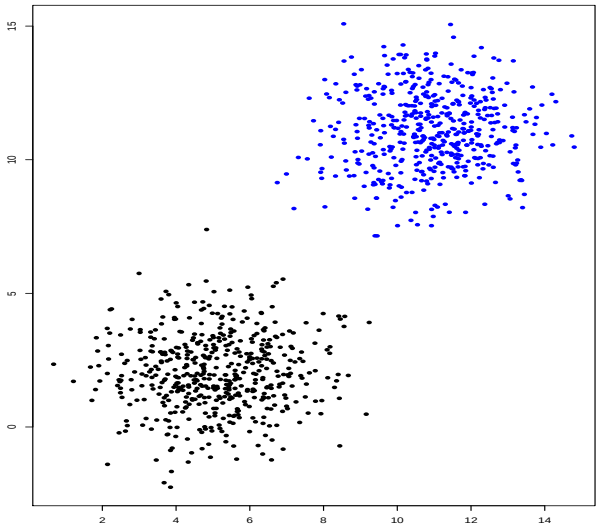
Institut für Angewandte Statistik
Johannes Kepler Universität Linz

Distanz- und modellbasiertes Clustern eines phänotypischen Datensatzes



Bachelorarbeit zur Erlangung des akademischen Grades eines
Bachelor of Science

Daniel Dober
Betreuerin: Mag.[a] Dr.[in] Gertraud Malsiner-Walli M.Stat.

August 2016

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Begriffserklärungen

Um die Thematik dieser Arbeit vollständig zu verstehen, ist es eine Notwendigkeit, zuerst ein paar wichtige Begriffe der Genetik zu erklären.

Der Begriff welcher zu allererst erklärt werden muss, ist ***Phänotyp***. Als Phänotyp wird das „Erscheinungsbild eines Organismus“ (vgl. Graw (2010)) verstanden. Damit sind Eigenschaften von Organismen gemeint, welche mit freiem Auge ersichtlich sind, z.B. die Größe einer Pflanze.

Ein weiterer grundlegender Begriff ist ***Genotyp***. In der Genetik bezeichnet man die „Gesamtheit aller erblichen Eigenschaften eines Organismus“ (vgl. Graw (2010)) als Genotyp. Im allgemeinen Sprachgebrauch wird dafür oft auch DNA bzw. DNA-Strang verwendet. Anders als eine phänotypische Eigenschaft ist der Genotyp nicht mit freiem Auge ersichtlich und kann nur mit zeitaufwendigen chemischen Verfahren bestimmt werden.

Schlussendlich ist noch der Begriff eines ***Genmarkers*** wichtig. Hierbei meint man „eindeutig identifizierbare, kurze DNA-Abschnitte, deren Ort im Genom bekannt ist“ (vgl. Graw (2010)). Speziell in dieser Arbeit sind damit die einzelnen Gene eines Genstrangs (Genotyp) gemeint.

1.2 Motivation

In der heutigen Zeit gibt es eine Vielzahl an Getreideprodukten, die in allen möglichen Geschäften angeboten werden. Um ein möglichst gutes Endprodukt zu erhalten, wird die Auswahl an richtigen Getreidesorten immer wichtiger, d.h. man möchte die „besten“ Getreidesorten züchten.

Um diese zu bekommen, möchte man herausfinden, welchen Einfluss bestimmte Genmarker (=Gene) auf die phänotypischen Eigenschaften des Getreides haben. Üblicherweise modelliert man ein Problem dieser Art mit einem Regressionsmodell, womit man den Effekt der einzelnen Genmarker auf eine bestimmte Eigenschaft ermitteln kann.

In dieser Arbeit geht es darum herauszufinden, ob diese Fragestellung auch mit einer Clusteranalyse lösbar ist. In der folgenden Arbeit werden Gerstenpflanzen (Beobachtungen) nach ihren phänotpyischen Eigenschaften geclustert, d.h. in Gruppen eingeteilt. Anschließend wird untersucht ob sich die gefundenen Gruppen hinsichtlich der Genmarker unterscheiden. Schlussendlich werden die Ergebnisse der Clusteranalyse mit denen eines Regressionsmodells verglichen.

Kapitel 2

Daten

2.1 Allgemein

Der in dieser Arbeit verwendete Datensatz stammt von dem in Nordamerika durchgeführten Projekt *North American Barley Genome Mapping Project (NABGMP)*. Dieser Datensatz wurde auch in den Artikeln von Xu (2007) und Zhao & Xu (2012) verwendet. In diesem Projekt wurden zwei Gerstenarten namens **Harrington** und **TR306** 145 mal gekreuzt. Von der jeweils entstandenen Tochtergeneration wurde der genetische Code (DNA) von 127 Genmarkern ermittelt. Dieser Genstrang wurde mit $+1$ für das Harrington Allel und -1 für das TR306 Allel codiert, d.h. die Ausprägung gibt an, aus welcher der beiden Elternpflanzen das Gen stammt.
Auf diese Weise wurden die Genstränge von 145 genetisch verschiedenen Gerstenkörner festgehalten. Dies ist der genotypische Teil des Datensatzes.

Die so erzeugten genetischen Muster wurden in Folge auf ihre „Tauglichkeit“ hinsichtlich Getreidequalität überprüft. Um die 145 genetisch verschiedenen Gerstenkörner in mehreren Umgebungen und somit unter verschiedenen Bedingungen (Umwelteinflüsse) testen zu können, wurde jedes genetische Muster mehrmals repliziert. Es handelt sich hierbei also um „Kopien“ der Pflanzen, da 2 Pflanzen welche exakt dasselbe Genmuster aufweisen theoretisch als ein und dieselbe Pflanze betrachtet werden können.
Im Weiteren wurde für jede der 145 genetisch verschiedenen Pflanzen je eine Kopie in 19 verschiedenen Umgebungen (Orte in Kanada und den USA) in zwei aufeindanderfolgenden Jahren angepflanzt. Das bedeutet, dass 145×38 Gerstenkörner gesetzt wurden. Von jedem angepflanzten Gerstenkorn wurden anschließend aus der resultierenden Gerstenpflanze bis zu 7 phänotypische Eigenschaften („Merkmale“) erhoben. Diese sind: „Tage bis zum Keimen der Pflanze“, „Höhe der Pflanze“, „Gewicht des Gerstenkorns“, „Biegung des Stiels“, „Tage bis zur Reife der Pflanze“, „Testgewicht der Pflanze“ und „Ernteertrag der Pflanze“. Die als stetige Variablen erhobenen Merkmale bzw. Eigenschaften stellen

den phänotypischen Datenteil dar.

Da nicht immer jede der 7 phänotypischen Eigenschaften in jeder Umgebung gemessen wurde, sind in Tabelle 2.1 die Abkürzungen der Eigenschaften, die Eigenschaften selbst und die Anzahl der Umgebungen, in denen diese gemessen wurden, angegeben.

		Anzahl der Umgebungen	
Kürzel	Variable	1992	1993
hed	days to heading	15	14
hgt	height	15	12
kwt	kernel weight	12	13
ldg	lodging	9	8
mat	maturity	7	8
twt	test weight	15	13
yld	yield	14	14

Tabelle 2.1: Übersicht der phänotypischen Eigenschaften

Da zwei verschiedene Pflanzen mit exakt dem selben genetischen Muster als dieselbe Pflanze betrachtet werden, hat man schlussendlich 145 Beobachtungen, an denen bis zu 7 Eigenschaften in bis zu 29 verschiedenen Umgebungen gemessen wurden.
Der Datensatz liegt somit in der in Tabelle 2.2 dargestellten Form vor.

	phänotypischer Teil					genotypischer Teil		
	hed			yld				
	1992	1993	...	1992	1993	M1	...	M127
	U1 ... U15	U1 ... U14		U1 ... U14	U1 ... U14			
1						1	...	1
2						1	...	1
...	stetige	Variablen	...	stetige	Variablen	...	...	...
144						-1	...	1
145						1	...	-1

Tabelle 2.2: Übersicht des gesamten Datensatzes, U1, ... ,U15 = Umgebungen, M1, ... ,M127 = Marker

In den 127 Variablen (M1 bis M127) des genotypischen Datenteils ist das genetische Muster der jeweiligen Beobachtung festgehalten. So stammt beispielsweise das „erste“ Gen (M1) der ersten Beobachtung von der Mutterpflanze namens Harrington.

Der Datensatz lag ursprünglich in 2 verschiedenen Formaten gesplittet vor. Die phänotypischen Daten waren in .phe-Dateien gespeichert, die genetischen in einer .csv-Datei. Der erste Schritt dieser Bachelorarbeit bestand darin, die R-Codes zum Einlesen der Daten und Abspeichern als txt-Datei zu schreiben. Die Codes sind im Anhang angefügt.

Um eine geeignete phänotypische Eigenschaft zum Clustern zu finden, wurden die Histogramme der Eigenschaften untersucht. Vergleicht man die Histogramme der Eigenschaft `maturity` aus dem Jahr 1992 (s. Abbildung 2.1) mit den Histogrammen der anderen Eigenschaften (hier nicht gezeigt), würde man bei der Eigenschaft `maturity` am ehesten eine Clusterstruktur vermuten. Der Pairs-Plot von `maturity` (1992) in Abbildung 2.2 bestärkt diese Vermutung. So würde man z.B. im Streudiagramm der ersten und siebten Variable 3 Gruppen vermuten. Aus diesen Gründen wurde in diesem Bericht diese Variable zur Clusteranalyse verwendet.

2.2 Variable `maturity`

Die Variable `maturity` gibt an, wie lange der Reifungsprozess der Gerstenpflanzen, gemessen in Tagen, gedauert hat. Diese Eigenschaft wurde in 15 verschiedenen Umgebungen, in 2 aufeinanderfolgenden Jahren (1992-1993), gemessen. Für die folgenden Clusterverfahren wurden nur die Werte von 1992 verwendet, um Querschnittsdaten zu erhalten.

Somit blieben 7 Umgebungen übrig, in denen die Zeit bis zur Reife der Gerstenpflanzen im Jahr 1992 gemessen wurde. Dieser finale phänotypische Datensatz besteht aus 145 Beobachtungen mit 7 Variablen und wird im folgenden `mat92` genannt.

In Tabelle 2.3 befinden sich die Mittelwerte und Standardabweichungen der 7 Variablen/Umgebungen von `mat92`.

Umgebungen	Mittelwert	Standardabweichung
Alberta - Irricana	97.85	3.65
Alberta - Edmonton	91.79	1.35
Saskatchewan - Outlook	102.38	2.21
Manitoba - Brandon	94.77	1.82
Craig - Ailsa	83.84	1.10
Ontario - Elora	97.92	0.69
Quebec - Ste.Anne.de.Bellevue	92.77	1.20

Tabelle 2.3: Übersicht - Teildatensatz `mat92`

Auf diesen Teildatensatz werden nun in dieser Arbeit die in den Kapiteln 3 und 4 beschriebenen Clusterverfahren angewendet und analysiert.

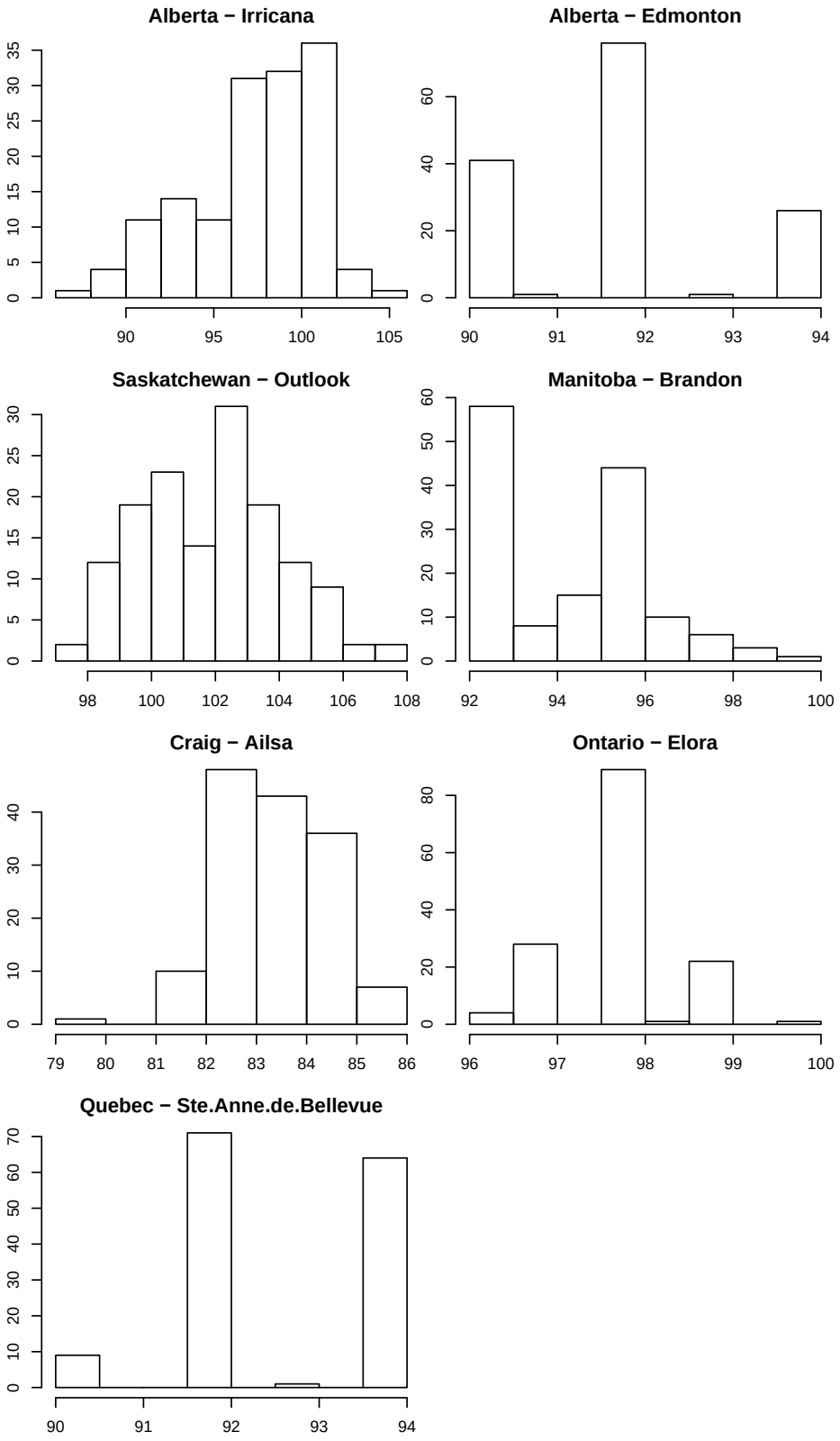


Abbildung 2.1: Histogramme der 7 Variablen von `mat92`

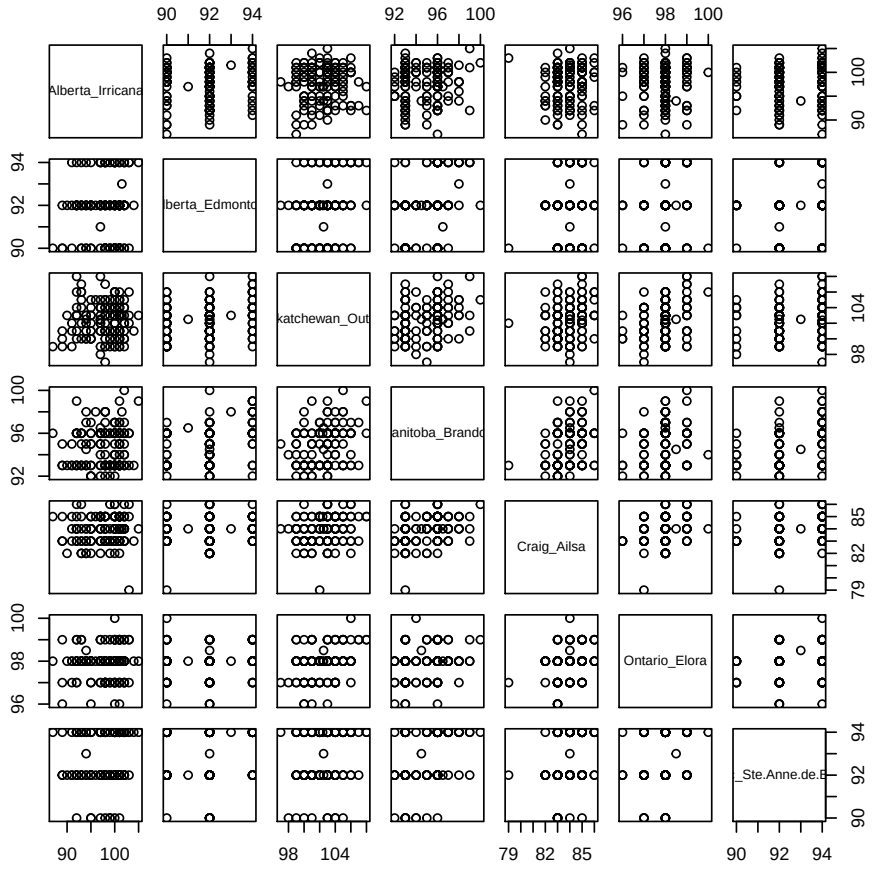


Abbildung 2.2: Pairs-Plot von `mat92`

Kapitel 3

Distanzbasiertes Clustern

3.1 Ähnlichkeits- und Distanzmaße

Möchte man eine Clusteranalyse „ohne Modellannahme“ durchführen, muss man Ähnlichkeit bzw. ein Distanzmaß definieren. Wie diese Maße definiert sind und wie sie auf verschiedene Arten berechnet werden können, wird in diesem Kapitel behandelt.

