Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering
IDP Institut für Datenanalyse
und Prozessdesign

Modul C: Data Mining mit Schwerpunkt auf Clustering und Klassifikation

Martin Frey

2. Juni 2020

Vorbemerkungen

Dieses Skript wurde als Ergänzung zum CAS Modul Clustering und Klassifikation konzipiert. Es enthält neben der Theorie direkt auch Anwendungen in der Statistiksoftware R. Das Skript ist noch in Arbeit. Der erste Teil zum Unüberwachten Lernen ist mehr oder weniger komplett. Der Teil zum Überwachten Lernen ist hingegen noch nicht vollständig.

Hinweise betreffend Fehlern oder Unklarheiten sowie Verbesserungsvorschläge werden gerne entgegengenommen.

Grosse Teile des Skripts wurden aus den Vorlesungsunterlagen von Prof. A. Ruckstuhl und von Prof. B. Sick zusammengestellt.

Inhaltsverzeichnis

1 Einführung

1.1 Einordung der behandelten Themen

Clustering und Klassifikationsverfahren werden oft im Zusammenhang mit *Data Mining* diskutiert. Von Data Mining wird meist gesprochen, wenn man es mit riesigen und komplexen Datenmengen - auch *Big Data* genannt - zu tun hat, die sehr vielschichtig sein können und nicht immer in Form von aufbereiteten Datentabellen vorliegen müssen. Insbesondere kann es vorkommen, dass an einer Beobachtungseinheit (z.B. ein Kunde, ein Patient, eine Firma, eine Sprachaufnahme oder eine Gesteinsprobe) zig-tausend Merkmale erhoben werden, so dass die Anzahl der Merkmale die Anzahl der Beobachtungseinheiten übersteigen kann. Charakteristisch für Data Mining Anwendungen ist ausserdem, dass die Daten nicht zu einem bestimmten Zweck - z.B. zur Überprüfung einer bestimmten Hypothese - erhoben wurden. Man zeichnet z.B. das Telefonie-Nutzungsverhalten zur Rechnungsstellung auf, und zieht aber dann Rückschlüsse über die Dauer der Vertragsbindung.

Clustering und Klassifikation werden aber nicht nur im Zusammenhang von Data Mining eingesetzt, sondern sind auch Teil der klassischen Statistik und können auch auf etliche der Datensätze, die Sie in den vorangegangenen Modulen analysiert haben, angewandt werden. An dieser Stelle soll erinnert werden, dass sich die angewandte Statistik grob in vier Gebiete einteilen lässt. Sie lauten wie folgt:

- **Versuchsplanung:** Hier stellt man sich die Frage, welche Daten überhaupt, in welcher Anzahl und auf welche Art und Weise erhoben werden sollen, damit die Frage von Interesse beantwortet werden kann.
- **Deskriptive Statistik:** Sind Daten vorhanden, so bilden diese zuerst in der Regel einen unübersichtlichen, *ungeordneten Haufen*. Mit den Methoden der deskriptiven Statistik werden die Daten zusammengefasst und visualisiert.
- **Schliessende Statistik:** In der schliessenden Statistik geht es darum, quantitative Folgerungen aus den Daten zu ziehen. Man versucht signifikante Unterschiede zwischen zwei verschiedenen Behandlungen zu finden und untersucht, welche Zusammenhänge in statistisch gesicherter Art und Weise bestehen.
- **Explorative Statistik / Data Mining** Hierbei handelt es sich um eine Art Suche *nach der Nadel im Heuhaufen*. Man sucht nach unbekannten und unerwarteten Strukturen in den Daten, nach neuen Hypothesen und Dingen, die man *noch nicht auf diese Art gesehen hat*.

  > Data are not taken for museum purposes; they are taken as a basis for doing something. If nothing is to be done with data, then there is no use in collecting any. The ultimate purpose of taking data is to provide a basis for action or a recommendation for action.
  > (W. Edwards Deming, 1942)

1.2 Spezielle Herausforderungen des Data Minings

Riesige Datenmengen können heute dank der Informatik in Unternehmen, in Forschungsprojekten, in Verwaltungen oder im Internet zusammengetragen werden. Um diese Daten für Entscheidungen und Dienstleistungen nutzbar zu machen, müssen sie jedoch in geeigneter Weise aufbereitet und interpretiert werden. Ein wichtiger Schritt darin ist es, Regeln, Muster oder Auffälligkeiten in den Datenbeständen aufzuspüren. So sollen Änderungen im Verhalten von Kunden oder Kundengruppen, oder Veränderungen im Finanzanlagemarkt, möglichst früh erkannt und in der Folge Geschäfts- bzw. Anlagestrategien darauf ausgerichtet werden. Es kann aber auch abweichendes Verhalten einzelner Personen erkannt werden, um z.B. Missbräuche (Versicherungsbetrug, Kreditkartenbetrug, ...) festzustellen.

