punkt entspricht einer Person
- Cluster farblich visualisiert

Agglomeratives hierarchisches Clustering

Beispiel: Identifizierung von Depressionssubtypen anhand von Entzündungsmarkern und neuronaler Aktivität

Dimensionen für das Clustering (mehr als 2):

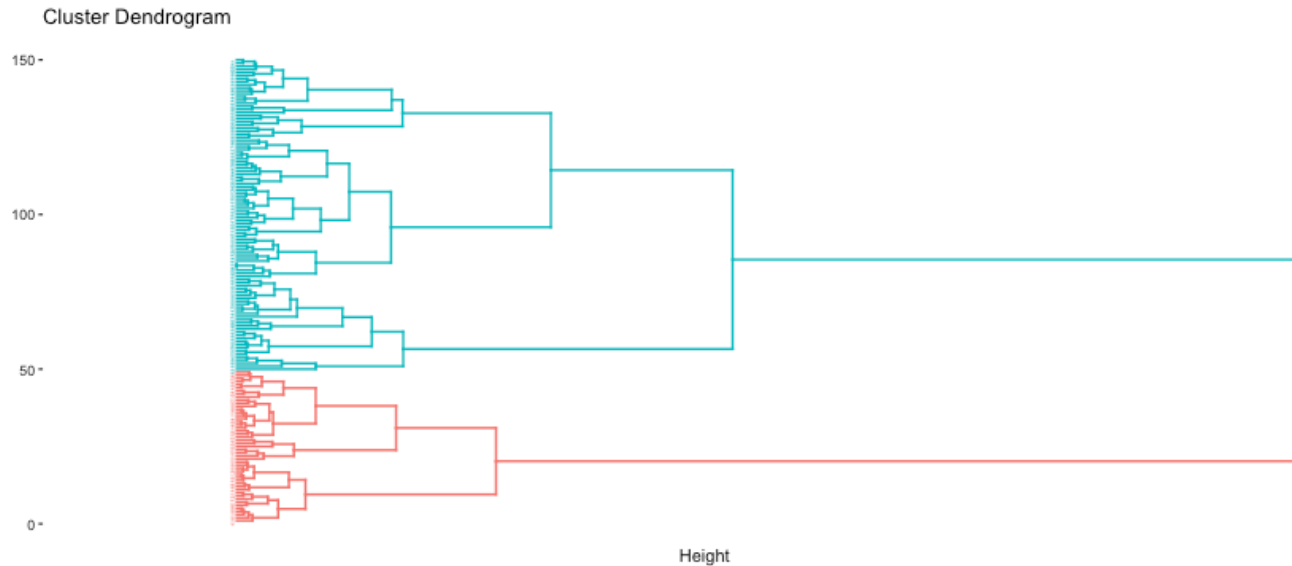
1. Entzündungsmarker: Interleukin-6 (IL-6)
2. C-reaktives Protein (CRP)
3. Tumor-Nekrose-Faktor-Alpha (TNF- α)
4. Neuronale Aktivität: DLPFC Aktivität

Ausschnitt von $N = 15$ Personen aus Datensatz (Werte sind z-standardisiert)

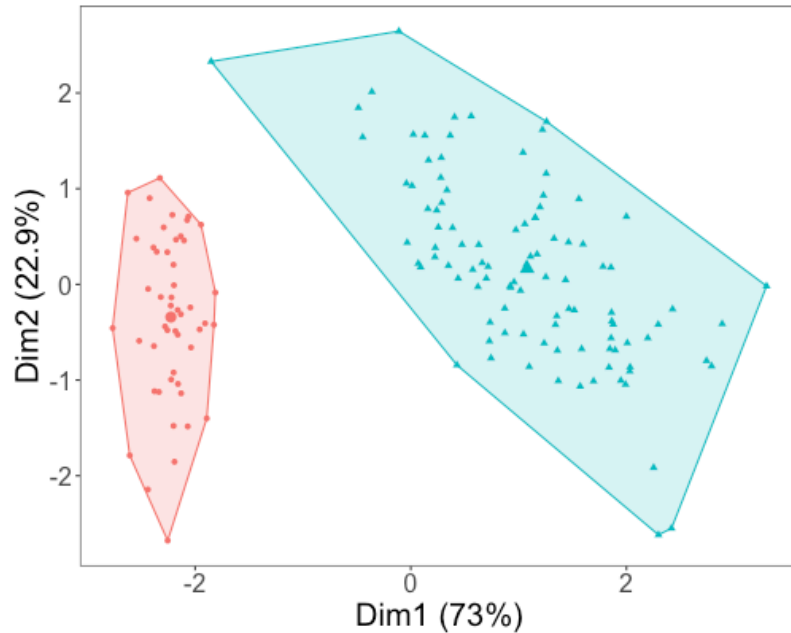
##		DLPFC	CRP	IL-6	TNF- α
## 1	-0.89767388	1.01560199	-1.335752	-1.311052	
## 2	-1.13920048	-0.13153881	-1.335752	-1.311052	
## 3	-1.38072709	0.32731751	-1.392399	-1.311052	
## 4	-1.50149039	0.09788935	-1.279104	-1.311052	
## 5	-1.01843718	1.24503015	-1.335752	-1.311052	
## 6	-0.53538397	1.93331463	-1.165809	-1.048667	
## 7	-1.50149039	0.78617383	-1.335752	-1.179859	
## 8	-1.01843718	0.78617383	-1.279104	-1.311052	
## 9	-1.74301699	-0.36096697	-1.335752	-1.311052	
## 10	-1.13920048	0.09788935	-1.279104	-1.442245	
## 11	-0.53538397	1.47445831	-1.279104	-1.311052	
## 12	-1.25996379	0.78617383	-1.222456	-1.311052	
## 13	-1.25996379	-0.13153881	-1.335752	-1.442245	
## 14	-1.86378030	-0.13153881	-1.505695	-1.442245	
## 15	-0.05233076	2.16274279	-1.449047	-1.311052	

Agglomeratives hierarchisches Clustering (mehr als 2 Dimensionen)

```
d <- dist(df, method = "euclidean")  
hc <- hclust(d = d, method = "ward.D2")  
fviz_dend(hc, cex = 0.2, k = 2, color_labels_by_k = TRUE) + coord_flip()
```



Agglomeratives hierarchisches Clustering (mehr als 2 Dimensionen)



```
cut <- cutree(hc, k = 2)

fviz_cluster(list(data = df,
                  cluster=cut),
              labelsize = 0) +
  mytheme +
  labs(title = "")
```

- Problem: 4 Dimensionen aber nur 2 Achsen
- Lösung: R rechnet erst eine Hauptkomponentenanalyse (PCA), um Daten auf 2 Komponenten herunterzubrechen
- Komponentenwerte werden auf X- und Y-Achse dargestellt → Clustertrennung sichtbar
- Nachteil: Variablen nicht in Ursprungseinheit sichtbar

Divisives hierarchisches Clustering

- Die divisive Clusteranalyse beginnt mit allen Objekten/Beobachtungen des Datensatzes in einem einzigen großen Cluster.
- In jeder Iteration wird der heterogenste Cluster in zwei Teilcluster aufgeteilt.
- Dieser Prozess wird so lange wiederholt, bis jedes Objekt seinen eigenen Cluster bildet

→ Gegenteil der agglomerativen Clusteranalyse.