Es gibt verschiedene Definitionen von Ähnlichkeits- bzw. Distanzmaßen. Ziel dieser Maße ist es, für n Beobachtungen, an denen p Merkmale erhoben wurden, eine Ähnlichkeit zu bestimmen. Dabei soll die Zahl s_{ij} die Ähnlichkeit zwischen dem i-ten und j-ten Objekt bzw. Beobachtung messen. Dies ist der sogenannte Ähnlichkeitskoeffizient, welcher, sofern dieser normiert ist, die Eigenschaft

$$0 \leq s_{ij} \leq 1$$

besitzt.
Je größer diese Zahl ist, desto ähnlicher sind sich die entsprechenden Objekte.

Umgekehrt kann auch das Distanzmaß d_{ij} betrachtet werden, welches die Unähnlichkeit zwischen dem i-ten und j-ten Objekt misst. Dieses ist umso kleiner, je ähnlicher sich diese Objekte sind. Distanzmaße können folgendermaßen aus normierten Ähnlichkeitskoeffizienten s_{ij} berechnet werden:

$$d_{ij} = 1 - s_{ij}$$

Daraus folgt:

$$0 \leq d_{ij} \leq 1$$

Alle Distanzen (zwischen allen Paaren der n Objekte) werden in der sogenannten Distanz-

matrix D zusammengefasst.

$$\mathbf{D} = \begin{pmatrix} d_{11} & \dots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \dots & d_{nn} \end{pmatrix}$$

Je nach Messniveau der Daten werden unterschiedliche Distanzmaße und Ähnlichkeitsmaße verwendet. Da in dieser Arbeit nur metrische Daten geclustert werden, werden nur die zwei gebräuchlichsten Maße für dieses Messniveau erläutert.

Metrik

Distanzen basieren auf Metriken. Unter einer Metrik versteht man eine Funktion, welche je zwei Elemente eines Raums einen nichtnegativen reellen Wert zuordnet, der als Abstand zwischen diesen beiden Elementen voneinander verstanden werden kann.

Definition

Sei X eine beliebige Menge. Dann ist die Abbildung $d : X \times X \to \Re$ eine Metrik auf X, wenn für beliebige Elemente x, y und z von X folgende Axiome gelten:

1. Positive Definitheit: $d(x, y) \geq 0$ und $d(x, y) = 0$ falls $x = y$
2. Symmetrie: $d(x, y) = d(y, x)$
3. Dreiecksungleichung: $d(x, y) \leq d(x, z) + d(z, y)$

3.2 Distanzmaße

Die hier verwendeten Distanzen zwischen dem i-ten und j-ten Objekt mit den Merkmalsvektoren

$$\mathbf{x}_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix}, \qquad \mathbf{x}_j = \begin{pmatrix} x_{j1} \\ \vdots \\ x_{jp} \end{pmatrix}$$

sind folgende:

Euklidische Distanz

Die euklidische Distanz ist das am häufigsten verwendete Distanzmaß und ist definiert durch die Wurzel der Summe der quadrierten Differenz zweier Beobachtungsvektoren:

$$d_{ij} = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)}.$$

Manhattan Metrik
Die Manhattan Metrik definiert die Distanz zwischen 2 Beobachtungen als die Summe ihrer absoluten Differenzen:

$$d_{ij} = \sum_{k=1}^{p} |x_{ik} - x_{jk}|.$$

Basierend auf diesen Distanzen werden die Daten in den Kapiteln 3.3 und 3.4 mithilfe des k-Means-Verfahrens und der hierarchischen Clusteranalyse geclustert.

3.3 k-Means

Das k-Means-Verfahren bietet eine einfache und schnelle Möglichkeit für eine gegebene Anzahl von Klassen/Clustern K, eine „optimale“ Partition der n Objekte in diese K Klassen zu finden. Als optimal wird eine Partition dabei angesehen, wenn die Objekte sich innerhalb einer Klasse sehr ähnlich sind, aber die Beobachtungen zwischen verschiedenen Klassen sich möglichst stark unterscheiden.

Input
Ausgangslage des k-Means-Verfahren sind die Daten mit n Beobachtungen für p Merkmale $(\mathbf{y}_i, \ldots, \mathbf{y}_n)$ und die Anzahl der Klassen K. Die Tatsache, dass die Anzahl der Klassen im Vorhinein bekannt sein muss, ist ein wesentlicher Nachteil dieses Verfahrens.

Ziel
Ziel des k-Means-Verfahren ist es, anhand des oben beschriebenen Inputs, folgende Funktion zu minimieren:

$$\sum_{k=1}^{K} \sum_{i \in C_k} ||\mathbf{y}_i - \mathbf{m}_k||^2$$

Dabei entspricht C_k der Menge der Beobachtungen in der Klasse k und $\mathbf{m}_k$ dem Klassenzentrum der jeweiligen Klasse k.
Im Laufe der Jahre wurden verschiedene Algorithmen entwickelt um die Zielfunktion zu minimieren. Bei diesen, zumeist iterativen, Verfahren muss beachtet werden, dass der Algorithmus auch gegen ein lokales Maximum konvergieren kann.

Einer der bekanntesten Algorithmen, welcher im folgenden beschrieben wird, ist der "`Lloyd`" bzw. "`Forgy`" Algorithmus.

Algorithmus

1. Jedes Objekt wird zufällig einer Klasse zugeordnet.

2. Gegeben die Klassenzugehörigkeiten, bestimmt man im zweiten Schritt die Klassenzentren $\mathbf{m}_k = \overline{\mathbf{y}}_k$, wobei $\overline{\mathbf{y}}_k$ der Mittelwert der k-ten Klasse ist.

3. Gegeben diese Klassenzentren, wird nun jede Beobachtung ihrer nächstgelegenen Klasse $C(i)$ zugeordnet.

$$C(i) = arg \underset{j=1,\ldots,K}{min} ||\mathbf{y}_i - \overline{\mathbf{y}}_j||^2$$

4. Nun iteriert man die Schritte 2 und 3 solange, bis keine Änderungen bei den Klassenzugehörigkeiten mehr auftreten und somit Konvergenz eintritt.

Da der Algorithmus für bestimmte Startwerte gegen ein lokales Minimum konvergieren kann, sollte dieser mit vielen verschiedenen Zuordnungen durchsimuliert werden.

Die Frage, welche sich nun stellt, ist, wieviele Klassen gewählt werden sollen. Eine der gängigsten Methoden, die optimale Klassenanzahl zu bestimmen, ist der durchschnittliche Silhouettenwert einer Clusterlösung. Der Silhouettenkoeffizient wird in Kapitel 3.5 erläutert.

3.4 Hierarchische Clusteranalyse

Die hierarchische Clusteranalyse hat wie das k-Means-Verfahren eine Distanzmatrix als Ausgangspunkt. Diese Art des Clusterns hat den Vorteil, dass die Anzahl der Cluster im Vorhinein nicht bekannt sein muss, denn es wird für $k = 1, \ldots, n$ eine Clusterlösung berechnet. Dennoch ist die Bestimmung der Anzahl der Cluster auch bei diesem Verfahren nicht einfach, bzw. oft nicht eindeutig.

Bei der hierarchischen Clusteranalyse wird eine Folge von Partitionen P_i der zu clusternden Objekte erzeugt. Der Index i gibt dabei an aus wievielen Klassen die jeweilige Partition besteht. Es gibt 2 verschiedene Arten wie diese Folge aussehen kann. Startet man mit n Klassen, wobei jede Beobachtung eine eigene Klasse bildet, so ist dies ein **agglomeratives** Verfahren und es wird die Partitionsfolge

$$P_n, P_{n-1}, \ldots, P_2, P_1$$

erzeugt.
Beginnt man hingegen mit einer Klasse, nennt man dies ein **divisives** Verfahren, wobei die umgekehrte Folge

$$P_1, P_2, \ldots, P_{n-1}, P_n$$

erzeugt wird.
Diese beiden Verfahren erzeugen nicht notwendigerweise dieselben Partitionen P_i.

Es gilt aber für beide Verfahren:

- die Partition P_i besteht aus i Klassen und
- die Partitionen P_i und P_{i+1} haben $i - 1$ Klassen gemeinsam

Sowohl beim agglomerativen als auch beim divisiven Verfahren braucht man zusätzlich zur Distanzmatrix, in der die Distanzen zwischen den Beobachtungen definiert sind, auch ein Distanzmaß zwischen Gruppen/Klassen von Beobachtungen. Für die Definition einer Distanz $D_{i.j}$ zwischen den Gruppen (i) und (j) gibt es verschiedene Möglichkeiten, die noch näher beschrieben werden.

3.4.1 Das agglomerative Verfahren

Man hat eine Distanzmatrix $\mathbf{D}$ und sei d_{ij} die zugehörige Distanz zwischen dem i-ten und j-ten Objekt. Weiter sei $D_{i.j}$ definiert als die Distanz zwischen der i-ten und j-ten Klasse.

Die Vorgehensweise des agglomerativen Clusterverfahrens kann folgendermaßen beschrieben werden:

1. Jedes Objekt wird als eigene Klasse definiert, sodass gilt $D_{i.j} = d_{ij}$

2. In diesem Schritt wird die minimale Distanz zwischen den Klassen ermittelt. Das heißt
$$min\{D_{i.j}|D_{i.j} > 0\}$$
wird bestimmt.
Sollte es mehrere minimale Werte geben, wird einer davon zufällig ausgewählt.

3. Sei $D_{a.b}$ der kleinste Wert. Dann werden die Klassen a und b zu einer Klasse D_{ab} verschmolzen.

4. Im vierten Schritt werden die Distanzen zwischen der neu erzeugten Klasse D_{ab} und den restlichen R Klassen bestimmt. Diese Distanzen $D_{ab.r}$ für $r = 1, \ldots, R$ ersetzen danach die a-te und b-te Zeile und Spalte von $\mathbf{D}$.

5. Die Schritte 2 bis 4 werden solange wiederholt bis nur noch eine Klasse vorliegt.

Bei dieser Beschreibung ist allerdings die in Schritt 4 zu bestimmende Distanz nicht eindeutig für eine Menge von Objekten definiert. Hierfür werden die sogenannten Linkage-Verfahren verwendet.

Linkage-Verfahren
Es gibt nun verschiedene Ansätze wie der Abstand zwischen Mengen von Objekten definiert sein kann. Diese werden *Linkage-Verfahren* genannt. Im wesentlichen unterscheiden wir 3 verschiedene Linkage-Verfahren, *Single-*, *Complete-*, und *Average-Linkage*.

Wir betrachten die i-te, j-te und k-te Klasse, mit den jeweiligen Abständen $D_{i.j}$, $D_{i.k}$ und $D_{j.k}$, welche gegeben sind. Außerdem sei $D_{i.j}$ der kleinste Abstand in der Distanzmatrix, was bedeutet, dass die Klassen i und j verschmolzen werden.
Gesucht ist jetzt der Wert $D_{ij.k}$ welcher je nach Linkage-Verfahren unterschiedlich definiert ist.

Single-Linkage
Bei diesem Verfahren wird das Minimum der Distanzen $D_{i.k}$ und $D_{j.k}$ als neue Distanz verwendet, d.h.
$$D_{ij.k} = min\{D_{i.k}, D_{j.k}\}$$

In der Abbildung 3.1 kann der durch das Single-Linkage-Verfahren definierte Abstand (blaue Linie) zwischen zwei, durch gestrichelte Kreise markierte Cluster, betrachtet werden.

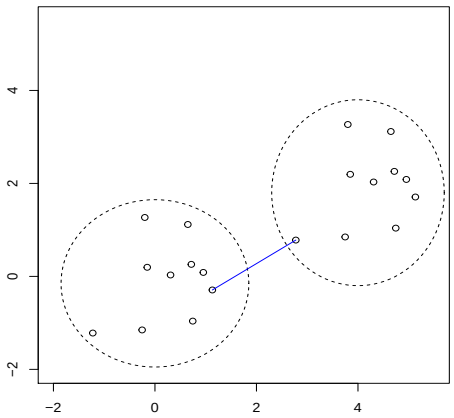


Abbildung 3.1: Beispiel für Single-Linkage

Diese Art der Bestimmung von Distanzen von Klassen führt dazu, dass dieses Verfahren *kontrahierend* ist. Dies bedeutet, dass tendenziell öfters einzelne Beobachtungen zu bestehenden Klassen hinzugefügt werden, anstatt das Gruppen von Beobachtungen zu einer Klasse verschmolzen werden. Durch diese sogenannte Kettenbildung können eventuelle Ausreißer erkannt werden, da diese spät zu einer Klasse hinzugefügt werden.

Complete-Linkage

Diese Definition ist das „Gegenstück“ zum Single-Linkage. Es verwendet das Maximum der Distanzen $D_{i.k}$ und $D_{j.k}$ als die neue Distanz, d.h.

$$D_{ij.k} = max\{D_{i.k}, D_{j.k}\}$$

Abbildung 3.2 zeigt dieselben Cluster wie Abbildung 3.1. Diesmal wurde der Abstand laut dem Complete-Linkage-Verfahren mit einer blauen Linie eingezeichnet.

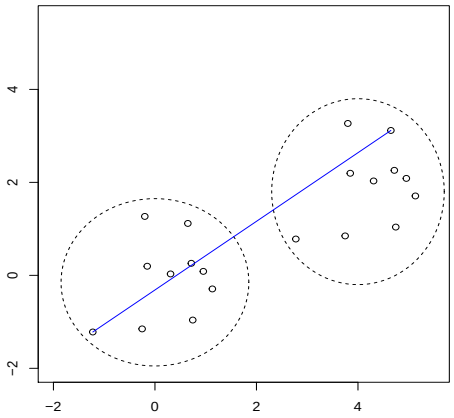


Abbildung 3.2: Beispiel für Complete-Linkage

Das Complete-Linkage-Verfahren ist ein *dilatierendes*. Damit ist gemeint, dass einzelne Objekte sich eher von Klassen entfernen und mit anderen Objekten eigene bzw. neue Klassen bilden.

Average-Linkage

Beim Average-Linkage-Verfahren ist die Distanz zwischen zwei Gruppen als der Mittelwert der Distanzen zwischen allen Elementen der Klassen definiert. Da dieses Verfahren durch die Art der Distanzenbildung weder kontrahierend noch dilatierend ist, wird es als *konservatives* Verfahren bezeichnet.

3.4.2 Das divisive Verfahren

Beim divisiven Verfahren wird mit einer Klasse begonnen und anschließend die zwei „unterschiedlichsten" Unter-Cluster gesucht, in welche die Objekte aufgeteilt werden. Diese zwei Unter-Cluster werden danach wieder in zwei Unter-Cluster aufgeteilt. Die Aufteilung in diese Unter-Cluster erfolgt solange, bis jedem Objekt ein eigener Cluster zugeordnet wird.

Bei diesem Verfahren geht man wie folgt vor:

1. Gestartet wird mit einer Klasse C, in welcher sich alle n Objekte befinden.

2. Im nächsten Schritt wird das Objekt gesucht, welches den größten Abstand zu allen anderen Objekten aufweist. Dieses Objekt bildet dann eine neue separate Klasse C_{neu}, in welcher sich nur dieses eine Objekt befindet, d.h.

   $$C \text{ wird in 2 Klassen } C_{alt} \text{ und } C_{neu} \text{ aufgeteilt,}$$

   welche aus n_{neu} bzw. n_{alt} Elementen bestehen.

3. Nun wird für jedes verbliebene Objekt im ersten Cluster C_{alt} aus dem durchschnittlichen Abstand zu allen Objekten im neuen Cluster

   $$d_{neu}(i) = \frac{1}{n_{neu}} \sum_{l \in C_{neu}} d_{il}$$

   und dem durchschnittlichen Abstand zu allen verbleibenden Objekten des ursprünglichen Clusters

   $$d_{alt}(i) = \frac{1}{n_{alt} - 1} \sum_{j \in C_{alt}, j \neq i} d_{ij}$$

die Differenz $D(i)$ berechnet:

$$D(i) = d_{alt}(i) - d_{neu}(i)$$

4. Anschließend wird das Objekt mit dem größten Wert für $D(i)$ der neuen Klasse zugeordnet.

5. Die Schritte 2 und 3 werden solange durchgefürt, bis D_M einen negativen Wert annimmt, wobei

   $$D_M = max\ D(i),\ \text{für } i = 1, \ldots, n_{alt}$$

   ist. Das heißt, solange bis es keine Objekte mehr im ursprünglichen Cluster gibt, die näher an Objekten des neuen Clusters liegen als an Objekten des ersten Clusters.