1.3 Unüberwachte Verfahren zur Strukturentdeckung

Unüberwachtes Lernen (englisch unsupervised learning) zeichnet sich durch das Nichtvorhandensein einer Zielvariablen aus. In diesem Gebiet geht es darum, ohne im Voraus bekannte Zielvariable verborgene oder unbekannte Strukturen in den Daten aufzudecken und/oder zu visualisieren sowie Objekte mit ähnlichen Eigenschaften in Gruppen zusammenzuführen.

Zum Beispiel hat man für jeden Kunden verschiedenste Merkmale gesammelt (Alter, Nationalität, Rechnungsbeträge, gekaufte Produkte, Bankauskünfte, Anrufe bei der Hotline, ...) und will nun herausfinden, ob es innerhalb der Kundschaft Gruppierungen gibt. Hätten wir für jeden Kunden nur 2 Merkmale gesammelt (z.B. Alter und höchster Rechnungsbetrag), so könnten wir die Daten in einer zwei-dimensionalen Darstellung visualisieren (z.B. ein Streudiagramm mit höchstem Rechnungsbetrag gegen Alter) und leicht optisch untersuchen, ob es interessante Strukturen gibt (z.B. könnte der höchste Rechnungsbetrag tendenziell mit dem Alter steigen).

Eine geeignete Visualisierung der Daten ist praktisch immer ein guter Einstieg in die Datenanalyse. Im Fall von hoch-dimensionalen Daten, bei denen es weit mehr als 2 Merkmale pro Untersuchungseinheit gibt, stellt eine Visualisierung der Daten jedoch eine Herausforderung dar. Es gibt verschiedene Ansätze Zusammenhänge oder Strukturen in hoch-dimensionalen Daten sichtbar zu machen. Wir werden in diesem Kurs folgende unüberwachte Verfahren behandeln:

- **Hauptkomponentenanalyse und multi-dimensional Scaling:** Die Hauptidee dieser Verfahren besteht darin, einen niedrig-dimensionalen Unterraum bzw. eine Hyper-Ebene zu identifizieren, in welchem die hoch-dimensionalen Daten gut angenähert werden können. Dabei werden aus den vielen ursprünglichen Variablen ein kleiner Satz von neuen Variablen – auch Komponenten genannt – konstruiert, durch die die Daten schon möglichst gut beschrieben werden können. Die ersten 2 oder 3 dieser Komponenten spannen dann einen 2- oder 3-dimensonalen Raum auf, in dem es dann wieder möglich ist, die Daten zu visualisieren.
- **Clusteranalyse:** Eine weitere Möglichkeit Strukturen in hoch-dimensionalen Daten sichtbar zu machen, bietet die Clusteranalyse. Die grundlegende Idee bei den verschiedenen Clusterana-

lyse Verfahren ist es, im hochdimensionalen Merkmalsraum Regionen zu finden, welche eine Anhäufung zusammenhängender Datenpunkte enthalten. Die Beobachtungen (z.B. Kunden mit allen ihren Merkmalen) in dieser Region bilden dann einen sogenannten Cluster oder eine Gruppe. Dabei versucht man diese Cluster bzw. deren Regionen so zu definieren, dass die Daten innerhalb eines Clusters möglichst homogen sind und sich Beobachtungen aus verschiedenen Gruppen stärker unterscheiden, als Beobachtungen, die zum gleichen Cluster gehören.

1.4 Überwachte Verfahren zur Strukturentdeckung

Wenn die Beobachtungen schon mit einem Gruppen- oder Klassenlabel vorliegen, so können überwachte Lernverfahren (englisch supervised learning) eingesetzt werden, um ein Regelwerk zu entwickeln, mit dem es möglich ist die Beobachtungen aufgrund ihrer Merkmale der korrekten Klasse zu zuordnen. Mit dem Regelwerk können dabei auch neue Beobachtungen klassifiziert werden. Solche Verfahren werden oft als Klassifikationsmethoden bezeichnet.