Divisives hierarchisches Clustering

```
library(cluster)

di <- diana(x = df,
           stand = TRUE, # Standardisieren vor dem clustern
           metric = "euclidean")

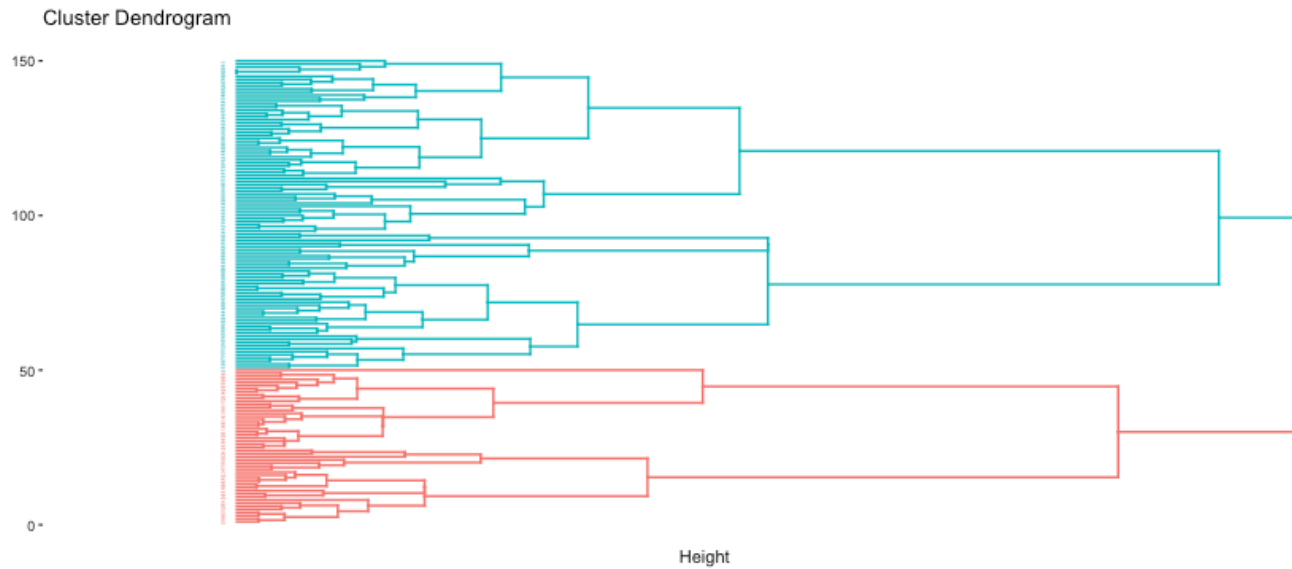
di$dc

## [1] 0.9408201
```

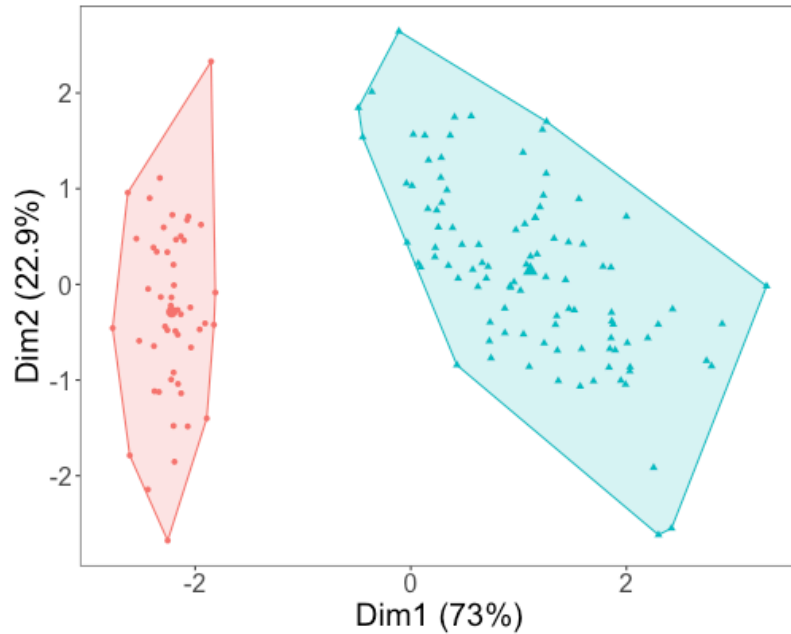
- Ergebnis: **Divisiver Koeffizient** - Maß für die gefundene Clusterstruktur
- Dieser entspricht im Wesentlichen der Unähnlichkeit zum ersten (gesamten) Cluster geteilt durch die Unähnlichkeit der Zusammenführung im letzten Schritt

Divisives hierarchisches Clustering

```
fviz_dend(di, cex = 0.2, k = 2) + coord_flip() # Einfärben passiert automatisch
```



Divisives hierarchisches Clustering

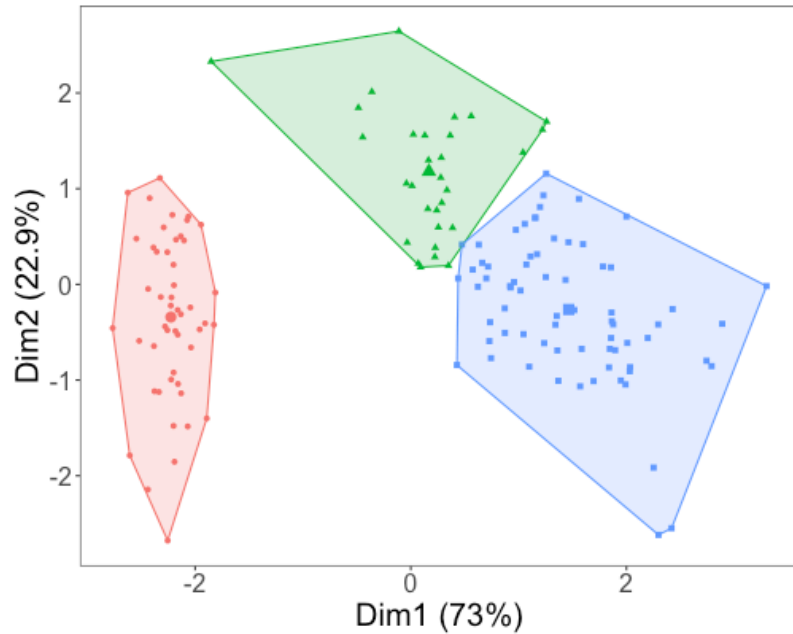


```
cut <- cutree(di, k = 2)

fviz_cluster(list(data = df,
                  cluster=cut),
             labelsize = 0) +
  mytheme +
  labs(title = "")
```