6. Dieses Splitting wird auf beide Unter-Cluster angewendet und solange fortgeführt, bis jedes Objekt in einem eigenen Cluster liegt.

3.5 Güte der Clusterlösung

Die Güte einer Clusterlösung kann mithilfe des Silhouettenkoeffizienten bzw. mit Silhouettenplots beurteilt werden.

3.5.1 Silhouetten

Diese Art der Visualisierung einer Clusterlösung wurde von Rousseeuw (1987) entwickelt. Der Vorteil von Silhouetten ist, dass die Lösungen beliebiger Clusterverfahren dargestellt und verglichen werden können.

Sind folgende Werte gegeben:

- eine beliebige Distanzmatrix $\mathbf{D} = (d_{ij})$,
- K Klassen, welche mit $C_1, \ldots, C_K$ bezeichnet sind,
- die Anzahl der Elemente der K Klassen $n_1, \ldots, n_K$,
- das Objekt i gehört zur Klasse C_k $(i \in C_k)$

dann ist der Silhouettenwert $s(i)$ der i-ten Beobachtung folgendermaßen definiert:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}.$$

Dabei ist

$$a(i) = \frac{1}{n_k - 1} \sum_{j \in C_k, j \neq i} d_{ij} \geq 0$$

der mittlere Abstand des i-ten Objektes zu allen anderen Objekten, welche in der gleichen Klasse sind, und

$$b(i) = \min_{j \neq k} d(i, C_j) \geq 0$$

ist der mittlere Abstand des i-ten Objektes zu derjenigen Klasse welche der eigenen am nächsten ist, mit

$$d(i, C_j) = \frac{1}{n_j} \sum_{l \in C_j} d_{il}.$$

Sollte eine Klasse nur aus einem Objekt i bestehen, dann gilt $s(i) = 0$.

Für die Silhouettenwerte gilt daher: $s(i) \in [-1, 1]$. Sie können folgendermaßen interpretiert werden:

- $s(i)$ nahe 1:
  Das i-te Objekt liegt im Mittel näher bei den Objekten der eigenen Klasse, als an den Objekten der nächstgelegenen Klasse.

- $s(i)$ nahe 0:
  Das i-te Objekt liegt im Mittel genauso nah an Objekten seiner Klasse wie an Objekten der Klasse, welche ihm am nächsten liegt.

- $s(i)$ nahe -1:
  Das i-te Objekt liegt im Mittel näher bei den Objekten der nächstliegenden Klasse, als an den Objekten seiner eigenen Klasse.

Die Darstellung der Silhouettenwerte erfolgt anhand eines Balkendiagramms. Die Werte werden aufgeteilt nach den Klassen und der Größe nach sortiert eingetragen. Neben jeder dieser K Klassen wird der Mittelwert der jeweiligen Gruppe angezeigt. Der größte Wert kommt dabei zuerst. Unter dem Balkendiagramm wird schließlich noch der Durchschnitt aller Silhouettewerte $\overline{s}(K)$ angezeigt.

Kaufman & Rousseeuw (1990) schlagen vor für $K = 2, \ldots, N - 1$ die durchschnittlichen Silhouettenwerte $\overline{s}(K)$ zu bestimmen und die Partition zu wählen, bei der $\overline{s}(K)$ am größten ist.

3.6 Umsetzung in R

In dieser Arbeit wurden die Funktionen `kmeans` und `hclust` verwendet, um das k-Means-Verfahren bzw. das agglomerative hierarchische Clusterverfahren durchzuführen. Beide befinden sich im Standardpaket `stats` in `R`.
Für das divisive hierarchische Clusterverfahren wurde die im R-Paket `cluster` enthaltene Funktion `diana` verwendet. In diesem Paket befinden sich außerdem noch die Funktionen `silhouette` und `daisy`, welche für die Berechnung der Silhouettenwerte bzw. Distanzen verwendet wurde.

Kapitel 4

Modellbasiertes Clustern

Im Gegensatz zu den im Kapitel 1 besprochenen Clusterverfahren beruht das modellbasierte Clustern nicht auf Distanzen. Somit muss also auch keine Distanzmatrix $\mathbf{D}$ berechnet werden. Stattdessen wird den Daten ein geeignetes stochastisches Modell angepasst bzw. wird dieses angenommen.
Die folgenden Erläuterungen des modellbasierten Clusterverfahrens wurden den Büchern von Mclachlan & Peel (2000) und Fahrmeir et al. (1996) entnommen.

4.1 Mischverteilung

Man geht bei dieser Art zu clustern davon aus, dass die Beobachtungen $\mathbf{x}_1, \ldots, \mathbf{x}_n$ Realisationen des Zufallsvektors $\mathbf{X}$ sind und die Klassenstruktur in den Daten daraus entsteht, dass $\mathbf{X}$ in jeder Klasse eine andere Verteilung besitzt. Bei dieser Art der Verteilung von $\mathbf{X}$ handelt es sich um eine *Mischverteilung*. Des Weiteren geht man davon aus, dass die Klassenverteilungen Verteilungen mit bekanntem Verteilungstyp (meist eine Normalverteilung), jedoch mit unbekannten Parametern sind.
Die unbekannte Klassenzugehörigkeit der Beobachtungen wird als diskrete Zufallsvariable S mit dem Wertebereich $\{1, \ldots, K\}$ angesehen.

4.2 Das Modell

Auch wenn für die Verteilung der Klassen eine beliebige Verteilung angenommen werden kann, wird üblicherweise von einer Normalverteilung ausgegangen.

Angenommen, eine Grundgesamtheit C zerfalle in K Klassen/Cluster $C_1, \ldots, C_K$, sodass gilt

$$C = \bigcup_{k=1}^{K} C_k$$

wobei die Klassen disjunkt sind, d.h.

$$C_k \cap C_j = \emptyset \text{ für } k \neq j.$$

Jedem Objekt i ist sowohl die Ausprägung eines p-dimensionalen Merkmals $\mathbf{X}$ als auch die Klassenzugehörigkeit $k(\Leftrightarrow i \in C_k)$ als Ausprägung der entsprechenden Zufallsgröße S zugeordnet.
Die Verteilung von S ist durch die a priori-Wahrscheinlichkeiten

$$P(S = k) = P(i \in C_k) = \pi_k \text{ mit } k = 1, \ldots, K$$

gegeben.
Des Weiteren sei $\mathbf{X}$ in der Klasse C_k mit der Dichte

$$f_N(\mathbf{X}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

verteilt (*Klassenverteilung*). f_N entspricht dabei der Dichte einer multivariaten Normalverteilung mit dem Mittelwertsvektor $\boldsymbol{\mu}_k$ und der Varianz-Kovarianzmatrix $\boldsymbol{\Sigma}_k$:

$$f_N(\mathbf{X}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^p|\boldsymbol{\Sigma}_k|}} exp\left[-\frac{1}{2}(\mathbf{X} - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{X} - \boldsymbol{\mu}_k)\right] \tag{4.1}$$

Somit ist dann die gemeinsame Verteilung von $\mathbf{X}$ und $\mathbf{S}$ gegeben durch

$$\pi_k f_N(\mathbf{X}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4.2}$$

für $k = 1, \ldots, K$.
Wir nehmen an, dass die Parameter $\pi_k, \boldsymbol{\mu}_k$ und $\boldsymbol{\Sigma}_k$ unbekannt sind und damit

$$\Theta = \{\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K\}$$

zu schätzende Parametervektor ist.
Nach dem Satz der totalen Wahrscheinlichkeit ist die Randverteilung von $\mathbf{X}$ durch die Mischverteilung

$$p(\mathbf{X}|\Theta) = \sum_{k=1}^{K} \pi_k f_N(\mathbf{X}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4.3}$$

gegeben.
Von den Mischwahrscheinlichkeiten π_k wird dabei folgendes gefordert:

$$0 \leq \pi_k \leq 1 \text{ und } \pi_k > 0$$

Die Parametermenge Θ ist, wie bereits erwähnt, unbekannt und muss nun aus den Daten geschätzt werden.

Maximum-Likelihood-Methode

Die Loglikelihood-Funktion der Mischverteilung in Gleichung 4.3 ist gegeben durch

$$\begin{aligned} \log L(\Theta|\mathbf{X}) &= \log \left[\prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f_N(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right] \\ &= \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \pi_k f_N(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]. \end{aligned} \tag{4.4}$$

Allerdings stellt sich die direkte Maximierung dieser Funktion als nicht geeignet heraus, da die Maximierung des Logarithmus einer Summe schwierig ist. Stattdessen verwendet man in diesem Fall den sogenannten EM-Algorithmus.

4.3 EM-Algorithmus

Der Expectation-Maximization-Algorithmus, kurz EM-Algorithmus genannt, ist ein Algorithmus, welcher unter bestimmten Bedingungen gegen den ML-Schätzer konvergiert. Diese Annäherung an das Maximum der Loglikelihood-Funktion erfolgt mittels eines iterativen Verfahrens.

Das Grundkonzept dieses Algorithmus besteht aus

1. der Annahme einer latenten Zufallsvariable und
2. einem iterativen Verfahren mit den Schritten
   a) Expectation (E-Step) und
   b) Maximization (M-Step).

Latente Indikatorvariable

Beim EM-Algorithmus für Mischverteilungen geht man davon aus, dass es unbeobachtete (latente) Zufallsvariablen $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ mit $z_{ik} \in \{0, 1\}$ gibt. Diese geben an, aus welcher Verteilungskomponente/Klasse eine Beobachtung stammt. Wurde die i-te Beobachtung aus der k-ten Komponente erzeugt, ergibt sich folgender Vektor

$$\mathbf{z}_i = \begin{pmatrix} & & & k & & \\ \ldots & 0 & 1 & 0 & \ldots \end{pmatrix}$$

welcher an der k-ten Stelle einen Einser aufweist und an den restlichen $(K-1)$ Stellen eine Null.

Die Wahrscheinschlichkeit dafür, dass die i-ten Beobachtung aus der k-ten Komponente stammt, ist

$$p(z_{ik} = 1) = \pi_k, \qquad \text{für } k = 1, \ldots, K.$$

Betrachtet man nun die Loglikelihood-Funktion, bei der nicht nur die Daten $\mathbf{X}$, sondern auch die latente Zuordnungsvariable $\mathbf{Z}$ als bekannt betrachtet werden, ergibt sich folgende Funktion, welche auch als *complete* Loglikelihood-Funktion bezeichnet wird:

$$\log L(\Theta|\mathbf{X}, \mathbf{Z}) = \sum_{i=1}^{n} \log \left[\pi_{g_i} f_N(\mathbf{x}_i|\boldsymbol{\mu}_{g_i}, \boldsymbol{\Sigma}_{g_i})\right]. \tag{4.5}$$

g_i entspricht dabei der Stelle, an der $\mathbf{z}_i$ einen Einser aufweist. Dies bedeutet also, dass $g_i \in \{1, \ldots, K\}$ ist und angibt, aus welcher Komponente die i-te Beobachtung stammt.

Vergleicht man nun diese Loglikelihood-Funktion mit jener aus Gleichung 4.4, kann man sehen, dass durch die zusätzliche Kenntnis von $\mathbf{Z}$ die Summe im Logarithmus wegfällt und sich somit die Maximierung dieser Funktion (Gleichung 4.5) leichter gestalten lässt.

Logischerweise ist das wahre $\mathbf{Z}$ jedoch nicht bekannt und muss deswegen aus den Daten geschätzt werden. Dies geschieht im sogenannten *Expectation-Step* (E-Step).

In den folgenden 2 Kapiteln werden der

- Expectation-Step: Berechnung des Erwartungswertes von $\mathbf{Z}$ und der
- Maximization-Step: Maximierung der Parameter der Klassenverteilungen

näher beschrieben.

4.3.1 E-Step

Möchte man nun die Verteilung der $\mathbf{z}_i$ berechnen, braucht man dafür zuerst die unbekannten Parameter Θ der normalverteilten Komponenten aus Gleichung 4.2. Das bedeutet, man nimmt für den Expectation-Step des EM-Algorithmus an, dass diese Parameter zu einem Zeitpunkt t (des iterativen Verfahrens) bereits gegeben sind:

$$\Theta^{(t)} = (\boldsymbol{\pi}^{(t)}, \boldsymbol{\mu}_1^{(t)}, \ldots, \boldsymbol{\mu}_K^{(t)}, \boldsymbol{\Sigma}_1^{(t)}, \ldots, \boldsymbol{\Sigma}_K^{(t)}) \tag{4.6}$$

Für die Berechnung der Parameter im ersten Schritt des Algorithmus werden die Beobachtungen zufällig einer Komponente zugeordnet, und danach die Mittelwerte und die Varianz-Kovarianz-Matrizen der K Komponenten berechnet.

Sind die Parameter gegeben, kann die Verteilung von $\mathbf{Z}$ ganz einfach mit der Bayesregel berechnet werden.

$$p(z_{ik} = 1|\mathbf{x}_i, \Theta^{(t)}) = \frac{\pi_k^{(t)} f_N(\mathbf{x}_i|\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{p(\mathbf{x}_i|\Theta^{(t)})} = \frac{\pi_k^{(t)} f_N(\mathbf{x}_i|\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{h=1}^{K} \pi_h^{(t)} f_N(\mathbf{x}_i|\boldsymbol{\mu}_h^{(t)}, \boldsymbol{\Sigma}_h^{(t)})} = \tau_{ik}$$

τ_{ik} entspricht nun der Wahrscheinlichkeit, dass die i-te Beobachtung aus der k-ten Komponente stammt und ist somit der Erwartungswert von z_{ik}, gegeben die Parameter $\Theta^{(t)}$ und die Daten $\mathbf{x}_i$.

Ist die Matrix $\boldsymbol{\mathcal{T}} = (\tau_{ik}),\ i = 1, \ldots, n,$ und $k = 1, \ldots, K$, der Zuordnungswahrscheinlichkeiten gegeben, können nun die Parameter $\Theta^{(t+1)}$ im anschließenden *Maximization-Step* berechnet werden.

4.3.2 M-Step

Im Maximization-Step wird der Erwartungswert der complete Loglikelihood-Funktion aus Gleichung 4.5, gegeben die Daten $\mathbf{X}$, die latente Zuordnungsvariable $\mathbf{Z}$ und die zum Zeitpunkt t gegebenen Parameter $\Theta^{(t)}$ aus Gleichung 4.6, maximiert. Dieser bedingte Erwartungswert wird als Q-Funktion bezeichnet.

Nach einigen Rechenschritten (s. Bilmes (1998)) bekommt man folgende vereinfachte Formel für die Q-Funktion:

$$\begin{aligned} Q(\Theta, \Theta^{(t)}) &= E_{\mathbf{Z}|\mathbf{X},\Theta^{(t)}} \left[\log L(\Theta|\mathbf{X}, \mathbf{Z})\right] \\ &= \sum_{k=1}^{K} \sum_{i=1}^{n} \tau_{ik}^{(t)} \log(\pi_k^{(t)} f_N(\mathbf{x}_i|\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})) \end{aligned}$$

Maximiert man nun die Q-Funktion erhält man die neuen Schätzer $\Theta^{(t+1)}|\Theta^{(t)}$:

$$\Theta^{(t+1)} = \underset{\Theta}{argmax}\ Q(\Theta, \Theta^{(t)}),$$

Diese sind:

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \tau_{ik}^{(t)} \mathbf{x}_i}{\sum_{i=1}^{n} \tau_{ik}^{(t)}}, \tag{4.7}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \tau_{ik}^{(t)} (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)})^T}{\sum_{i=1}^{n} \tau_{ik}^{(t)}}, \tag{4.8}$$

$$\pi_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \tau_{ik}^{(t)}. \tag{4.9}$$

Wie man sieht, werden für die Berechnung der Schätzer der Parameter lediglich die Zuordnungswahrscheinlichkeiten $\tau_{ik}^{(t)}$ des vorangegangen E-Steps verwendet.
Diese neu berechneten Parameter werden anschließend verwendet um neue Zuordnungswahrscheinlichkeiten $\tau_{ik}^{(t+1)}$ zu erhalten, mit welchen der Algorithmus wieder von vorne beginnt.

Zusammenfassend kann man sagen, dass der E-Step den Erwartungswert von $\mathbf{Z}$ auf Basis der derzeitigen Parameter $\Theta^{(t)}$ berechnet (*Expectation*), um danach mit diesem Erwartungswert $\boldsymbol{\mathcal{T}}$ die Q-Funktion zu maximieren. (*Maximization*)

Jede Iteration erzielt eine Erhöhung der Loglikelihood-Funktion und konvergiert unter schwachen Bedingungen gegen ein lokales Maximum.
Der Algorithmus wird solange durchgeführt, bis

$$Q(\Theta, \Theta^{(t+1)}) \leq Q(\Theta, \Theta^{(t)}) + \epsilon,$$

ist, wobei ϵ einer Genauigkeitsschwelle entspricht.