Anhand der gefunden Klassifikationsregeln, die sehr komplex sein können, lässt sich bei den meisten Klassifikationsverfahren auch ableiten, welche Merkmale besonders wichtig sind, um Klassenzugehörigkeit korrekt vorherzusagen. Darauf basierend wird auch eine Variablenselektion vorgenommen oder eine Charakterisierung der einzelnen Klassen anhand dieser wichtigen Variablen vorgenommen.

Für die Klassifikation gibt es nicht nur ein Verfahren. Je nach Fragestellung und Daten funktioniert die eine oder andere Methode besser. Wir wollen im Rahmen dieses Kurses einige typische, oft verwendete Konzepte kennenlernen. Daneben ist es eine wichtige Aufgabe für ein spezifisches Problem den “besten” Ansatz zu evaluieren. Wir werden dazu die wichtigsten Konzepte zur Modellevaluation besprechen.

Geplant ist es, dass wir in diesem Kurs folgende überwachte Verfahren behandeln:

- **Nächste-Nachbarn-Klassifizierer**
- **Lineare Diskriminanzanalyse**
- **Support Vector Machine**
- **Klassifikationsbäume**, welche mit Ensemble-Verfahren zu **Random Forest** erweitert werden können.

Lernziele des Moduls

Sie können . . .

- hochdimensionale Daten mit geeignten Methoden visualisieren.
- gängige Methoden zur Strukturentdeckung in Daten anwenden.
- einem Objekt mit einer Auswahl von Klassifikationsverfahren Daten-gestützt seine Klassenzugehörigkeit ermitteln.
- die Performance eines Klassifikationsverfahrens bei einem gegebenen Datensatz ermitteln.

Teil I

Unüberwachtes Lernen

2 Hauptkomponentenanalyse (PCA)

2.1 Einführung

Die Hauptkomponentenanalyse (englisch: principal component analysis, PCA) ist ein von Pearson (1901) und Hotelling (1933) entwickeltes Verfahren der multivariaten Statistik das zur **Dimensionsreduktion**, **Datenkompression**, **Merkmalsextraktion** und **Ausreisserdetektion** verwendet wird. Im Folgenden wird zuerst veranschaulicht, was unter einer Dimensionsreduktion zu verstehen ist. Anschliessend wird dann beschrieben, wie diese Dimensionsreduktion technisch bewerkstelligt wird.

Ausgangspunkt ist ein hochdimensionaler Datensatz, bei dem jede Beobachtung durch eine grosse Anzahl von Merkmalen $X_1, X_2, \ldots, X_p$ beschrieben wird. D.h. die Beobachtungen liegen in einem p-dimensionalen Merkmalsraum. Da grafische Darstellungen nur im 2- oder maximal 3-dimensionalen Raum gut möglich sind, ist es nicht ohne weiteres möglich, sich einen visuellen Überblick über die Daten zu verschaffen.

In vielen Anwendungen ist es wünschenswert, den hochdimensionalen Datensatz so zu reduzieren, dass man mit nur wenigen Merkmalen auskommt. Zum einen ist es dann besser möglich, die Daten zu visualisieren oder auch durch Kennzahlen zu präsentieren und zum anderen können auch mögliche Zusammenhänge zwischen wenigen Variablen besser erfasst und interpretiert werden. **Das Ziel ist es also, aus den vielen Merkmalen 2 oder 3 neue Merkmale zu bilden, die das Wesentliche, was in den Daten steckt, enthalten**. Im Idealfall kann dies aufgrund des Fachwissens geschehen. Es gibt aber auch rein Datengetriebene Ansätze und einer der bekanntesten Verfahren ist die hier diskutierte **PCA**.

Durch die PCA soll eine grosse Anzahl von Merkmalen durch eine kleinere Anzahl "dahinter liegender" Variablen ersetzt werden, welche bei der PCA durch die ersten paar Hauptkomponenten (englisch: principal components, PC) gegeben sind. Insbesondere wenn die ursprünglichen Merkmale untereinander stark korreliert sind, sollte es möglich sein, die Dimension des Merkmalsraums ohne grossen Informationsverlust stark zu reduzieren, indem unkorrelierte Hauptkomponenten konstruiert werden.