- Divisives findet eine ähnliche, jedoch nicht identische Clusterzuweisung

Wahl der optimalen Anzahl von Clustern

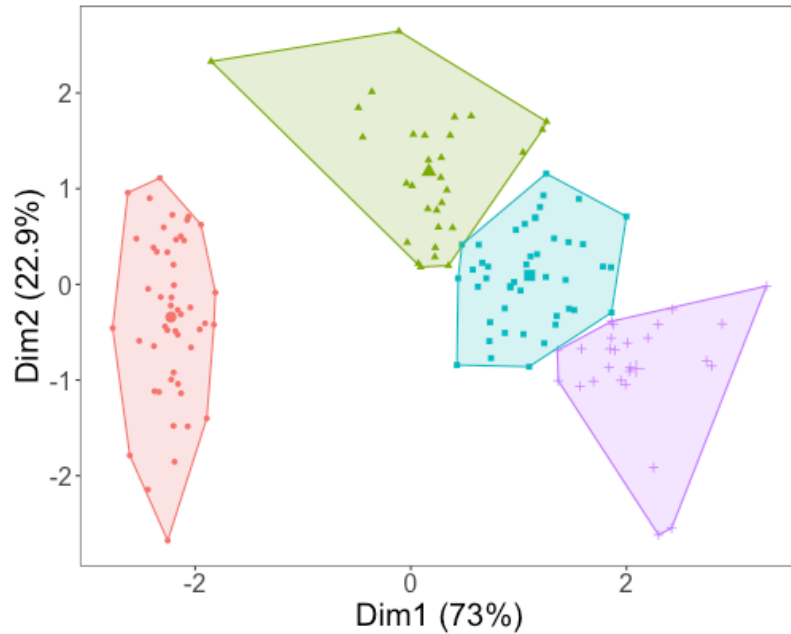


```
cut <- cutree(hc, k = 3)

fviz_cluster(list(data = df,
                  cluster=cut),
              labelsize = 0) +
  mytheme +
  labs(title = "")
```

- Prinzipiell wären auch andere Clusterzahlen denkbar
- Optimale Zahl muss bestimmt werden
- Visuelle Inspektion reicht dafür i.d.R. nicht

Wahl der optimalen Anzahl von Clustern



```
cut <- cutree(hc, k = 4)

fviz_cluster(list(data = df,
                  cluster=cut),
             labelsize = 0) +
  mytheme +
  labs(title = "")
```

- Prinzipiell wären auch andere Clusterzahlen denkbar
- Optimale Zahl muss bestimmt werden
- Visuelle Inspektion reicht dafür i.d.R. nicht

Wahl der optimalen Anzahl von Clustern

- k ist die Anzahl der Cluster, die ein Algorithmus erzeugt.
- Die Wahl beeinflusst direkt die Ergebnisse und die Interpretierbarkeit des Clusterings.
- Zu wenige Cluster: Heterogene Gruppen, wichtige Unterschiede werden übersehen.
- Zu viele Cluster: Übersegmentierung, Cluster sind schwer interpretierbar.

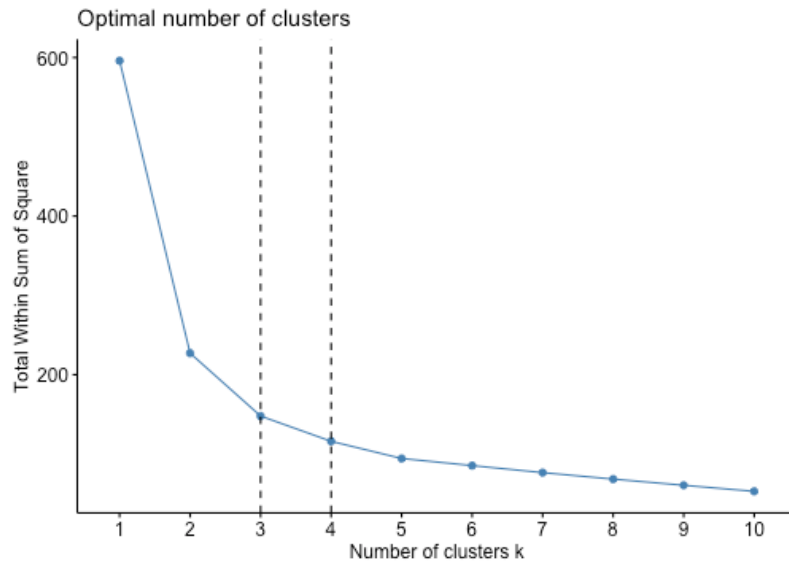
Wahl der optimalen Anzahl von Clustern

Methoden zur Wahl der optimalen Clusterzahl (Beispiele)

- Elbow-Methode:
 - Berechnung der Gesamt-Within-Cluster-Summe der Quadrate (WSS) für verschiedene k-Werte.
 - Identifikation des “Knicks” (Elbow), bei dem die WSS stark abflacht.
 - Beispiel: $k = 3$ liefert eine gute Balance zwischen Komplexität und Genauigkeit.
- Silhouettenanalyse:
 - Misst die Qualität eines Clusterings anhand der Kohäsion (Zusammenhalt innerhalb eines Clusters) und Separation (Trennung der Cluster).
 - Silhouetten-Koeffizient (-1 bis +1): Werte nahe +1 deuten auf gut getrennte Cluster hin.
- Gap-Statistik:
 - Vergleicht die Clusterung der tatsächlichen Daten mit zufälligen Daten.
 - Das optimale k maximiert den Unterschied zwischen beiden.