4.4 Umsetzung in R

Das modellbasierte Clustern ist in R im Package `mclust` implementiert und kann mit den Befehlen

```
> library(mclust)
> Mclust(data, G)
```

ausgeführt werden. `G` entspricht dabei der Anzahl der Komponenten, für die ein Modell angepasst werden soll (vgl. Fraley et al. (2012)).

In der Funktion `Mclust` werden jedoch nicht nur Modelle mit verschiedener Anzahl von Komponenten berechnet und verglichen, sondern sie vergleicht auch für jede Komponentenanzahl verschiedene Parametrisierungen der Varianz-Kovarianz-Matrizen $\boldsymbol{\Sigma}_k$ aus Gleichung 4.8.
Durch diese Vorgehensweise soll ein Modell gefunden werden, welches die Klassenstruktur der Daten möglichst gut und mit möglichst wenigen Parametern erklärt.

Die Parametrisierung der Varianz-Kovarianz-Matrizen $\boldsymbol{\Sigma}_k$ erfolgt mittels Eigenwertzerlegung der Form

$$\boldsymbol{\Sigma}_k = \lambda_k D_k A_k D_k^T,$$

wobei D_k eine orthogonale Matrix von Eigenvektoren und A_k eine Diagonalmatrix mit Diagonalelementen proportional zu den Eigenwerten ist, und λ_k dem größten Eigenwert von $\boldsymbol{\Sigma}_k$ entspricht. D_k, A_k und λ_k werden dabei als unabhängige Parameter betrachtet, welche so festgelegt sind, dass sie entweder für alle Komponenten gleich oder unterschiedlich sind (vgl. Fraley & Raftery (2007)). D_k, A_k und λ_k entsprechen den geometrischen

Eigenschaften der Verteilung der Gruppe k. λ_k entspricht dabei dem Volumen, A_k der Gestalt und D_k der Ausrichtung der Klassenverteilung.
Die Funktion `Mclust` bietet derzeit 10 Arten der Parametrisierung von $\mathbf{\Sigma}_k$ für den multivariaten Fall an, welche in der Tabelle 4.1 ersichtlich sind. So wird z.B. beim Modell „`EVI`“ angenommen, dass die Verteilungen der Komponenten das gleiche Volumen, jedoch verschiedene Ausdehungen (Varianzen) in den einzelnen Dimensionen haben und dass die Ellipsen entlang der Koordinatenachsen ausgerichtet sind.

Kürzel	Modell	Volumen	Gestalt	Ausrichtung
`EII`	λI	gleich	gleich	
`VII`	$\lambda_k I$	verschieden	gleich	
`EEI`	λA	gleich	gleich	Koordinatenachsen
`VEI`	$\lambda_k A$	verschieden	gleich	Koordinatenachsen
`EVI`	λA_k	gleich	verschieden	Koordinatenachsen
`VVI`	$\lambda_k A_k$	verschieden	verschieden	Koordinatenachsen
`EEE`	$\lambda D A D^T$	gleich	gleich	gleich
`EEV`	$\lambda D_k A D_k^T$	gleich	gleich	verschieden
`VEV`	$\lambda_k D_k A D_k^T$	verschieden	gleich	verschieden
`VVV`	$\lambda_k D_k A_k D_k^T$	verschieden	verschieden	verschieden

Tabelle 4.1: Parametrisierungen der Varianz in `Mclust`

4.5 Modellauswahl

Wie auch schon beim k-Means-Verfahren (s. Kapitel 3.3), wo die Anzahl der Klassen gegeben sein muss, ist auch beim modellbasierten Clustern die Anzahl der Komponenten ein Input-Parameter. Somit müssen auch hier wieder Modelle für verschiedene Anzahlen an Komponenten erstellt werden.
Doch anders als beim k-Means-Verfahren, muss man die Lösungen nicht subjektiv anhand eines Silhouettenplots bewerten, sondern kann, da es sich um ein Modell handelt, das Bayessche Informationskriterium (BIC) zur Beurteilung der Clusterlösung verwenden:

$$BIC = 2\ \log L_M(\mathbf{X}|\Theta^*) - p_M\ \log(n)$$

$\log L_M$ entspricht der maximierten Loglikelihood-Funktion des Modells M, p_M ist die Anzahl der zu schätzenden Parameter (s. Tabelle 4.2) von M und n die Anzahl der Beobachtungen.

Modellkürzel	# zu schätzende Parameter
`EII`	1
`VII`	G
`EEI`	d
`VEI`	$G + (d-1)$
`EVI`	$1 + G(d-1)$
`VVI`	Gd
`EEE`	$d(d+1)/2$
`EEV`	$1 + (d-1) + G[d(d-1)/2]$
`VEV`	$G + (d-1) + G[d(d-1)/2]$
`VVV`	$G[d(d+1)/2]$

Tabelle 4.2: Anzahl der zu schätzenden Parameter der Kovarianzmatrix bei verschiedenen Varianzparametrisierungen in `Mclust` (G = Anzahl der Komponenten, d = Anzahl der Variablen)

Unsicherheit

Hat man aufgrund des BIC ein Modell ausgewählt, kann mithilfe des Unsicherheits-Plot die Clusterlösung beurteilt werden. Gegeben die Matrix $\boldsymbol{\mathcal{T}}$ der Zuordnungswahrscheinlichkeiten, kann nun die Unsicherheit dieser Lösung berechnet werden. Für eine Beobachtung i mit dem Zuordnungsvektor $\boldsymbol{\tau}_i$ sei $\tau_{ij} = max\ \boldsymbol{\tau}_i$. Das bedeutet, dass die i-te Beobachtung mit einer Sicherheit τ_{ij} aus der j-ten Komponente stammt.
Dann ist die Unsicherheit (u_i) der i-ten Beobachtung in der j-ten Klasse zu sein, definiert durch

$$u_i = 1 - \tau_{ij} = 1 - max\ \boldsymbol{\tau}_i.$$

Diese Unsicherheit, welche möglichst nahe bei 0 liegen sollte, wird für alle n Beobachtungen berechnet, anschließend der Größe nach geordnet und in einem Linienplot eingezeichnet.

Kapitel 5

Ergebnisse

In diesem Kapitel werden nun die Hauptergebnisse dargestellt, wenn man den Teildatensatz `mat92` (s. Kapitel 2.2) mithilfe der in den Kapiteln 3 und 4 erklärten Clusterverfahren clustert.
Bei allen Ergebnissen der distanzbasierten Clusterverfahren, außer in Kapitel 5.1, wurde das Euklidische Distanzmaß verwendet, da dieses die besten Clusterlösungen lieferte.

5.1 k-Means-Verfahren

Wendet man das k-means-Verfahren auf den Datensatz `mat92` an, teilt sich die gesamte quadratische Abweichung (SS) von 3817.06 je nach Anzahl der Cluster wie folgt auf die Abweichung innerhalb der Cluster (WSS) und zwischen den Clustern (BSS) auf:

K	2	3	4	5	6	7	8
WSS	2399.14	1895.57	1693.86	1561.62	1426.92	1308.41	1217.6
BSS	1417.92	1921.49	2123.19	2255.44	2390.13	2508.64	2599.45

Tabelle 5.1: k-Means: Aufteilung der gesamten Quadratischen Abweichung bzgl. der Klassen, K = Anzahl der Klassen

Mit steigender Anzahl der Klassen wird die WSS immer kleiner und BSS immer größer.

Diese Werte geben allerdings keinen Aufschluss darüber, wie gut eine Clusterlösung für die jeweilige Anzahl der Klassen ist bzw. ob das k-means-Verfahren generell ein gutes Clusterergebnis für diesen Datensatz liefert. Deswegen betrachtet man den im Kapitel 3.5 besprochenen durchschnittlichen Silhouettenwert je Anzahl der Cluster.

Abbildung 5.1 liefert eine grafische Darstellung der durchschnittlichen Silhouettenwerte für das Euklidische- und das Manhattan-Distanzmaß, jeweils für 2 bis 8 Cluster. Den besten durchschnittlichen Silhouettenwert von 0.35 liefert dabei eine 2-Clusterlösung mit

dem euklidischen Distanzmaß.

Betrachtet man nun den Silhouettenplot dieser Clusterlösung (s. Abbildung 5.2), ist ein großer Cluster mit 104 Beobachtungen und ein kleiner Cluster mit 41 Beobachtungen zu erkennen.
Zeichnet man nun diese Clusterlösung in den Pairs-Plot aus Abbildung 2.2 ein, erhält man das in Abbildung 5.3 dargestellte Bild. Hier würde man eher sagen, dass die Lösung für die meisten Variablenkombinationen eher wilkürlich ist und man keine klare Struktur erkennen kann. Das bedeutet, es sollten in jedem Fall noch andere Clusterverfahren in Betracht gezogen werden.

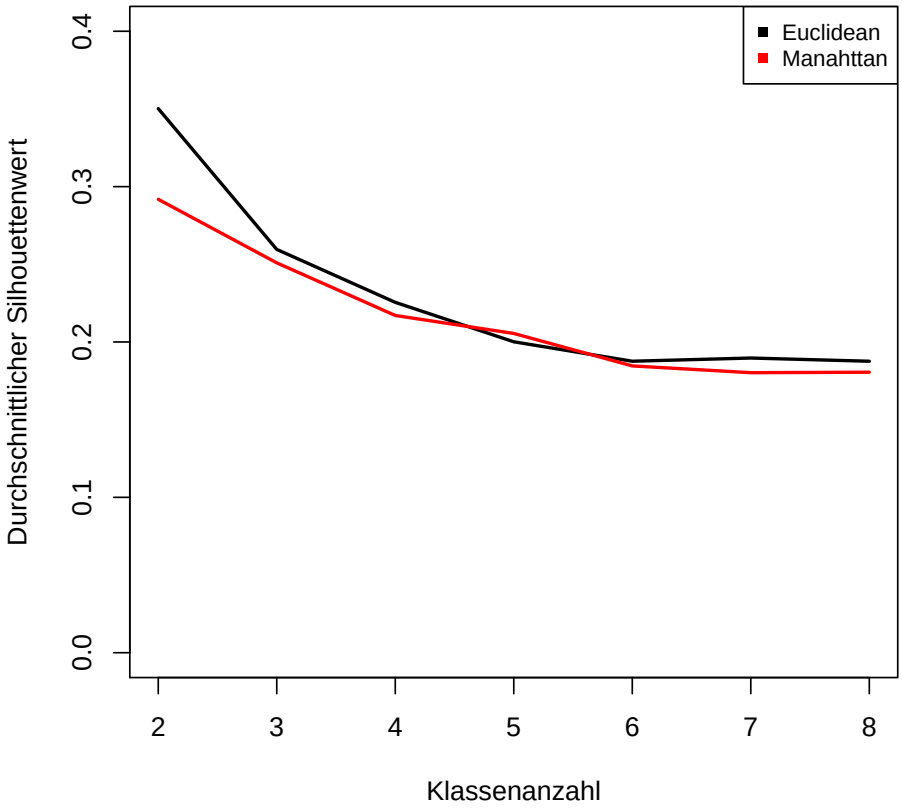


Abbildung 5.1: k-Means: Silhouettenwerte der Distanzmaße je Klasse

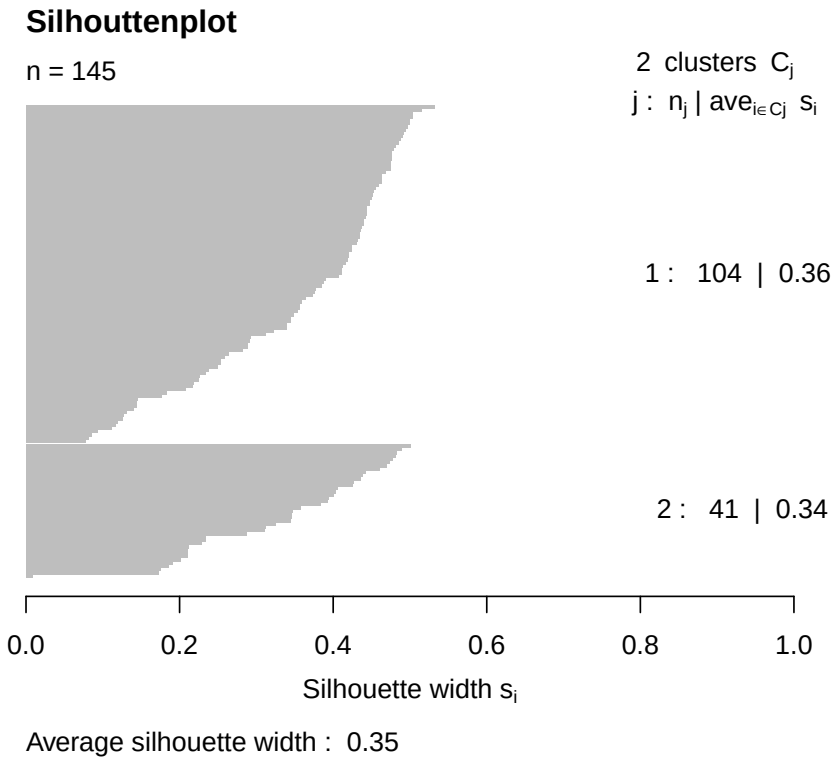


Abbildung 5.2: k-Means: Silhouettenplot - 2 Clusterlösung

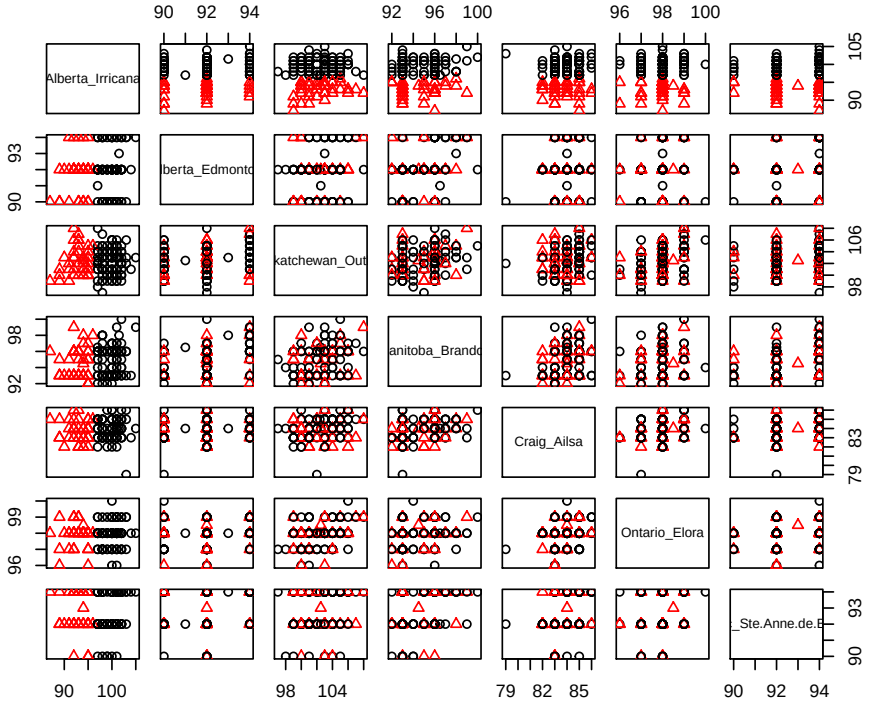


Abbildung 5.3: k-Means: Pairs-Plot - 2 Clusterlösung

5.2 Hierarchische Clusteranalyse

5.2.1 agglomeratives Verfahren

Wendet man das agglomerative hierarchische Clusterverfahren auf den Teildatensatz `mat92` mit verschiedenen Linkage-Methoden an, erhält man folgende durchschnittliche Silhouettenwerte für Clusterlösungen mit 2, 3 und 4 Cluster, die in Tabelle 5.2 angegeben sind.

	2 Cluster	3 Cluster	4 Cluster
Single-Linkage	0.19	0.10	0.08
Complete-Linkage	0.35	0.28	0.12
Average-Linkage	0.33	0.32	0.22

Tabelle 5.2: Agglomeratives Verfahren: Silhouettenwerte

Die besten 3 Silhouettenwerte ergeben sich bei einer 2-Clusterlösung mit der Complete-Linkage- (0.35) bzw. der Average-Linkage-Methode (0.33) und bei einer 3-Clusterlösung der Average-Linkage-Methode (0.32).

Betrachtet man nun den Silhouettenplot in Abbildung 5.4 (2-Clusterlösung für Complete-Linkage) ist eine Clusterstruktur zu erkennen, welche sehr ähnlich der Lösung des k-Means-Verfahrens ist. Das dazugehörige Dendrogramm (Abbildung 5.5) lässt auch auf eine 2-Clusterlösung schließen.

Die 2-Clusterlösung der Average-Linkage-Methode liefert allerdings ein Ergebnis, bei welchem sich nur eine Beobachtung in einem Cluster befindet, während sich die restlichen 144 Beobachtungen in einem zweiten Cluster zusammenschließen (s. Abbildung 5.6). Somit diese Clusterlösung trotz des relativ hohen Silhouettenwerts nicht interessant.
Interessant ist jedoch, dass die 3-Clusterlösung der Average-Linkage-Methode im wesentlichen der 2-Clusterlösung der Complete-Linkage-Methode gleichkommt, bei welcher zusätzlich eine Beobachtung einen dritten Cluster bildet (s. Abbildung 5.7).
Das Dendrogramm der Linkage-Methode (s. Abbildung 5.8) liefert allerdings keinen „eindeutigen“ Hinweis, so wie in Abbildung 5.5, auf die Anzahl der Cluster.