Betrachten wir als einfaches Beispiel den Fall in der Abbildung 2.1, in welcher Beobachtungen in einem 3-dimensionalen Raum dargestellt sind, welcher durch die Merkmale „feature 1", „feature 2" und „feature 3" aufgespannt wird. In der 3-dimensionalen Darstellung (linke Grafik) ist sichtbar, dass alle Beobachtungen in erster Näherung in einer Ebene liegen. Werden die Beobachtungen senkrecht auf diese Ebene projiziert, so können sie auch in diesem 2-dimensionalen Raum dargestellt

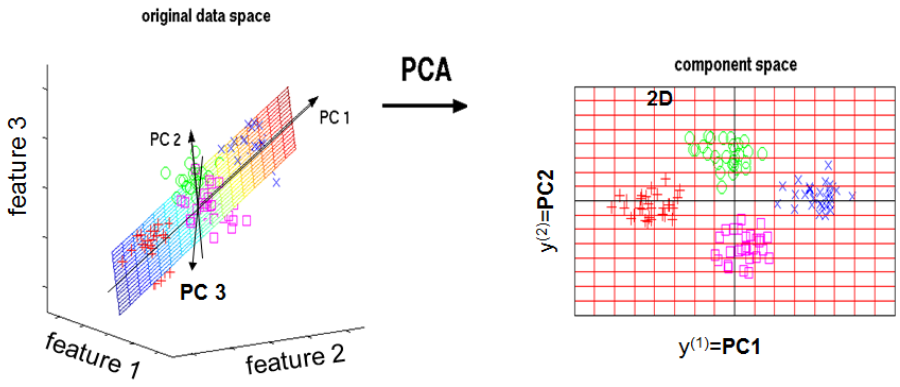

Abbildung 2.1: Grundidee der PCA: Verschiebe das Koordinatensystem in den Datenschwerpunkt und Drehung, so dass die erste Hauptkomponente (PC1) in die Richtung maximaler Varianz zeigt und die 2. Hauptkomponente (PC2) unkorreliert zur ersten Hauptkomponente ist und in die Richtung mit nächstgrössten Varianz zeigt.

werden (rechte Grafik), wobei nur wenig Information verloren geht – umso weniger Information geht verloren, je genauer die Beobachtungen im ursprünglichen Koordinatensystem schon in dieser (Hyper-) Ebene liegen. Die Farbe der Datenpunkte ist willkürlich und wird hier nur benutzt, um Gruppierungen sichtbar zu machen und zu demonstrieren, dass diese Gruppierungen auch in einer 2-dimensionalen Darstellung noch sichtbar sind (die Gruppeninformation wird bei diesem Verfahren in keiner Weise benutzt). Dies unterstreicht die Hoffnung, dass auch nach einer Dimensionsreduktion wichtige Strukturen in den Daten immer noch sichtbar sind.

Das Streudiagramm in der rechten Grafik von Abbildung 2.1 wird durch die ersten beiden Hauptkomponenten PC1 und PC2 aufgespannt. **Dabei ist die erste Hauptkomponente (PC1) parallel zu der Richtung, entlang der die projizierten Daten die grösste Streuung aufweisen und PC2 ist senkrecht zu PC1 und entlang der Richtung mit der zweitgrössten Streuung der Daten.** Als Mass für die Streuung wird die Varianz verwendet. Senkrecht (bzw. orthogonal) zu einander stehen die Achsen, damit sie unkorreliert sind.

In der linken Grafik von Abbildung 2.1 ist auch die Richtung der 3. Hauptkomponente (PC3) angedeutet, die wiederum senkrecht auf der Ebene steht, die von PC1 und PC2 aufgespannt wird. Dies wäre auch im Fall von mehr als 3 Merkmalen der Fall und es kann gezeigt werden, dass PC3 ausserdem in die Richtung zeigt, in der die Daten ausserhalb der PC1-PC2-Ebene die nächstgrösste Varianz haben. Im 3-dimensionalen Fall ist es allerdings auch gleichzeitig die Richtung, für die die projizierten Punkte die kleinste Varianz haben.

2.2 Die Hauptachsentransformation

Der Übergang von den ursprünglichen Variablen $X_1, X_2, \ldots, X_p$ zu den Hauptkomponenten $PC_1, PC_2, \ldots PC_n$ kann als Wechsel in ein neues Koordinatensystem aufgefasst werden. **Der Ursprung des Hauptachsen-Koordinatensystems liegt im Mittelpunkt bzw. Schwerpunkt**