Wahl der optimalen Anzahl von Clustern

Elbow-Methode:



```
library(factoextra)

fviz_nbclust(x = df,
             FUN = hcut,
             method = c("wss"))
```

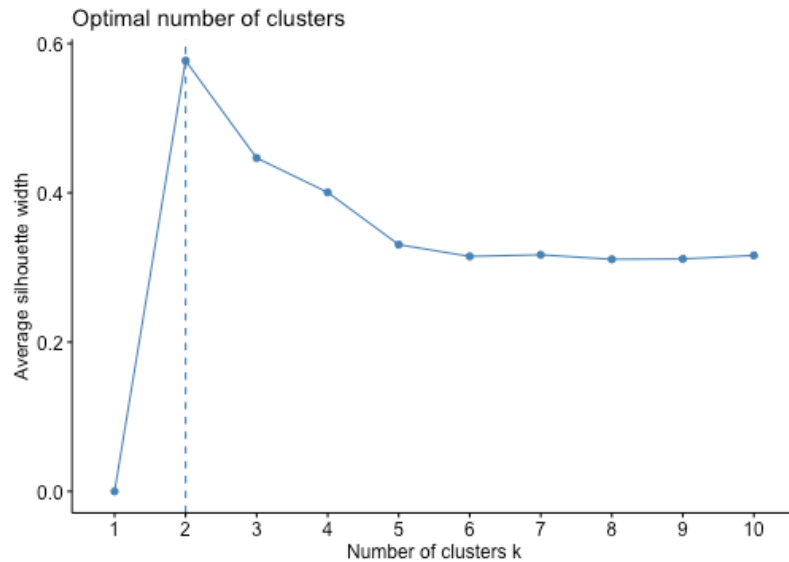
Wahl der optimalen Anzahl von Clustern

Elbow-Methode:

- Wir suchen nach dem „Knick“ im Diagramm, der darauf hinweist, dass zusätzliche Cluster nur noch geringfügigen Nutzen bringen.
- Hier: Der Knick deutet darauf hin, dass $k = 3$ wahrscheinlich angemessen ist.
- Im Wesentlichen gilt: Wenn der Liniendiagramm wie ein Arm aussieht, dann ist der „Ellbogen“ des Arms der Wert von k , der am besten geeignet ist.
- Die Idee ist, dass wir eine geringe WSS (Within-Cluster Sum of Squares) anstreben, aber die WSS dazu neigt, gegen 0 zu sinken, wenn k erhöht wird.
- Unser Ziel ist es, einen kleinen Wert für k zu wählen, der dennoch eine niedrige WSS aufweist.
- Der Knickpunkt („Ellbogen“) repräsentiert normalerweise den Punkt, an dem der Nutzen abnimmt, wenn k weiter erhöht wird.

Wahl der optimalen Anzahl von Clustern

Silhouettenanalyse:



```
fviz_nbclust(x = df,  
             FUN = hcut,  
             method = c("silhouette"))
```

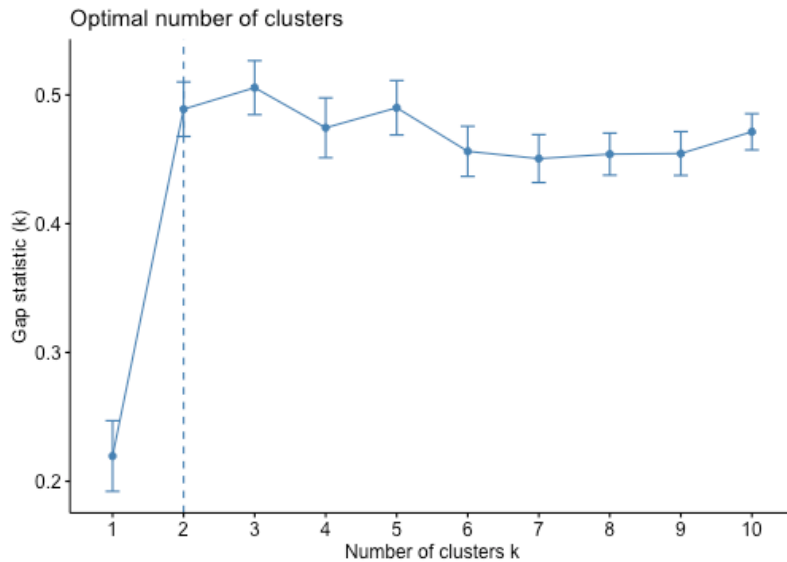
Wahl der optimalen Anzahl von Clustern

Silhouettenanalyse:

- Für jeden Punkt p wird zunächst die durchschnittliche Distanz zwischen p und allen anderen Punkten im selben Cluster berechnet (dies misst die Kohäsion, nennen wir sie A).
- Anschließend wird die durchschnittliche Distanz zwischen p und allen Punkten im nächsten Cluster berechnet (dies misst die Separation vom nächstgelegenen anderen Cluster, nennen wir sie B).
- Der Silhouetten-Koeffizient für p wird definiert als der Unterschied zwischen B und A , geteilt durch den größeren der beiden Werte ($\max(A, B)$).
- Ziel: Den Abstand zwischen Clustern messen.
- Wenn die Clusterkohäsion gut ist (A ist klein) und die Clustertrennung gut ist (B ist groß), wird der Zähler groß, was auf gut getrennte Cluster hinweist.
- Im Beispiel zeigt der zweite Plot, dass $k = 2$ der ideale Wert ist.

Wahl der optimalen Anzahl von Clustern

Gap-Statistik:



```
gap_stat <- clusGap(x = df,  
                    FUN = hcut,  
                    K.max = 10,  
                    B = 10)  
fviz_gap_stat(gap_stat)
```


Wahl der optimalen Anzahl von Clustern

Gap-Statistik:

- Logik: Qualität der Clusterung zu bewerten, indem die Variation innerhalb der Cluster mit der Variation verglichen wird, die bei einer zufälligen Verteilung der Daten zu erwarten wäre.
- Für jeden Wert von k wird die Summe der quadratischen Abstände zwischen den Punkten und ihren Clusterzentren berechnet (Kohäsion).
- Ein zufälliger Datensatz wird generiert, der die gleiche Anzahl von Datenpunkten und die gleichen Grenzen wie die Originaldaten hat.
- Für jeden potentiellen Wert von k wird die Variation innerhalb der Cluster auch im zufälligen Datensatz berechnet.
- Der Unterschied (Gap) zwischen der Variation im Referenzdatensatz und im Originaldatensatz wird berechnet:

$$\text{Gap}(k) = E[\log(W_k^{\text{random}})] - \log(W_k^{\text{data}})$$

- Der optimale Wert für k ist der Punkt, an dem die Gap-Statistik den größten Wert hat.
- Im Beispiel zeigt der dritte Plot, dass $k = 2$ der ideale Wert ist

Wahl der optimalen Anzahl von Clustern - Cluster Validity Indices und Majority rule

- Manche R Pakete berechnen viele Cluster Validity Indices (CVIs) auf einmal und entscheiden dann nach dem Mehrheitsprinzip (Majority rule)

```
library(NbClust)
nb = NbClust(df, distance = "euclidean", min.nc = 2, max.nc = 10, method = "ward.D2")
```