Somit deuten die Ergebnisse des agglomerativen hierarchischen Clusterverfahrens auf eine Clusterlösung mit 2 Clustern hin. Dabei ist bemerkenswert, dass nicht nur die endgültige Clusteranzahl des hierarchischen Verfahrens mit jener des k-Means-Verfahren übereinstimmt (s. Tabelle 5.3), sondern auch die Einteilung der 145 Beobachtungen in diese Cluster bei beiden Verfahren fast ident ist.

Methode	Cluster 1	Cluster 2	Cluster 3
k-Means	104	41	
agglo. hier. Ver. - Complete-Linkage	108	37	
agglo. hier. Ver. - Average-Linkage	106	38	1

Tabelle 5.3: Vergleich von k-Means mit agglomerativen Verfahren

Der Pairs-Plot für die Lösung der Complete- bzw. Average-Linkage-Methode ist fast ident mit jener aus Abbildung 5.3. Somit wird an dieser Stelle auf die Darstellung verzichtet.

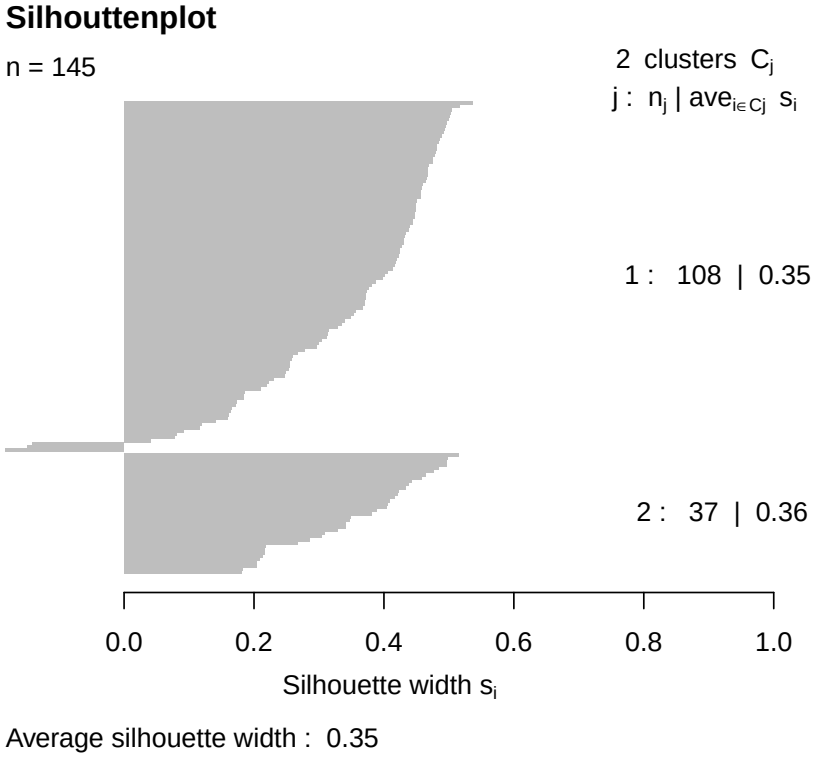


Abbildung 5.4: Complete-Linkage: Silhouettenplot - 2 Clusterlösung

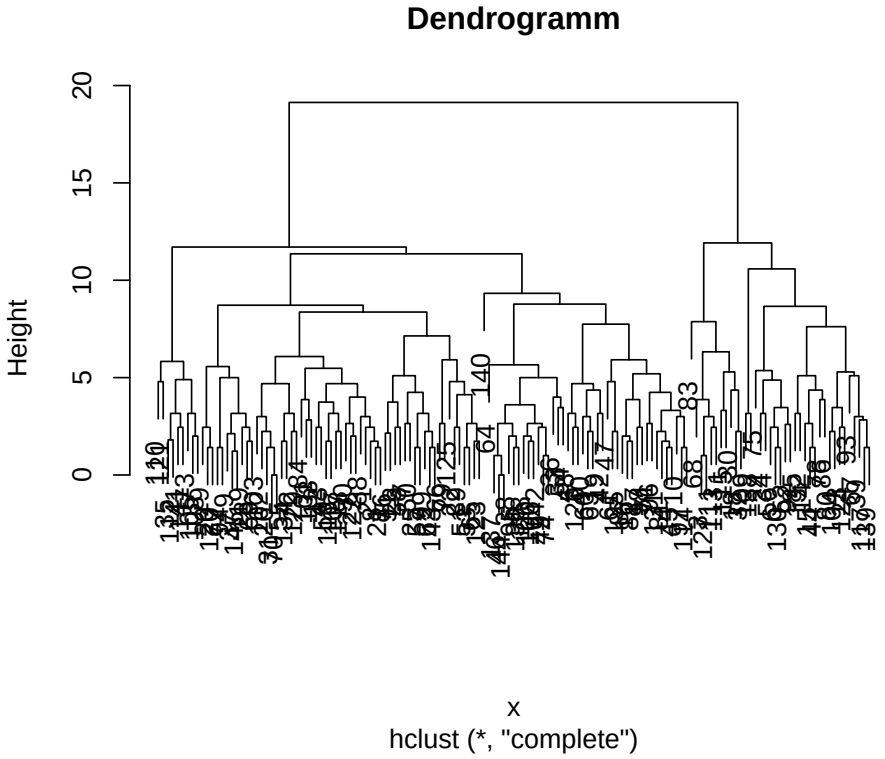


Abbildung 5.5: Complete-Linkage: Dendrogramm

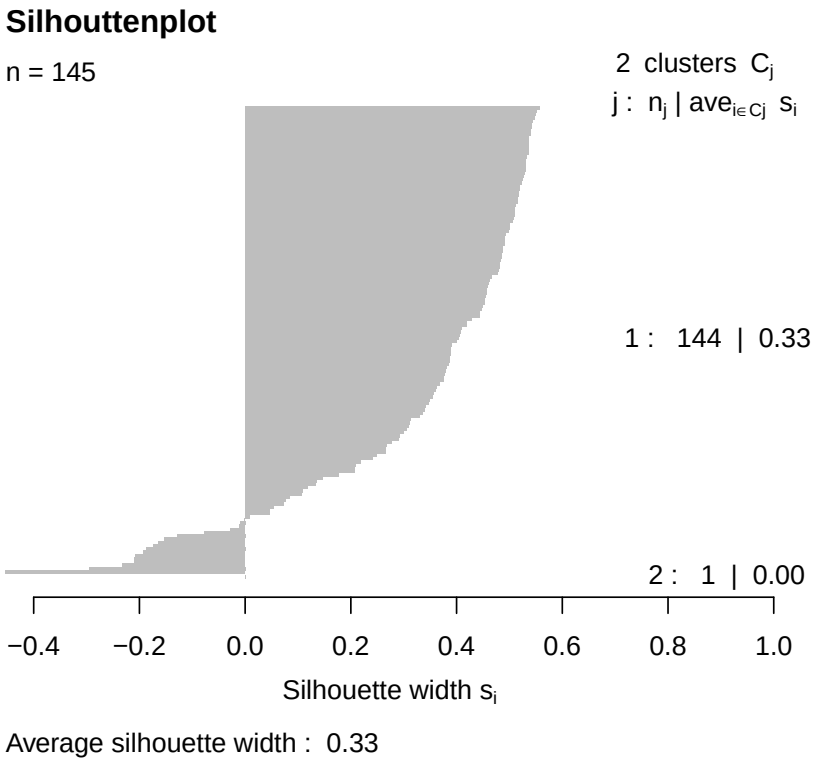


Abbildung 5.6: Average-Linkage: Silhouettenplot - 2 Clusterlösung

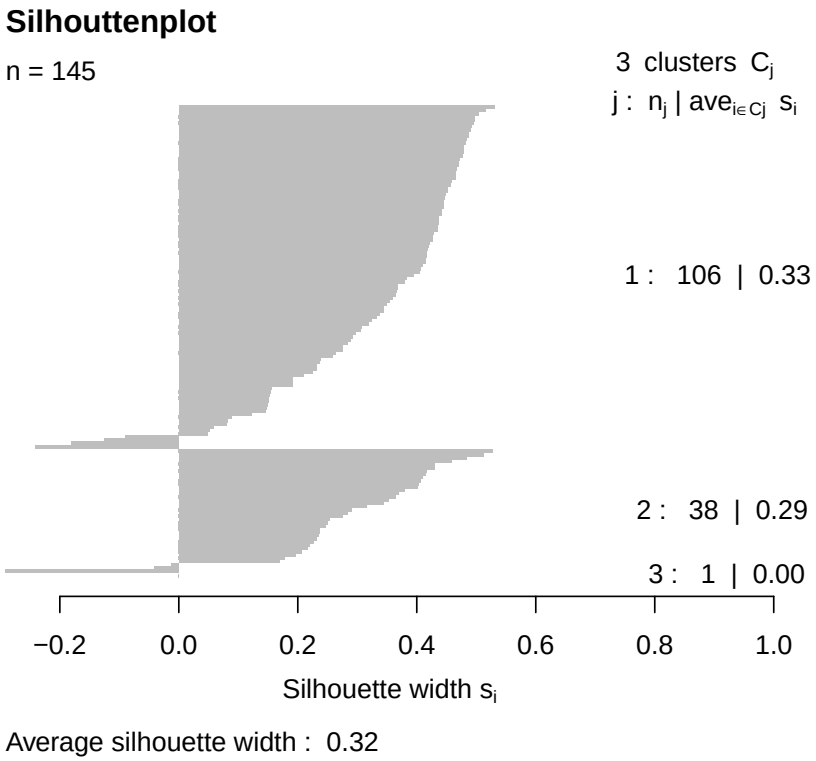


Abbildung 5.7: Average-Linkage: Silhouettenplot - 3 Clusterlösung

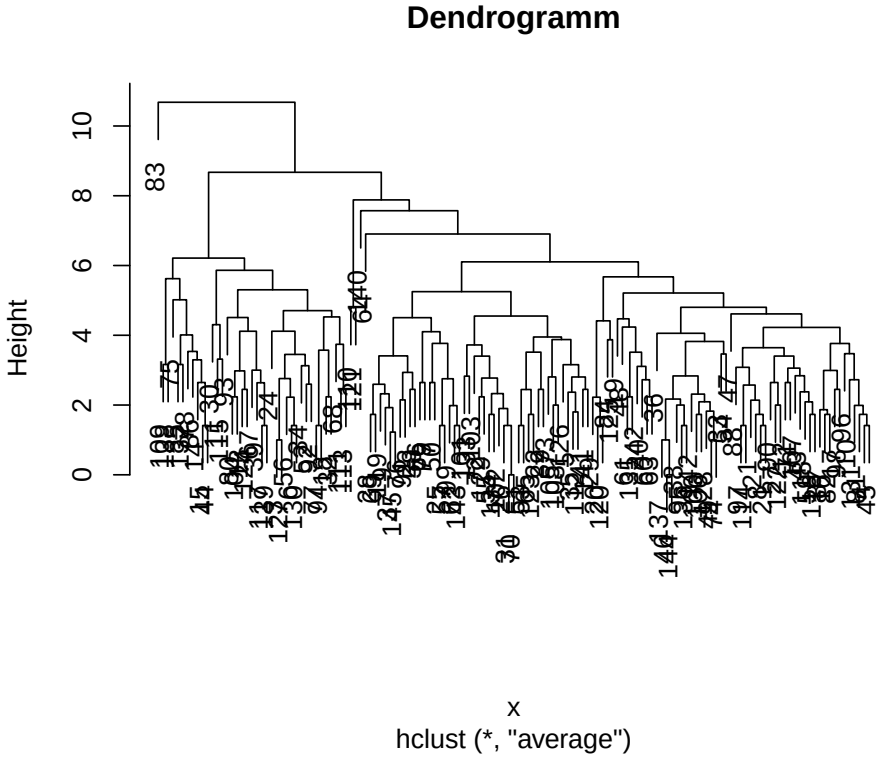


Abbildung 5.8: Average-Linkage: Dendrogramm

5.2.2 Divisives Verfahren

Wendet man das divisive Clusterverfahren auf den Datensatz an, deuten wiederum der durchschnittliche Silhouettenwert (s. Tabelle 5.4) und das Dendrogramm (s. Abbildung 5.9) auf eine Clusterlösung mit 2 Clustern hin.

	2 Cluster	3 Cluster	4 Cluster	5 Cluster
s(i)	0.35	0.27	0.22	0.21

Tabelle 5.4: Divisives Verfahren - Silhouettenwerte

Vergleicht man den Silhouettenplot von k-Means aus Abbildung 5.2 mit jenem des divisiven Verfahrens aus Abbildung 5.10 erkennt man, dass diese exakt die gleiche Clustergrößen liefern. Wie bereits im agglomerativen Fall ist nicht nur die Größe der Cluster dieser beiden Verfahren gleich, sondern auch die Einteilung der Beobachtungen in diese Cluster.
Das bedeutet, das divisive Verfahren liefert genau dasselbe Ergebnis wie schon das k-Means-Verfahren, nämlich einen großen Cluster mit 104 Beobachtungen und einen kleineren mit 41 Beobachtungen.

Da die Darstellung dieser Clusterlösung in einem Pairs-Plot mit jener aus Abbildung 5.3 übereinstimmt, wird an dieser Stelle darauf verzichtet.

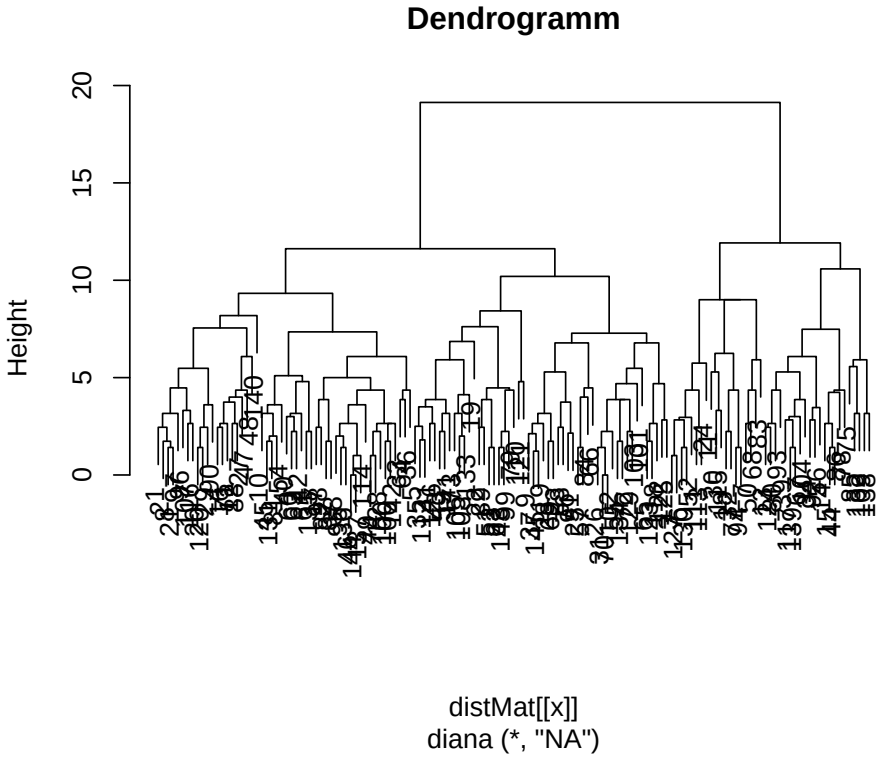


Abbildung 5.9: Divisives Verfahren: Dendrogramm

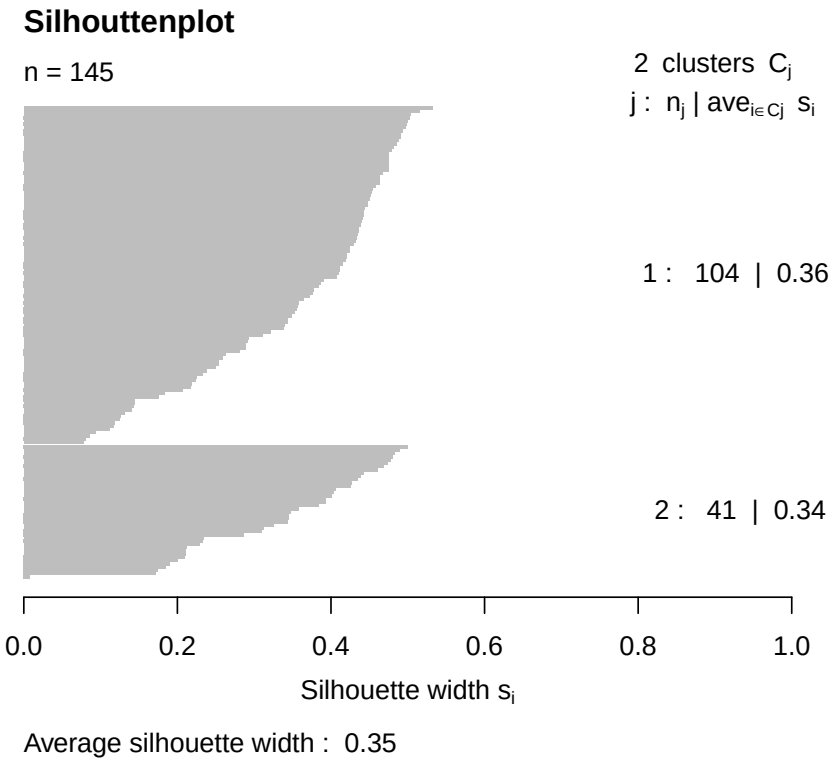


Abbildung 5.10: Divisives Verfahren: Silhouettenplot

5.3 Mclust

Verwendet man das im Kaptiel 4.4 beschriebene Paket `mclust` um `mata92` modellbasiert zu clustern, erhält man folgenden Output:

```
>   library(mclust)
>   Mcluster1 <- Mclust(data=mat92, G=1:8)
>   summary(Mcluster1)

----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust EVI (diagonal, equal volume, varying shape) model with 2 components:

 log.likelihood   n df       BIC       ICL
      -1791.798 145 28 -3722.944 -3748.725

Clustering table:
 1  2
67 78
```

`Mclust` hat nun Mischverteilungen mit verschiedener Anzahl an Komponenten (1 bis 8) und verschieden Parametrisierungen für $\mathbf{\Sigma}_k$ (s. Tabelle 4.1) an den Datensatz angepasst und anhand des BIC verglichen. Die BIC-Werte all dieser Modelle können in Abbildung 5.11 betrachtet werden.
Sowohl am Output als auch am BIC-Plot in Abbildung 5.11 kann abgelesen werden, dass ein `EVI`-Modell ($\oplus$) mit 2 Kompententen ausgewählt wird. `EVI` bedeutet, dass die 2 Komponenten das gleiche Volumen haben (`E..`), verschiedene Formen aufweisen (`.V.`) und entlang der Koordinatenachsen ausgerichtet sind (`..I`). Die beiden entstandenen Cluster bestehen dabei aus 67 bzw. 78 Beobachtungen.
Interessant ist, dass `Mclust` genauso wie das „Clustern ohne Modellannahme“ 2 Cluster in den Daten schätzt, jedoch unterscheiden sich die Größen der beiden Clusterlösungen.