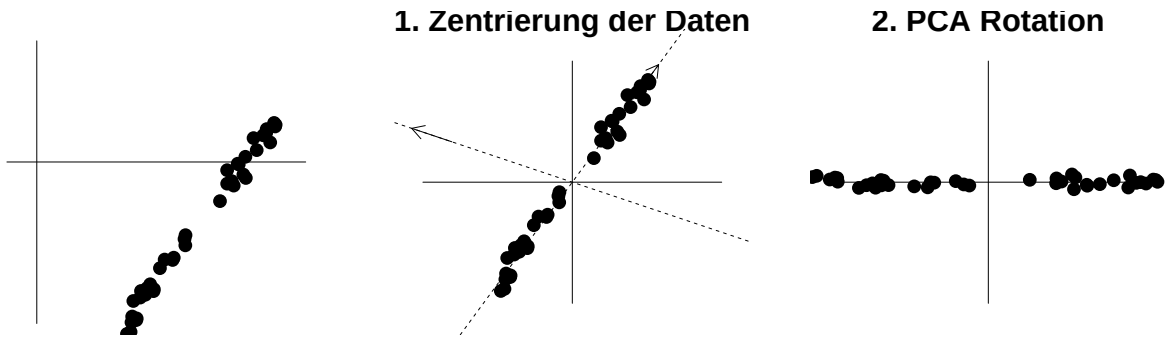

Abbildung 2.2: PCA in zwei Schritten: 1) zentrieren der Daten, 2) Rotation in das Hauptachsensystem.

der Punktwolke. Dann werden die Achsen so gedreht, dass die **erste Hauptkomponente** PC_1 **in die Richtung der grössten Varianz zeigt.** Die zweite Hauptkomponente steht senkrecht zur ersten Hauptkomponente und es wird unter allen orthogonalen Richtungen diejenige gewählt, entlang der die projizierten Punkte die nächstgrössten Varianz haben. Für die weiteren Achsen gilt analoges.

Durch einen Wechsel des Koordinatensystems ist per se noch keine Dimensionsreduktion erreicht. Da aber die Hauptkomponenten so konstruiert werden, dass sie unkorreliert voneinander sind und in absteigender Ordnung immer einen möglichst grossen Teil der verbleibenden Varianz abdecken, sollten die ersten Hauptachsen einen Grossteil der Varianz der Daten abdecken. Wenn man voraussetzt, dass Variablen, in denen es zwischen den Beobachtungen grosse Unterschiede in ihren Werten gibt (und die deshalb eine grosse Varianz haben), besonders viel Information beinhalten, so steckt in der (Hyper-) Ebene, die durch die 2 ersten Hauptkomponenten aufgespannt wird, mehr Information als in jeder anderen (Hyper-) Ebene im Merkmalsraum. Damit ist ein Streudiagramm mit den ersten beiden Hauptkomponenten als Achsen optimal, wenn es darum geht, die Beobachtungen in einem 2-dimensionalen Streudiagramm darzustellen.

Schritte der Hauptachsentransformation (siehe Abbildung 2.2):

1. Zentrieren der einzelnen Variablen → zentrierte Variablen $X_1, X_2, \ldots, X_p$ mit $mean(X_1) = mean(X_2) = \ldots = mean(X_p) = 0$. Gegebenenfalls standardisieren (siehe Kapitel 2.5) → sd(X1) = sd(X2) =. . . = sd(Xp)=1

2. Rotation des Koordinatensystems ins Hauptachsensystem $Y_1, Y_2, \ldots, Y_p$, so dass

- die PCs unkorreliert bzw. orthogonal sind
- $Var(Y_1) \geq Var(Y_2) \geq \ldots \geq Var(Y_p)$

Bei der PCA werden neue Variablen $Y_1, Y_2, \ldots, Y_p$ – die sogenannten Hauptkomponenten – konstruiert, indem Linearkombinationen der ursprünglichen, zentrierten (d.h. die ursprünglichen Daten

haben schon den Mittelwert Null) Variablen $X_1, X_2, \ldots, X_p$ gebildet werden:

$$Y_k = a_{1k}X_1 + a_{2k}X_2 + ... + a_{pk}X_p$$

Die Koeffizienten a_{ij} werden auch als „loadings“ bezeichnet, da sie die Beiträge der einzelnen ursprünglichen Merkmale $X_1, \ldots, X_p$ zu der jeweiligen Hauptkomponente angeben.