##	KL	CH	Hartigan	CCC	Scott	Marriot	TrCovW	TraceW	Friedman	Rubin	Cindex	DB	Silhouette	Duda	Pseudot2	Beale	Ratkowsky	Ball	Ptbiserial	Frey	McCl
## 2	4.22	240.25	79.38	2.86	344.78	1605323.8	1417.97	227.20	46.68	2.62	0.25	0.68	0.58	0.58	72.86	1.76	0.55	113.60	0.77	1.38	0.
## 3	2.71	222.72	40.31	4.00	460.89	1665618.1	1131.29	147.88	54.55	4.03	0.32	0.90	0.45	0.61	43.89	1.51	0.50	49.29	0.70	1.14	0.
## 4	1.14	201.25	33.55	3.28	568.42	1445874.1	1083.84	116.06	61.53	5.14	0.29	1.08	0.40	0.46	55.29	2.78	0.45	29.01	0.63	1.24	1.
## 5	11.95	192.68	15.18	2.72	635.44	1445078.9	683.25	94.37	66.05	6.32	0.32	1.07	0.33	0.65	14.76	1.23	0.41	18.87	0.58	0.45	1.
## 6	0.26	172.12	16.69	2.02	689.40	1452249.1	670.89	85.43	69.85	6.98	0.35	1.08	0.31	0.63	14.13	1.36	0.38	14.24	0.57	0.54	1.
## 7	0.73	161.71	17.14	1.89	759.18	1241321.6	511.50	76.56	79.63	7.79	0.36	1.07	0.32	0.44	34.44	2.97	0.35	10.94	0.56	0.81	1.
## 8	0.80	156.57	18.16	2.08	796.54	1263919.0	392.42	68.36	82.16	8.72	0.37	1.04	0.31	0.37	27.45	3.90	0.33	8.55	0.55	0.21	1.
## 9	0.80	155.69	20.46	2.57	853.89	1091330.6	339.73	60.61	87.72	9.83	0.37	0.95	0.31	0.71	17.95	0.98	0.32	6.73	0.55	0.77	1.
## 10	1.27	159.60	17.51	3.45	911.37	918477.9	242.05	52.93	91.14	11.26	0.38	0.99	0.32	0.58	16.42	1.65	0.30	5.29	0.50	0.55	1.
##	SDindex	Dindex	SDbw																		
## 2	1.51	1.06	0.41																		
## 3	1.68	0.88	0.53																		
## 4	2.17	0.78	0.54																		
## 5	1.92	0.70	0.47																		
## 6	2.07	0.68	0.22																		
## 7	2.04	0.64	0.17																		
## 8	2.09	0.60	0.15																		
## 9	2.08	0.58	0.14																		
## 10	2.44	0.54	0.10																		

Wahl der optimalen Anzahl von Clustern - Majority rule

```
library(NbClust)
nb = NbClust(df, distance = "euclidean", min.nc = 2, max.nc = 10, method = "ward.D2")
```

```
*****
* Among all indices:
* 10 proposed 2 as the best number of clusters
* 7 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 3 proposed 5 as the best number of clusters
* 2 proposed 7 as the best number of clusters
* 1 proposed 10 as the best number of clusters

***** Conclusion *****

* According to the majority rule, the best number of clusters is 2
```

Wahl der optimalen Anzahl von Clustern

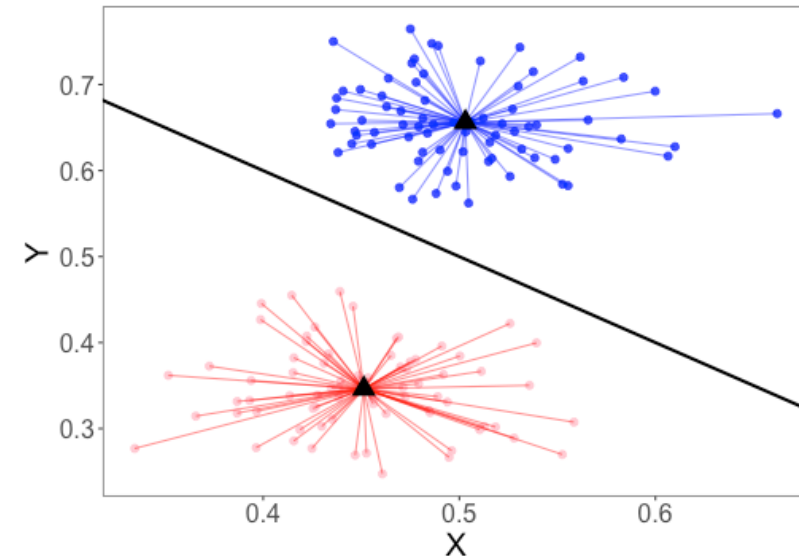
Top-Down - Vorwissen:

- Eine weitere Methode, die wir für das Clustering verwenden können, bezieht sich auf das theoretische „Vorwissen“
- Daraus könnten wir ableiten, dass $k = 3$ eine bessere Wahl sein könnte.
- Diese Methode sollte jedoch mit Vorsicht angewendet werden.
- Es ist ratsam, den Algorithmus mehrmals auszuführen und die Cluster zu analysieren, um zu überprüfen, welches k am besten geeignet ist.

Partitionierende Clusteranalyse

- Teilt die Daten in eine **vorgegebene Anzahl von Clustern** ein.
- Der bekannteste Ansatz ist der **K-Means-Algorithmus**:
- Ziel: Minimierung der **Intra-Cluster-Varianz**

Partitionierende Clusteranalyse (Centroid-based)



Partitionierende Clusteranalyse

- Im Allgemeinen bezeichnet partionelles Clustering einen Ansatz, bei dem eine gegebene Menge von n Objekten in k Partitionen aufgeteilt wird.

→ **k muss gegeben sein!**

- Ein Partitionierungsalgorithmus erstellt diese k Partitionen basierend auf den Daten.
- Der Wert für k wird von uns vorgegeben, wie nutzen die oben vorgestellten Methoden, um das geeignetste k zu bestimmen.
- Die grundlegenden Partitionierungsmethoden verwenden in der Regel eine exklusive Clustertrennung, bei der jede Beobachtung nur einem Cluster zugeordnet werden darf.
- Häufiger Algorithmus: K-Means Clustering

Partitionierende Clusteranalyse - K-Means

Der k-means-Algorithmus funktioniert im Allgemeinen in den folgenden Schritten:

1. Anzahl der Cluster k festlegen: Dies wird vom Datenwissenschaftler/Forscher bestimmt.
2. Initiale Clusterzentren auswählen: k Objekte werden zufällig aus dem Datensatz als anfängliche Clusterzentren (Mittelwerte) ausgewählt.
3. Zuweisung der Beobachtungen zu Clustern: Jede Beobachtung wird dem nächstgelegenen Clusterzentrum zugewiesen (euklidischen Distanz)
4. Aktualisierung der Clusterzentren: Für jedes der k Cluster wird das Clusterzentrum durch die Berechnung der neuen Mittelwerte aller Datenpunkte im Cluster aktualisiert.
5. Iterative Optimierung:
 - Minimierung der Gesamtsumme der Quadrate innerhalb der Cluster (Within Sum of Squares, WSS).
 - Schritte 3 und 4 werden wiederholt, bis sich die Clusterzuweisungen nicht mehr ändern oder die maximale Anzahl an Iterationen erreicht ist.

```
set.seed(123) # Zufällig gezogene Daten reproduzierbar machen

km <- kmeans(df, 2, nstart = 25) # wir wählen  $k = 2$ , da dies zuvor die beste Lösung war
km
```