Sowohl in Abbildung 5.12 als auch in Abbildung 5.13 befindet sich ein Pairs-Plot von `mat92`, in welchem die von `Mclust` erzeugten Cluster eingezeichnet sind. In Abbildung 5.12 wurden zusätzlich die Komponentenverteilungen grau eingezeichnet.
Vergleicht man die Klassifizierung mit der des k-Means-Verfahrens im Pairs-Plot in Abbildung 5.3, fällt auf, dass die Clustereinteilung hier weniger willkürlich wirkt. Bei Variablenkombinationen, bei welchen man eine Clusterstruktur mit 2-3 Clustern vermuten

würde (z.B. zwischen den Variablen 1 und 7), werden zumindest 2 gefunden.

Um einen Überblick der gefundenen Clusterlösung zu bekommen, wurden in Abbildung 5.14 die Histogramme der phänotypischen Daten, für die 2 gefunden Gruppen (Spalten), für alle 7 Variablen von `mat92` (Zeilen) dargestellt. In allen Umgebungen kamen in Gruppe 1 größere Werte häufiger vor, d.h. die Pflanzen in Gruppe 1 brauchten durchschnittlich länger bis zur Reife.

In Abbildung 5.15 wurde die Unsicherheit (s. Kapitel 4.5) der Beobachtungen geplottet. Anzumerken ist hierbei, dass 102 Beobachtungen eine Unsicherheit kleiner 0.05 (gestrichelte Linie) aufweisen. Das heißt also, man ist sich bei 70% der Daten mindestens zu 95% sicher, zu welchem Cluster sie gehören.

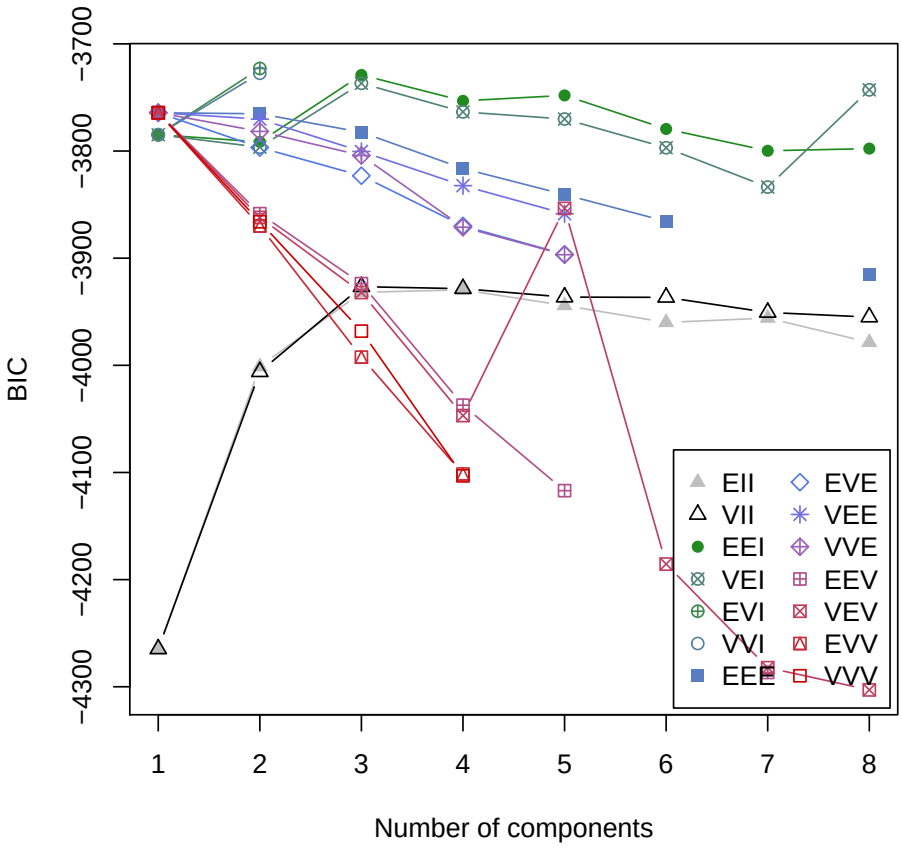


Abbildung 5.11: Mclust: BIC-Plot

```
>   plot(Mcluster1, what="classification")
```

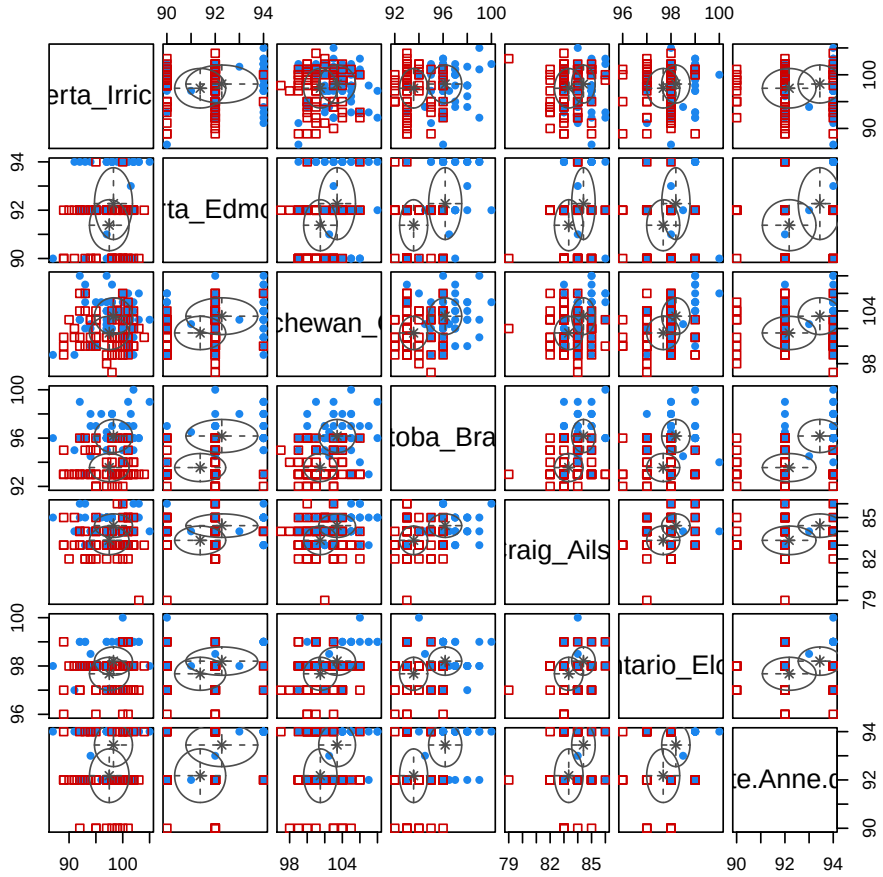


Abbildung 5.12: Mclust: Klassifikation-Plot

```
>   pairs(mat92, col=Mcluster1$classification, pch=Mcluster1$classification)
```

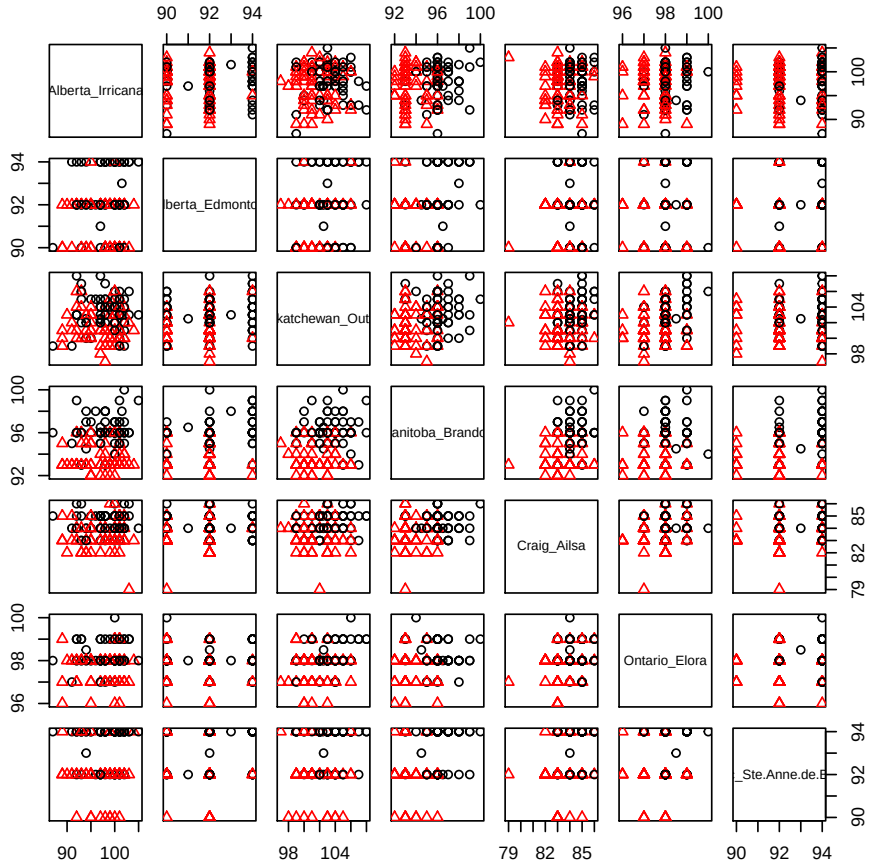


Abbildung 5.13: Mclust: Pairs-Plot

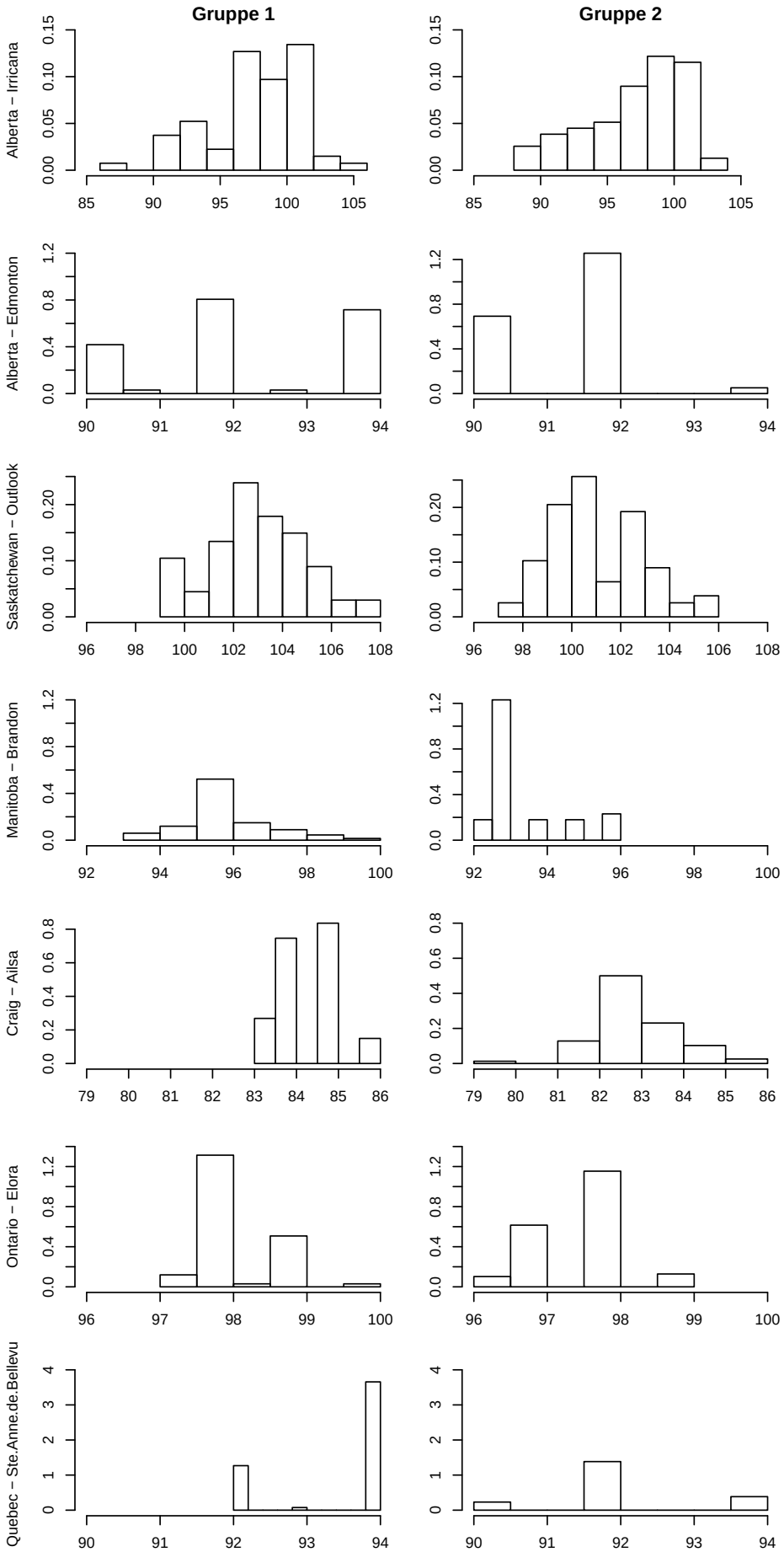


Abbildung 5.14: Vergleich zw. Gruppen der Mclust-Lösung mittels Histogramme

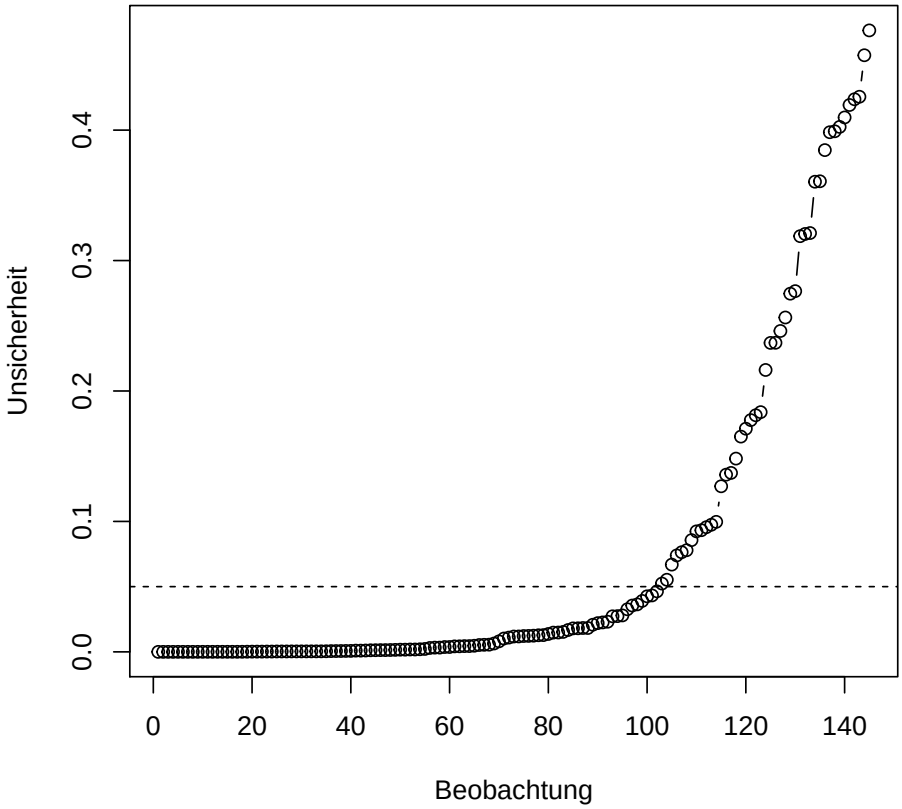


Abbildung 5.15: Mclust: Unsicherheitsplot

5.4 Zusammenfassung der Ergebnisse

In Tabelle 5.5 werden die Clusterlösungen aller gezeigten Verfahren zusammengefasst.

Methode	Cluster 1	Cluster 2	Cluster 3
distanzbasiertes Clustern			
k-Means	104	41	
agglo. Verfahren - Complete	108	37	
agglo. Verfahren - Average	106	38	1
div. Hier.	104	41	
modellbasiertes Clustern			
Mclust	78	67	

Tabelle 5.5: Überblick aller Clusterlösungen

In dieser Tabelle wird noch einmal deutlich, dass die distanzbasierten Clusterverfahren alle sehr ähnliche Lösungen liefern, während das modellbasierte Clusterverfahren zwar auch 2 Cluster schätzt, aber eine andere Zuordnung der Beochbachtungen in diese Cluster trifft.

Die entscheidende Frage, welche sich jetzt stellt ist, „Welche Rückschlüsse können nun aufgrund dieser Clusterlösungen auf den genotypischen Datenteil getroffen werden?“, d.h. „Unterscheiden sich die Genmarker zwischen den gefundenen Gruppen?“

Kapitel 6

Rückschlüsse auf den genotypischen Datensatz

In diesem Kapitel werden nun aufgrund der erzeugten Cluster des k-Means-Verfahrens (stellvertretend für alle besprochenen distanzbasierten Clusterverfahren) und des modellbasierten Clusterverfahrens (`Mclust`), Aussagen über den genotypischen Datensatz getroffen.

Zuerst wird der genotypsche Datenteil des Datensatzes `mat92` eingelesen. Danach werden die fehlenden Werte, welche einen Anteil von ca. 5% aufweisen, mit zufällig erzeugten Werten imputiert und die Ausprägung -1 durch 0 ersetzt. Dadurch entspricht der Anteil er Ausprägung 1 dem Mittelwert. Dieser neu erzeugte Datensatz wird von nun an als `gen0` bezeichnet.

Anschließend wird der Datensatz in 2 Teildatensätze `gen1` und `gen2` auf Basis der im phänotypischen Datenteil erzeugten Cluster aufgeteilt.

Danach wurde mithilfe eines χ^2-Tests für jeden Genmarker getestet, ob sich der Anteil der Ausprägung 1 zwischen den Clustern signifikant unterscheidet. Da dieser Test für jeden Genmarker angewendet wird, handelt es sich um multiples Testen.
Durch diese Vielzahl an Tests entsteht das Problem, dass der globale α-Fehler nicht mehr gleich α sondern um ein Vielfaches größer ist. Zur Adjustierung der p-Werte wird im folgenden die Benjamini & Hochberg Korrektur verwendet.

Benjamini & Hochberg Korrektur
Die Korrektur laut Benjamini & Hochberg (1995) kontrolliert die „False Discovery Rate“ (FDR). Dies bedeutet, dass der erwartete Anteil der fälschlich abgelehnten Hypothesen an allen abgelehnten kontrolliert wird. Der Grund, warum diese Korrektur in dieser Arbeit

anderen Prozeduren, wie z.B der Bonferroni-Korrektur, vorgezogen wird, ist der, dass sie eine größere Power besitzt und weniger strikt ist. Diese Korrektur wird beim Testen von signifikanten Genmarkern oft verwendet, damit keine möglichen Ansatzpunkte (Marker) übersehen werden.

Vorgehen
Benjamini & Hochberg schlagen vor, die p-Werte zuerst der Größe nach aufsteigend zu ordnen, um p_j^* $(j = 1, \ldots, N_p)$ zu erhalten. Man suche dann das größte j, für das

$$p_j^* < \alpha \frac{j}{N_p}$$

gilt.
Sei p_s^* der signifikante p-Wert mit dem größten j. Dann gelten alle p_l^*, für die $l < s$ gilt, als signifikant.
Wendet man diese Korrektur auf die 127 p-Werte an, bekommt man, je nach gewünschtem globalen α-Level und angewendetem Clusterverfahren, die in Tabelle 6.1 dargestellte Anzahl an signifikanten Genmarkern.

	Globaler Fehler	
	$\alpha = 0.1$	$\alpha = 0.05$
k-Means-Verfahren	0	0
modellb. Clustern (`Mclust`)	22	8

Tabelle 6.1: Anzahl der signifikanten Genmarker

Interessanterweise sind bei den distanzbasierten Clusterverfahren (k-Means) keine signifikanten Unterschiede in der Anzahl der Genmarker zwischen den Gruppen zu beobachten. Beim modellbasierten Verfahren waren jedoch 22 Unterschiede und somit die entsprechenden Genmarker signifikant zum Niveau $\alpha = 0.1$ und sogar 8 signifikant zum Niveau $\alpha = 0.05$. In Tabelle 7.1 sind die Genmarker, welche durch ein modellbasiertes Clusterverfahren (`Mclust`) gefunden wurden, aufgelistet.

In den Abbildungen 6.1 und 6.2 sind die Differenzen der Anteile der Ausprägung 1 je Marker zwischen den Clustern für beide Clusterlösungen, k-Means und `Mclust`, anhand von Barplots dargestellt. Die roten und orangen Balken symbolisieren dabei die signifikanten Unterschiede, während die grauen Balken nichtsignifikante Unterschiede bedeuten. Die Tatsache, dass die Balken bei der Mclust-Klassifikation (Abbildung 6.2) höher sind als jene der k-Means-Klassifikation (Abbildung 6.1), lässt zumindest mehr signifikante Unterschiede vermuten.

Nun bleibt die Frage bestehen ob ein Regressionsverfahren die gleichen Marker als signifikant bestimmen würde.

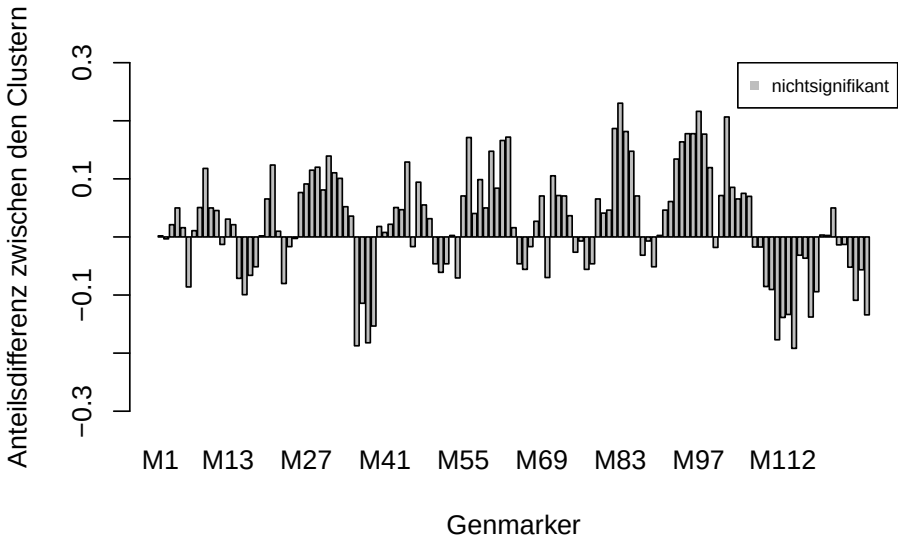


Abbildung 6.1: Barplot der Anteilsunterschiede für k-Means-Klassifikation

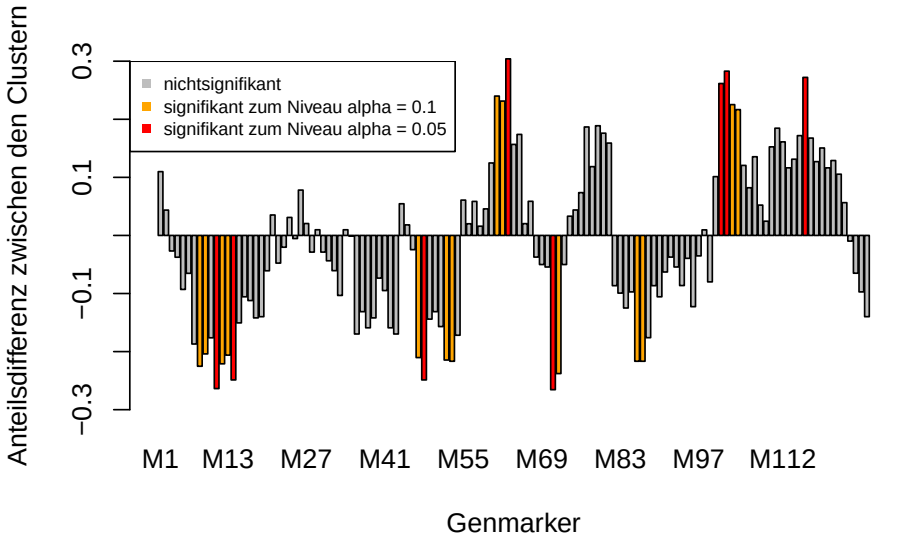


Abbildung 6.2: Barplot der Anteilsunterschiede für Mclust-Klassifikation

Kapitel 7

Vergleich mit LASSO-Regression

Da das Hauptaugenmerk dieser Arbeit auf diversen Clusterverfahren liegt, wird an dieser Stelle die LASSO-Regression nur oberflächlich behandelt. Eine detailliertere Beschreibung kann in Hastie et al. (2008) nachgeschlagen werden.

7.1 Theorie

Bei der penalisierten Regression geht es genau wie bei der normalen Regression darum die Koeffizienten der Kovariablen zu schätzen. Der Unterschied besteht darin, dass die Koeffizientenschätzungen mit einem Penalty-Term versehen werden. Durch den Penalty-Term sollen kleine β-Koeffizienten zu 0 „gezogen“ werden. Bei der LASSO-Regression sind die Koeffizientenschätzer definiert durch

$$\begin{aligned} \hat{\beta}^{lasso} &= \underset{\beta}{argmin} \left[\frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right] \\ &= \underset{\beta}{argmin} \left[\frac{1}{2} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1 \right]. \end{aligned} \tag{7.1}$$

Je nachdem wie der Penalty-Parameter λ gewählt wird, erhält man so eine gewisse Anzahl an Kovariablen, deren Effekte ungleich 0 sind. Je größer λ gewählt wird, desto weniger Koeffizienten sind von Null verschieden. Das optimale λ wird bei der Penalty-Regression mittels Kreuzvalidierung ermittelt.

7.2 Ergebnisse

Es wurde die Funktion `glmnet` aus dem gleichnamigen Paket in `R` verwendet, um mit allen 7 Umgebungen von `mat92` als abhängige Variablen eine multivariate LASSO-Regression (s. auch Xu (2007)) durchzuführen.

Um die Koeffizientenschätzer der multivariaten LASSO-Regression mit der Clusterlösung von `Mclust` vergleichen zu können, wird hierbei nicht das „beste“ λ gesucht, sondern jenes, bei welchem ungefähr gleich viele Genmarker Effekte aufweisen, wie sie zuvor in der Clusteranalyse gefunden wurden.
Um genau 9 von 0 verschiedene Regressionen zu erhalten, muss $\lambda = 0.8041$ gewählt werden, für 22 von 0 verschiedene Koeffizienten ist $\lambda = 0.5816$. Die gefundenen Genmarker für beide λ sind in Tabelle 7.1 dargestellt.

Vergleicht man nun die mit LASSO-Regression gefundenen Genmarker mit denen welche vom modellbasierten Clustern gefunden wurden (s. Abbildung 6.2), kommt man zu dem Ergebnis, dass 2 der 9 bzw. 8 der 22 auch durch das Anwenden einer Clusteranalyse als signifikant zum Niveau $\alpha = 0.05$ bzw. $\alpha = 0.1$ eingestuft wurden (s. Tabelle 7.1).

	χ^2-Test der Anteile		LASSO-Regression	
Genmarker	α =0.1	α =0.05	λ =0.5816	λ =0.8041
8	x			
9	x			
11	x	x	x	x
12	x			
13	x		x	
14	x	x	x	
26			x	
31			x	
36			x	x
47	x			
48	x	x	x	
51			x	
52	x		x	x
53	x			
61	x			
62	x		x	x
63	x	x	x	x
71	x	x		
72	x			
79			x	
83			x	
84			x	
86	x			
87	x			
94			x	
97			x	x
101	x	x		
102	x	x	x	
103	x			
104	x			
110			x	
111			x	x
112			x	x
116	x	x		
117			x	
127			x	x

Tabelle 7.1: Signifikante Genmarker

Kapitel 8

Zusammenfassung

In dieser Bachelorarbeit wurde versucht folgende Frage zu beantworten: „Welche genetischen Voraussetzungen müssen Gerstenpflanzen erfüllen, um je nach phänotypischer Eigenschaft, die besten Nachkommen zu erzeugen? “

Aufgrund der begrenzten Zeit für diese Arbeit, wurden jedoch nicht alle 7 zur Verfügung stehenden phänotypischen Eigenschaften analysiert, sondern nur eine. Diese Eigenschaft ist die Zeit einer Gerstenpflanze bis zu ihrer Reife in Tagen. Um die Genmarker, die mit dieser Reifezeit assoziiert sind, zu ermitteln, muss man sich Methoden überlegen, welche den Einfluss einzelner Genmarker auf die Reife der Pflanzen aufdeckt.

Dieser Einfluss bzw. Effekt der Genmarker wird üblicherweise mit einem Regressionmodell geschätzt. In dieser Arbeit ging es darum zu untersuchen, ob und wie effektiv diese Fragestellung auch mit einem Clusterverfahren lösbar ist. Hierfür wurden zwei Ansätze verwendet, einerseits das distanzbasierte Clustern (s. Kapitel 3) und andererseits ein modellbasiertes Clusterverfahren (s. Kapitel 4).
Die distanzbasierten Clusterverfahren lieferten dabei durchwegs Lösungen, welche für 2 Cluster in den Daten sprechen. Hierbei wurden immer ein großer Cluster mit ca. 104 Beobachtungen und ein kleiner Cluster mit 41 Beobachtungen (s. Kapitel 5.1 bis 5.2.2) gefunden. Das modellbasierte Clusterverfahren lieferte zwar die gleiche Clusteranzahl, jedoch wurde hier eine andere Gruppenzugehörigkeit der Beobachtungen gefunden (s. Kapitel 5.3). Der rein optische Vergleich mittels Pairs-Plots mit eingezeichneter Gruppenzugehörigkeit legte an dieser Stelle der Arbeit die Vermutung nahe, dass durch das Clustern mit Modellannahme die Clusterstruktur besser eingefangen werden konnte.

Anschließend wurden sowohl für das distanzbasierte Clusterverfahren (nur k-Means) als auch für das modellbasierte Clusterverfahren bei den gefundenen Gruppen untersucht, ob sich die Genmarker „wesentlich“ unterscheiden. D.h. jeder Genmarker wurde mittels

χ^2-Tests getestet und überprüft, ob sich die Anteile an „1“ und „-1“ , Kodierungen für das Harrington bzw. das TR306-Allel, zwischen den Gruppen unterscheiden.
Interessanterweise wurden hierbei nur bei den Clustern, die durch das modellbasierte Clustern gefunden wurden, signifikante Unterschiede festgestellt und somit einflussreiche Genmarker.

Die so gefundenen Genmarker wurden abschließend mit den Genmarkern verglichen, die mithilfe einer LASSO-Regression gefunden wurden, wobei jedoch relativ wenig Übereinstimmung gefunden wurde. Es gibt Genmarker, welche von beiden Varianten als signifikant eingestuft werden, die Mehrheit der von den Clusterverfahren gefundenen Genmarker unterscheidet sich jedoch von denen einer (LASSO-)Regression.

Man könnte die Arbeit so fortsetzen, dass zuerst analog zu dieser Arbeit auch noch die übrigen 6 Eigenschaften der Gerstenpflanzen einzeln oder gemeinsam geclustert und mit einem Regressionsmodell verglichen werden. Hierbei wäre es interessant zu überprüfen, wieviel gleiche Genmarker bei diesen Eigenschaften gefunden werden bzw. ob überhaupt eine Clusterstruktur vorliegt bzw. gefunden wird.
Es wäre auch interessant, umgekehrt die genetischen Marker zu clustern und zu untersuchen, ob sich die gefundenen Gruppen phänotypisch unterscheiden. Dabei müsste man sich überlegen wie man 127 binäre Variablen sinnvoll clustern kann.