Partitionierende Clusteranalyse - K-Means

Limitationen von K-Means

1. Erfordert Vorwissen über die Daten:
 - Der Analyst muss die geeignete Anzahl der Cluster (k) im Voraus festlegen.
2. Empfindlich gegenüber der zufälligen Auswahl der Startwerte:
 - Das Endergebnis hängt stark von der initialen zufälligen Auswahl der Clusterzentren ab.
 - Bei jedem Durchlauf des Algorithmus mit demselben Datensatz können unterschiedliche Startwerte gewählt werden.
 - Dies kann zu unterschiedlichen Clustering-Ergebnissen bei verschiedenen Durchläufen führen.
 - Daher ist die Verwendung von `set.seed(x)` entscheidend, um reproduzierbare Ergebnisse zu erzielen.
3. Empfindlich gegenüber Ausreißern:
 - Ausreißer können die Ergebnisse erheblich beeinflussen, da sie die Berechnung der Clusterzentren verzerren können.

Dichtebasierte Clusteranalyse - DBSCAN

- Partitionierungsmethoden (z.B. k-means) und hierarchisches Clustering eignen sich gut zur Identifikation von sphärisch geformten Clustern oder konvexen Clustern.
- Sie funktionieren gut, wenn die Cluster klar definiert und relativ deutlich voneinander getrennt sind.
- Diese Methoden haben jedoch Schwierigkeiten, wenn:
 - Extremwerte oder Ausreißer vorhanden sind.
 - Starkes Rauschen in den Daten vorliegt

Dichtebasierte Clusteranalyse - DBSCAN

- Density-Based Clustering of Applications with Noise (DBSCAN) ein interessanter alternativer Ansatz, der ein unüberwachter, nicht-linearer Algorithmus ist.
- Anstatt sich ausschließlich auf die Distanz zwischen Objekten/Datenpunkten zu konzentrieren, liegt der Fokus hier auf der Dichte.
- Die Daten werden in Gruppen mit ähnlichen Eigenschaften oder Clustern unterteilt, ohne dass die Anzahl der Cluster vorab spezifiziert werden muss.
- Ein Cluster wird als eine maximale Menge dicht verbundener Punkte definiert.
- Vorteil von DBSCAN: Kann Cluster beliebiger Formen in rauschbehafteten räumlichen Datenbanken erkennen.

Dichtebasierte Clusteranalyse - DBSCAN

- Ziel: ähnlich dichte Regionen innerhalb eines Datensatzes zu identifizieren.
- Es gibt zwei relevante Parameter für DBSCAN:
 - Epsilon (ϵ): Definiert den Radius der Nachbarschaft um einen gegebenen Punkt.
 - Minimale Anzahl von Punkten (MinPts): Die minimale Anzahl von Nachbarn innerhalb des eps-Radius.
- Kernpunkte (core points):
 - Wenn ein Punkt mindestens MinPts Nachbarn hat, gilt er als Kernpunkt.
 - Kernpunkte befinden sich typischerweise im Zentrum des Clusters, da diese Regionen als dicht angesehen werden.
- Randpunkte (border points):
 - Wenn ein Punkt weniger Nachbarn als MinPts hat, gilt er als Randpunkt.
 - Randpunkte sind eher peripher und befinden sich am Rand eines Clusters.
 - Rauschen oder Ausreißer:
- Punkte, die so weit entfernt sind, dass sie nicht in einen eps-Radius mit genügend Nachbarn fallen, werden als Rauschen oder Ausreißer klassifiziert.

Dichtebasierte Clusteranalyse - DBSCAN

DBSCAN funktioniert im Allgemeinen wie folgt:

1. Zufällige Auswahl eines Punktes p :
 - Für den Punkt p werden alle Punkte ermittelt, die "erreichbar" sind.
 - Das bedeutet, sie liegen innerhalb des Maximalradius der Nachbarschaft (ϵ) und erfüllen die Bedingung der minimalen Anzahl von Punkten (MinPts) in der ϵ -Nachbarschaft.
2. Markierung von Kernpunkten:
 - Jeder Punkt, der mindestens MinPts Nachbarn hat, wird als Kernpunkt oder als besucht markiert.
3. Clusterbildung:
 - Für jeden Kernpunkt, der noch keinem Cluster zugewiesen ist, wird ein neuer Cluster erstellt.
 - Rekursiv werden alle dichteverbundenen Punkte des Kernpunktes gefunden und dem gleichen Cluster zugewiesen.
4. Durchlaufen der restlichen unbesuchten Punkte:
 - Der Algorithmus wiederholt die Schritte für die verbleibenden unbesuchten Punkte im Datensatz.

Dichtebasierte Clusteranalyse - DBSCAN

Beispiel: Clusteranalyse von aktivierten Voxeln in einem fMRI-Experiment

- Ein Experiment untersucht die neuronale Aktivierung im präfrontalen Kortex während einer Aufgabe, bei der Probanden emotionale Bilder betrachten.
- Die Aktivierung wird mittels funktioneller Magnetresonanztomographie (fMRI) gemessen.
- Wir nehmen der Einfachheit halber 2D Daten an (normalerweise werden 3D Voxel genutzt)
- Jeder Pixel hat eine Position (x, y) und eine Aktivität

Ziel:

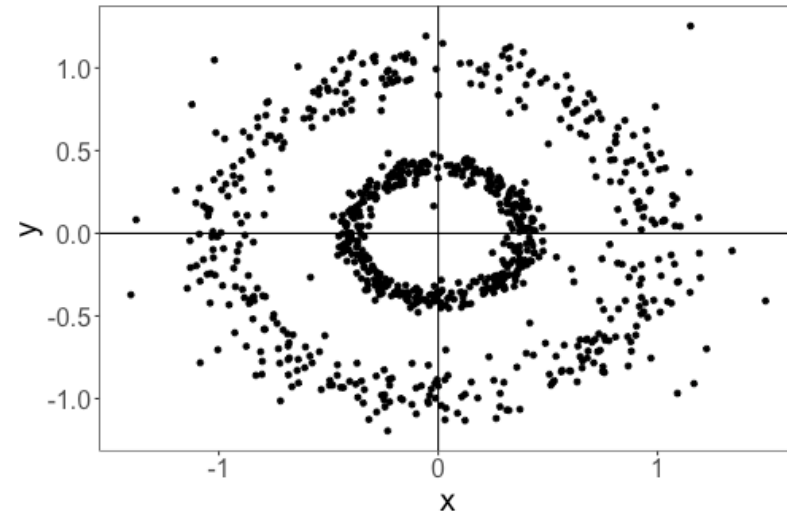
- Clusterbildung der aktivierten Regionen basierend auf den räumlichen Koordinaten (x, y, z) und der Aktivierungsintensität.
- Identifikation von funktionalen Clustern im Gehirn, die mit der Verarbeitung emotionaler Bilder zusammenhängen.

Dichtebasierte Clusteranalyse - DBSCAN

```
ggplot(df, aes(x, y)) +  
  geom_vline(xintercept = 0) +  
  geom_hline(yintercept = 0) +  
  geom_point()
```

Beispiel: Clusteranalyse von aktivierten Voxeln in einem fMRI-Experiment

- Jeder Pixel hat eine Position (x, y) und eine Aktivität
- Aktive Pixel werden im Streudiagramm rechts dargestellt

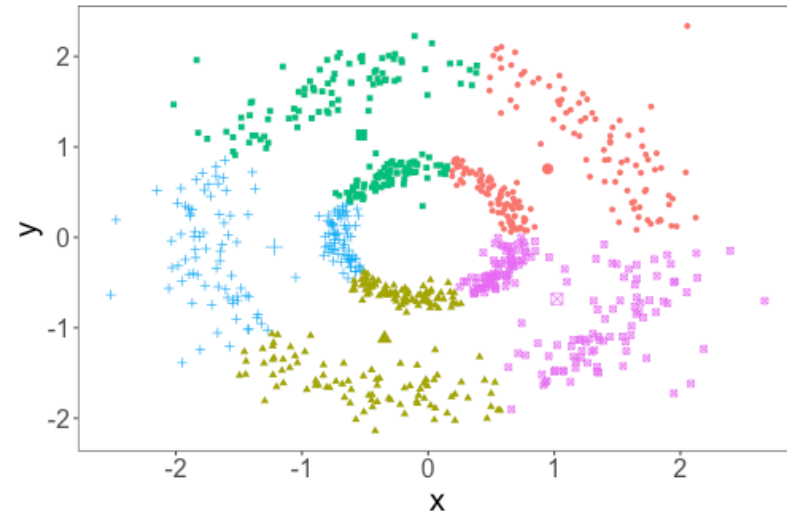


Dichtebasierte Clusteranalyse - DBSCAN

```
set.seed(123)
km <- kmeans(df, 5, nstart = 25)
fviz_cluster(km, df,
             ellipse.type = "point",
             geom = c("point"))
```

- kmeans-Clustering führt zu suboptimalen Ergebnissen
- Voxel werden aufgrund ihrer Nähe, aber nicht basierend auf der klar sichtbaren Ringform zugeordnet
- Dies könnte zu Fehlinterpretationen (falsch-Lokalisierung von Hirnfunktionen und Netzwerken führen)

→ Ein flexiblerer Algorithmus wird benötigt



Dichtebasierte Clusteranalyse - DBSCAN

Parameterbestimmung

- DBSCAN-Algorithmus erfordert, dass Benutzer die optimalen Werte für ϵ und den Parameter MinPts festlegen.
- Einschränkung von DBSCAN: Algorithmus ist empfindlich gegenüber der Wahl von ϵ ist, insbesondere wenn die Cluster unterschiedliche Dichten aufweisen.
- Konsequenzen schlecht gewählter ϵ -Werte
 - Wenn der ϵ -Wert zu klein gewählt wird, werden dünner besetzte Cluster als Rauschen definiert.
 - Wenn der ϵ -Wert zu groß gewählt wird, können dichter besetzte Cluster miteinander verschmolzen werden.

Dichtebasierte Clusteranalyse - DBSCAN

Parameterbestimmung

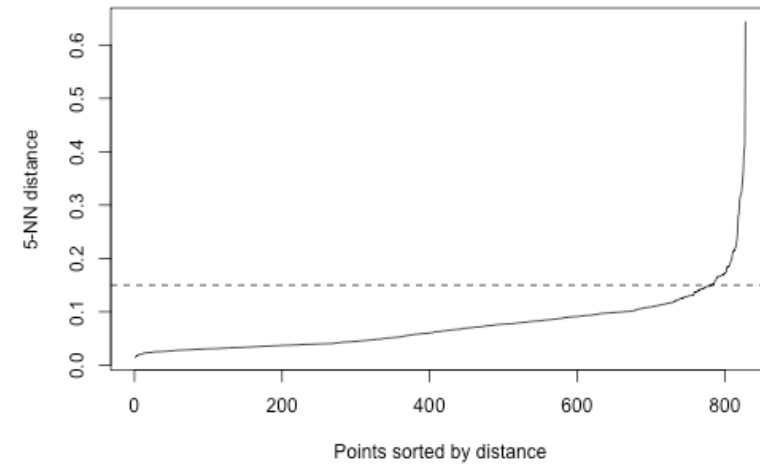
- Eine gängige Methode besteht darin, die k-nearest neighbour-Distanzen zu berechnen.
- Damit wird die durchschnittliche Distanz jedes Punktes zu seinen nächsten Nachbarn berechnet.
- Der Wert von k wird dabei vom Datenwissenschaftler/Analysten festgelegt und entspricht dem MinPts-Wert.
- Anschließend erstellen wir ein sogenanntes knee-Diagramm (ähnlich dem „Elbow-Plot“ bei k-means).
- Wir suchen den Punkt, an dem ein starker Knick nach oben erkennbar ist → dieser Punkt wird als unser ϵ -Wert verwendet.

Dichtebasierte Clusteranalyse - DBSCAN

Parameterbestimmung

```
library('dbscan')  
library(fpc)  
  
dbscan::kNNdistplot(df, k = 5) +  
  abline(h = 0.15, lty = 2)
```

- Der Schnittpunkt für ϵ scheint bei ~ 0.15 zu liegen



```
## integer(0)
```

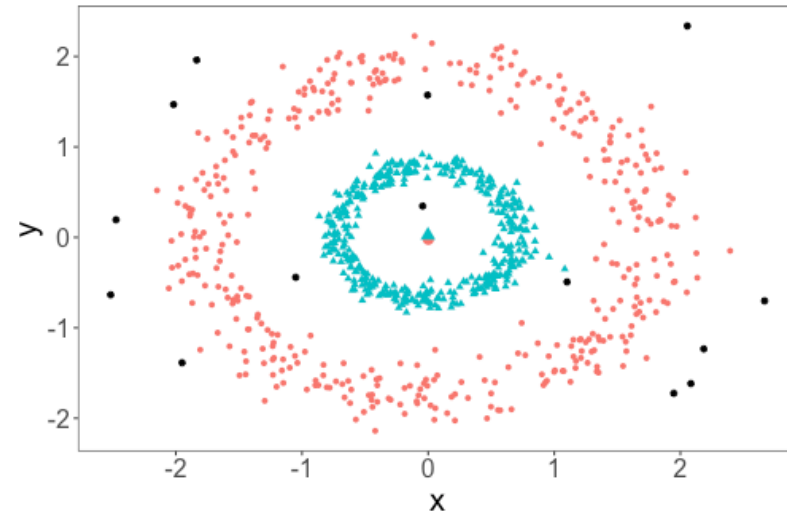
Dichtebasierte Clusteranalyse - DBSCAN

```
set.seed(123)

db <- dbscan(df, eps = 0.15, MinPts = 5)

fviz_cluster(db, df,
  ellipse.type = "point",
  geom=c("point"))
```