Literaturverzeichnis

Benjamini, Y. & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B 57*, 289–300.

Bilmes, J. (1998). A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute Berkeley CA*.

Fahrmeir, F., Hamerle, A. & Tutz, G. (1996). *Multivariate statistische verfahren*.

Fraley, C. & Raftery, A. (2007). Model-based methods of classification: Using the mclust software in chemometrics. *Journal of Statistical Software*, *18*.

Fraley, C., Raftery, A., Murphy, T. & Scrucca, L. (2012). mclust version 4 for r: Normal mixture modeling for model-based clustering, classification, and density estimation. *Department of Statistics University of Washington*.

Graw, J. (2010). *Genetik*.

Hastie, T., Tibshirani, R. & Friedman, J. (2008). *The elements of statistical learning* (Bd. 2).

Mclachlan, M. & Peel, D. (2000). *Finite mixture models*.

Xu, S. (2007). An empirical bayes method for estimating epistatic effects of quantitative trait loci. *International Biometric Society*, *63*, 513–521.

Zhao, F. & Xu, S. (2012). Genotype by environment interaction of quantitative traits: A case study in barley. *Department of Botany and Plant Sciences*, *2*, 779–788.

Anhang

a) Funktionen

Funktion `phe2csv`

Funktion zur Umwandlung einer phe-Datei in eine csv-Datei und Abspeicherung dieser. Wird zum Einlesen und Speichern der phänotypischen Daten benötigt.

```
> phe2csv <- function(filename, category=TRUE){
+
+   if(!is.character(filename)){
+     stop("Argument filename has to be a character!")
+   }
+
+   scan_txt <- scan(filename, what=character(), quiet=TRUE)
+   junk <- grep("*progeny*", scan_txt)
+   scan_txt <- scan_txt[-(1:junk)]
+
+   new_var <- grep("\\*", scan_txt)
+   start_varname <- grep("{", scan_txt, perl=T)
+   end_varname <- grep("*}", scan_txt)
+   p <- length(new_var)
+
+   scan_txt[start_varname]
+   year <- sub("(,)","" , sub("(\\{*)","",scan_txt[start_varname]))
+   envir <- sub("(,)","",scan_txt[start_varname+1])
+   city <- sub("(,)","",scan_txt[start_varname+2])
+   land <- sub("(,)","" , sub("(*})","",scan_txt[end_varname]))
+
+   var_names <- cbind(land, city, envir, year)
+   var_names <- apply(var_names,1,function(x){
+     paste(x[1],"_",x[2],"_",x[3],"_",x[4], sep="")})
```

```
+
+   df <- matrix(NA, nrow=145, ncol=p)
+   colnames(df) <- var_names
+
+   new_var2 <- c(new_var[-1],length(scan_txt))
+   for(j in 1:p){
+     c <- as.numeric(scan_txt[(end_varname[j]+1):(new_var2[j]-1)])
+     df[1:length(c),j] <- c
+   }
+
+   df <- as.data.frame(df)
+
+   filename_save <- sub("(.phe)", ".csv", filename)
+   write.csv(df, file=filename_save, row.names=FALSE)
+
+   cat(filename_save,"saved in", getwd(),"\n")
+
+   if(category){
+     return(list(land=land,city=city,envir=envir,year=year))
+   }
+ }
```

Funktion `order_class`

Funktion zum „ordnen“ der Clusterlösungen.

```
>   order_class <- function(vec){
+       new_class <- as.numeric(rownames(sort(table(vec))))[length(table(vec)):1]
+       new_vec <- vector("numeric", length(table(vec)))
+
+       for(i in 1:length(table(vec))){
+         new_vec[vec==i] <- new_class[i]
+       }
+
+       new_vec
+   }
>   clust_euc_kmeans <- order_class(kmeans2_8[[1]]$cluster)
>   clust_euc_hclust_c <- order_class(cutree(c_hclust[[1]],2))
>   clust_euc_hclust_a <- order_class(cutree(a_hclust[[1]],3))
```

```
>   clust_euc_diana <- order_class(cutree(mat92_diana[[1]],2))
```

b) Einlesen und Bereinigen des Datensatzes

Datensatz maturity wird eingelesen. Zusätzlich werden die mit -9999 eingetragenen fehlenden Werte durch NA ersetzt.

```
>   library(xtable)
>   mat <- read.csv("CSV_Data//mat.csv")
>   mat[mat==-9999] <- NA
>   #3 NA's enthalten
>
>   mat_cat <- read.csv("CSV_Data//category_mat.csv")
>   mat_cat$city <- as.character(mat_cat$city)
>   mat_cat$envir <- as.character(mat_cat$envir)
```

Bereinigung des Datensatzes.

```
>   #Nur die Daten aus dem Jahre 1992 werden ausgewählt.
>   mat92 <- mat[mat_cat$year==1992]
>   #1 NA enthalten
>
>   #Imputation mit Mittelwert der Spalte des fehlenden Wertes.
>   mat92[is.na(mat92[,3]),3] <- mean(na.omit(mat92[,3]))
>   #mat92 enthält nun 0 NA's
>
>   #Canada_ und _1992 der Variablennamen wird entfernt.
>   colnames(mat92) <- sub("Canada_","",sub("_1992","",colnames(mat92)))
```

Einlesen des genotypischen Datensatzes.

```
>   gen <- read.table("CSV_Data//gen_barley.csv", sep=",")
>   colnames(gen) <- paste("M",1:127,sep="")
>   #Imputation der fehlenden Werte:
>   gen0 <- gen
>   set.seed(11)
>   gen0[gen0==0] <- sample(c(1,-1), size=sum(gen0==0), replace=TRUE,
+                     prob=c(0.5,0.5))
>   gen0[gen0==-1] <- 0
```

c) Distanzbasiertes Clustern

Definition der verwendeten Distanzmaße.

```
>   library(cluster)
>   library(fpc)
>   #Hier können beliebig viele Distanzmaße hinzugefügt werden.
>   distMat <- list(
+     euc = daisy(mat92, metric="euclidean"),
+     man = daisy(mat92, metric="manhattan")
+   )
>   #Namen der verwendeten Distanzmaße werden festgelegt.
>   distNames <- c("Euclidean","Manahttan")
```

k-Means-Verfahren

k-Means-Verfahren für die gewählten Distanzmaße, für 2 bis 8 Cluster.

```
>   kmeans2_8 <- lapply(2:8, function(x){kmeans(mat92, x, nstart=1000)})
```

Agglomeratives Verfahren

```
>   s_hclust <- lapply(distMat, function(x){hclust(x, method="single")})
>   c_hclust <- lapply(distMat, function(x){hclust(x, method="complete")})
>   a_hclust <- lapply(distMat, function(x){hclust(x, method="average")})
>   s_coph <- lapply(s_hclust, function(x){cophenetic(x)})
>   c_coph <- lapply(c_hclust, function(x){cophenetic(x)})
>   a_coph <- lapply(a_hclust, function(x){cophenetic(x)})
>   cor_mat <- rbind(
+         sapply(1:2, function(i){cor(s_coph[[i]], distMat[[i]])}),
+         sapply(1:2, function(i){cor(c_coph[[i]], distMat[[i]])}),
+         sapply(1:2, function(i){cor(a_coph[[i]], distMat[[i]])})
+   )
>   colnames(cor_mat) <- distNames
>   rownames(cor_mat) <- c("single","complete","average")
```

Silhouttenplots/-werte für 2 Klassen:

```
>   average_hir.C2 <- matrix(0,3,2)
>   sil_hir.s.C2 <- lapply(1:2, function(x){
+   silhouette(cutree(s_hclust[[x]],2), distMat[[x]])})
```

```
>   average_hir.C2[1,] <- sapply(sil_hir.s.C2, function(x){mean(x[,3])})
>   sil_hir.c.C2 <- lapply(1:2, function(x){
+   silhouette(cutree(c_hclust[[x]],2), distMat[[x]])})
>   average_hir.C2[2,] <- sapply(sil_hir.c.C2, function(x){mean(x[,3])})
>   sil_hir.a.C2 <- lapply(1:2, function(x){
+   silhouette(cutree(a_hclust[[x]],2), distMat[[x]])})
>   average_hir.C2[3,] <- sapply(sil_hir.a.C2, function(x){mean(x[,3])})
```

Silhouttenplots/-werte für 3 Klassen:

```
>   average_hir.C3 <- matrix(0,3,2)
>   sil_hir.s.C3 <- lapply(1:2, function(x){
+   silhouette(cutree(s_hclust[[x]],3), distMat[[x]])})
>   average_hir.C3[1,] <- sapply(sil_hir.s.C3, function(x){mean(x[,3])})
>   sil_hir.c.C3 <- lapply(1:2, function(x){
+   silhouette(cutree(c_hclust[[x]],3), distMat[[x]])})
>   average_hir.C3[2,] <- sapply(sil_hir.c.C3, function(x){mean(x[,3])})
>   sil_hir.a.C3 <- lapply(1:2, function(x){
+   silhouette(cutree(a_hclust[[x]],3), distMat[[x]])})
>   average_hir.C3[3,] <- sapply(sil_hir.a.C3, function(x){mean(x[,3])})
```

Silhouttenplots/-werte für 4 Klassen:

```
>   average_hir.C4 <- matrix(0,3,2)
>   sil_hir.s.C4 <- lapply(1:2, function(x){
+   silhouette(cutree(s_hclust[[x]],4), distMat[[x]])})
>   average_hir.C4[1,] <- sapply(sil_hir.s.C4, function(x){mean(x[,3])})
>   sil_hir.c.C4 <- lapply(1:2, function(x){
+   silhouette(cutree(c_hclust[[x]],4), distMat[[x]])})
>   average_hir.C4[2,] <- sapply(sil_hir.c.C4, function(x){mean(x[,3])})
>   sil_hir.a.C4 <- lapply(1:2, function(x){
+   silhouette(cutree(a_hclust[[x]],4), distMat[[x]])})
>   average_hir.C4[3,] <- sapply(sil_hir.a.C4, function(x){mean(x[,3])})
```

Divisives Verfahren

```
> mat92_diana <- lapply(1:2, function(x){diana(distMat[[x]])})
> sil_diana <- lapply(2:5, function(x){
+ silhouette(cutree(mat92_diana[[1]],x), distMat[[1]])})
> average_diana <- sapply(sil_diana, function(x){mean(x[,3])})
```

d) Modellbasiertes Clustern

Modellbasiertes Clusterverfahren mittels der R-Funktion `Mclust`.

```
>   library(mclust)
>   Mcluster1 <- Mclust(data=mat92, G=1:8)
>   summary(Mcluster1)
>   #BIC-Plot:
>   plot(Mcluster1, what="BIC")
>   #Klassifikations-Plot: (Mclust)
>   plot(Mcluster1, what="classification")
>   #Klassifikations-Plot: (pairs)
>   pairs(mat92, col=Mcluster1$classification)
>   #Unsicherheits-Plot
>   plot(u,type="b", ylab="Unsicherheit", xlab="Beobachtung")
>   abline(h=0.05, lty=2)
```

e) Ermittlung der signifikanten Genmarker

Signifikante Genmarker: k-Means-Verfahren

Ermittlung der p-Werte der χ^2-Tests für die Clusterlösung des k-Means-Verfahrens.

```
>   anteile_euc_kmeans <-
+   cbind(
+   Gruppe1=apply(gen0[clust_euc_kmeans==1,],2,mean),
+   Gruppe2=apply(gen0[clust_euc_kmeans==2,],2,mean)
+   )
>   n1 <- sum(clust_euc_kmeans==1)
>   n2 <- sum(clust_euc_kmeans==2)
>   num_suc1 <- apply(gen0[clust_euc_kmeans==1,],2,sum)
>   num_suc2 <- apply(gen0[clust_euc_kmeans==2,],2,sum)
>   p0.05 <- sapply(1:127, function(x){
+   prop.test(matrix(c(num_suc1[x], num_suc2[x],
+                      n1-num_suc1[x], n2-num_suc2[x]),2,2),
+             correct=F, conf.level = 0.95)$p.value
+   }
+   )
>   p0.1 <- sapply(1:127, function(x){
+   prop.test(matrix(c(num_suc1[x], num_suc2[x],
```

```
+                     n1-num_suc1[x], n2-num_suc2[x]),2,2),
+            correct=F, conf.level = 0.90)$p.value
+   }
+   )
```

Adjustierung der p-Werte (k-Means-Verfahren):

```
>   padj0.05 <- p.adjust(p0.05, "BH")
>   padj0.1 <- p.adjust(p0.1, "BH")
```

Barplot der signifikanten Anteilsunterschiede (k-Means-Verfahren):

```
>   barpoint <- rep("grey",127)
>   barpoint[padj0.1<0.1] <- "orange"
>   barpoint[padj0.05<0.05] <- "red"
>   barplot(anteile_euc_kmeans[,1]-anteile_euc_kmeans[,2], ylim=c(-0.3,0.3),
+   col=barpoint, ylab="Anteilsdifferenz zwischen den Clustern",
+   xlab="Genmarker")
>   legend("topright",legend=c("nichtsignifikant"),
+          col="grey", cex=0.7, seg.len=1, pch=15)
```

Signifikante Genmarker: `Mclust`

Ermittlung der p-Werte der χ^2-Tests für die Clusterlösung von `Mclust`.

```
>   clust_mclust <- Mcluster1$classification
>   anteile_clust_mclust <-
+   cbind(
+   Gruppe1=apply(gen0[clust_mclust==1,],2,mean),
+   Gruppe2=apply(gen0[clust_mclust==2,],2,mean)
+   )
>   n1 <- sum(clust_mclust==1)
>   n2 <- sum(clust_mclust==2)
>   num_suc1 <- apply(gen0[clust_mclust==1,],2,sum)
>   num_suc2 <- apply(gen0[clust_mclust==2,],2,sum)
>   p0.05 <- sapply(1:127, function(x){
+   prop.test(matrix(c(num_suc1[x], num_suc2[x],
+                     n1-num_suc1[x], n2-num_suc2[x]),2,2),
+            correct=F, conf.level = 0.95)$p.value
+   }
+   )
```

```
>   p0.1 <- sapply(1:127, function(x){
+   prop.test(matrix(c(num_suc1[x], num_suc2[x],
+                      n1-num_suc1[x], n2-num_suc2[x]),2,2),
+             correct=F, conf.level = 0.90)$p.value
+   }
+   )
```

Adjustierung der p-Werte (`Mclust`):

```
>   padj0.05 <- p.adjust(p0.05, "BH")
>   padj0.1 <- p.adjust(p0.1, "BH")
```

Barplot der signifikanten Anteilsunterschiede (`Mclust`):

```
>   barpoint <- rep("grey",127)
>   barpoint[padj0.1<0.1] <- "orange"
>   barpoint[padj0.05<0.05] <- "red"
>   barplot(anteile_clust_mclust[,1]-anteile_clust_mclust[,2], ylim=c(-0.3,0.3),
+   col=barpoint, ylab="Anteilsdifferenz zwischen den Clustern",
+   xlab="Genmarker")
>   legend("topleft",legend=c("nichtsignifikant",
+                             "signifikant zum Niveau alpha = 0.1",
+                             "signifikant zum Niveau alpha = 0.05"),
+          col=c("grey","orange","red"), cex=0.7, seg.len=1, pch=15)
```

f) Multivariate LASSO-Regression

```
> library("glmnet")
> #Modell:
> X <- as.matrix(gen0)
> y <- as.matrix(mat92)
> mod2 <- glmnet(x=X, y=y, family=c("mgaussian"), alpha = 1, nlambda = 200)
>
> #bestes Lambda durch Kreuzvalidierung ermitteln:
> #cvfit2 = cv.glmnet(x=X, y=y, family=c("mgaussian"))
> #coef(cvfit2, s = "lambda.min")
```

g) Tabelle der signifikanten Genmarker

```
> signi_lasso_multi_9 <- coef(mod2, s=0.8041)[[1]]@i[-1] #9
> signi_lasso_multi_22 <- coef(mod2, s=0.5816)[[1]]@i[-1] #22
```

```
> signi_mclust_0.05 <- c(11,14,48,63,71,101,102,116)
> signi_mclust_0.1 <- c(8,9,11,12,13,14,47,48,52,53,61,62,63,
+                       71,72,86,87,101,102,103,104,116)
> signi <- matrix("",nrow=127,ncol=4)
> rownames(signi) <- 1:127
> signi[signi_mclust_0.1,1] <- "x"
> signi[signi_mclust_0.05,2] <- "x"
> signi[signi_lasso_multi_22,3] <- "x"
> signi[signi_lasso_multi_9,4] <- "x"
> signi_table <- subset(signi, signi[,1]!="" | signi[,2]!=""
+                       | signi[,3]!="" | signi[,4]!="")
> colnames(signi_table) <- c("alpha=0.1","alpha=0.05",
+                            "lambda=0.5816","lambda=0.8041")
```