- DBSCAN ist sensitiv für komplexe Datenstrukturen (Ring)
- Es findet zwei distinkte Hirnregregionen, in welchen aktivierte Voxel liegen
- Graue Punkte = Ausreißer



Modellbasiertes Clustering (Soft Clustering)

- Modellbasierte Clusteranalyse basiert auf einem statistischen Modell für die Daten, üblicherweise einem **Mixture Modell**
- Zugehörigkeit zu einem Cluster wird über Wahrscheinlichkeiten modelliert (maximum likelihood estimation)
- Häufig genutzte Modelle:
 - Bei kontinuierlichen Daten: Gaussian mixture model (GMM)
 - Bei kategorialen Daten: Latent class model (LCM)
- Vorteile:
 - Bietet eine fundierte statistische Grundlage für das Clustern.
 - Bewertet die Unsicherheit der Clusterzuweisung.
 - Identifiziert Ausreißer, die keiner Gruppe zugehörig sind.

Modellbasiertes Clustering (Soft Clustering)

- Für jede der n Beobachtungen liegen Daten zu d Variablen vor, die für Beobachtung i als $y_i = (y_{i,1}, \dots, y_{i,d})$ bezeichnet werden.
- Modellbasierte Clusteranalyse drückt die Wahrscheinlichkeitsdichtefunktion von y_i als endliche Mischung oder gewichteten Durchschnitt von G Komponenten-Wahrscheinlichkeitsdichtefunktionen aus:

$$p(y_i) = \sum_{g=1}^G \tau_g f_g(y_i \mid \theta_g)$$

wobei f_g eine Wahrscheinlichkeitsdichtefunktion mit dem Parameter θ_g ist.

- τ_g ist die entsprechende Mischungswahrscheinlichkeit, wobei gilt: $\sum_{g=1}^G \tau_g = 1$.
- In ihrer einfachsten Form betrachtet die modellbasierte Clusteranalyse jede Komponente des Mixture Modells als ein einzelnes Cluster.
- Die Methode schätzt die Modellparameter und weist jeder Beobachtung den Cluster zu, der der wahrscheinlichsten Mischungs-Komponente entspricht.

Arten der Clusteranalyse - Hard vs. Soft vs. Fuzzy

Merkmal	Hard Clustering	Soft Clustering (Modell basiert)	Fuzzy Clustering
Mitgliedschaftsrepräsentation	Jeder Datenpunkt gehört genau zu einem Cluster (binäre Zuordnung).	Wahrscheinlichkeiten (Summe = 1), die die Wahrscheinlichkeit der Zugehörigkeit zu jedem Cluster anzeigen.	Mitgliedschaftsgrade (zwischen 0 und 1), die den Grad der Zugehörigkeit zu jedem Cluster angeben (Summe = 1).
Grenzen	Klare und genau definierte Cluster Grenzen.	Überlappende Grenzen mit probabilistischer Mitgliedschaft.	Überlappende Grenzen mit abgestuften Mitgliedschaften.
Konzepte	Deterministisch, feste Zuordnungen.	Wahrscheinlichkeitstheorie (basierend auf Likelihood).	Distanzbasiert (Fuzziness-Parameter m).
Beispiele für Algorithmen	K-Means, hierarchisches Clustering.	Gaussian Mixture Models (GMMs).	Fuzzy C-Means (FCM).
Interpretierbarkeit	Einfach und leicht zu interpretieren.	Mittlere Komplexität durch Wahrscheinlichkeiten.	Mittlere Komplexität durch Mitgliedschaftsgrade.
Optimaler Anwendungsfall	Klar abgegrenzte, nicht überlappende Cluster.	Bei Unsicherheiten oder überlappenden Clustern.	Wenn Clusterzugehörigkeiten nicht diskret sind und variieren.

Grenzen der Clusteranalyse

1. Interpretierbarkeit der Ergebnisse

- Die Bedeutung der gefundenen Cluster ist oft nicht eindeutig und hängt stark von der Domänenkenntnis ab.
- Es besteht das Risiko von überinterpretierten Clustern, die keine echte Trennung in den Daten widerspiegeln.
- Beispiel: Cluster können nur durch technische Artefakte (z.B. Datenskalen) entstehen und keinen inhaltlichen Wert haben.

2. Stabilität der Clusterlösung

- Ergebnisse der Clusteranalyse sind oft empfindlich gegenüber Parametereinstellungen (z.B. Distanzmetrik).
- Bei kleineren Änderungen der Daten können sich die Cluster komplett ändern, was die Reproduzierbarkeit erschwert.
- Beispiel: Unterschiedliche Startwerte bei k-means führen zu unterschiedlichen Lösungen.

3. Generalisierbarkeit auf neue Daten

- Clusteranalysen werden meist auf einem statischen Datensatz durchgeführt und sind oft nicht direkt auf neue Daten anwendbar.
- Beispiel: Cluster aus einer Stichprobe von Patienten könnten in einer anderen Population nicht auftreten.

- **Ziel der Clusteranalyse** ist die Identifikation von Gruppen in den Daten, die intern möglichst homogen und extern möglichst heterogen sind.
- Clusteranalyse umfasst **verschiedene Ansätze** (z. B. k-means, hierarchisches Clustering, DBSCAN), die unterschiedliche Anforderungen und Stärken haben.
- Die Wahl der richtigen **Anzahl von Clustern (k)** oder anderen **Parametern** (z. B. ϵ bei DBSCAN) ist entscheidend für die Qualität der Ergebnisse.
- Zur Überprüfung der **Qualität und Stabilität** der Clusterlösungen können Methoden wie Silhouettenanalyse, Gap-Statistik oder externe Validierung genutzt werden.
- Die Ergebnisse können von Parametern, Ausreißern, der Wahl der Distanzmetrik und der Datenstruktur stark **beeinflusst** werden.
- Die Bedeutung der Cluster ist oft nicht eindeutig und erfordert **Fachwissen**, um sie sinnvoll zu deuten.