Bei einer linearen Transformation transformiert sich der Mittewert mit. Daraus ergibt sich, dass die Hauptkomponenten wie die ursprünglichen (zentrierten) Variablen den Mittelwert Null haben, also zentriert sind. Wir beschränken uns auf sogenannte Standard-Linearkombinationen, bei denen sich die quadrierten Koeffizienten zu 1 aufaddieren (d.h. der Vektor der Koeffizienten hat den Betrag 1):

$$a_{1k}^2 + a_{2k}^2 \ldots a_{pk}^2 = 1$$

Durch die Normierung wird gesichert, dass die Transformation orthogonal ist, d.h. dass die Abstände zwischen den Datenpunkten im p-dimensionalen Raum erhalten bleiben, wie es bei einer Rotation der Fall sein sollte.

Für eine konkrete Beobachtung, welche bezüglich der p ursprünglichen (zentrierten!) Merkmale den Merkmalsvektor x mit den Werten $x_1, x_2, \ldots, x_p$ hat, können die Einträge des neuen Merkmalsvektors $y = (y_1, \ldots, y_p)$ bezüglich der p neuen Merkmale (PCs) dementsprechend wie folgt berechnet werden:

$$\begin{aligned}
y_1 &= a_{11}x_1 + a_{21}x_2 + \ldots + a_{p1}x_p \\
y_2 &= a_{12}x_1 + a_{22}x_2 + \ldots + a_{p2}x_p \\
&\vdots \\
y_p &= a_{1p}x_1 + a_{2p}x_2 + \ldots + a_{pp}x_p
\end{aligned}$$

Die Matrix mit den y-Werten wird auch als **Score-Matrix** bezeichnet.

Die p Hauptkomponenten werden so konstruiert, dass sie als Linearkombination der ursprünglich beobachteten (zentrierten) Merkmalen $X = (X_1, \ldots, X_p)$ beschrieben werden können, wobei gilt:

- Der Ursprung des neuen Koordinatensystems liegt im Schwerpunkt der Datenwolke, d.h. alle Hauptkomponenten haben den Mittelwert Null.
- Die erste Hauptkomponente PC1 zeigt in die Richtung der grössten Varianz.
- Die zweite Hauptkomponente PC2 ist senkrecht zu PC1 und zeigt in die Richtung der nächstgrössten Varianz ... usw.
- Alle Hauptkomponenten sind unkorreliert.
- Alle Hauptkomponenten sind zentriert.

Diese Hauptkomponenten lassen sich immer konstruieren. Die Konstruktion erfordert etwas lineare Algebra, ist im Prinzip aber nicht sehr kompliziert. Mathematischen Details und eine Skizze der Herleitung ist im Anhang zu finden.

2.3 Dimensionsreduktion

Nachdem aus den p ursprünglichen (zentrierten) Variablen die p (zentrierten) Hauptkomponenten konstruiert wurden, wollen wir diese zur Dimensionsreduktion nutzen, indem wir nicht alle p, sondern nur die ersten paar Hauptkomponenten verwenden.

Damit stellt sich die Frage, wie viele Hauptkomponenten ausreichen und was "ausreichen" eigentlich heissen soll. D.h., wir brauchen ein Kriterium, um zu beurteilen, wie "gut" eine vorgegebene Anzahl von Hauptkomponenten die "wahre" p-dimensionale Darstellung der Daten approximiert.

Als Gütekriterium wird üblicherweise angeschaut, welcher Anteil der Gesamtvarianz der Beobachtungen, bereits im reduzierten Raum der ersten k Hauptkomponenten enthalten ist.

Die Varianz ergibt sich aus der Summe der Varianzen der einzelnen Variablen und ändert sich durch eine Drehung des Koordinatensystems nicht. D.h. die totale Varianz V_{total} lässt sich sowohl mit den ursprünglichen Variablen als auch mit Hauptkomponenten wie folgt berechnen:

totale Varianz: $V_{total} = \sum_{j=1}^{p} var(X_j) = \sum_{j=1}^{p} var(Y_j)$

Als Faustregel gilt, dass durch die ersten k Hauptkomponenten mindestens 75% bis 80% der totalen Varianz der Daten erfasst sein sollte, damit von einer angemessenen Repräsentation der Daten im k-dimensionalen Raum gesprochen werden kann.

Als Parameter für die Güte wird der Anteil P (P für "proportion") der erfassten Varianz der ersten k Hauptkomponenten an der totalen Varianz verwendet:

$P_k = \frac{\sum_{j=1}^{k} var(Y_j)}{V_{total}} \in [0, 1]$

Um zu entscheiden, ob zwei Hauptkomponenten genügen, um die wesentlichen Eigenheiten der Daten zu erfassen, können wir diese Faustregel wie folgt anwenden: Wir addieren die Varianzen der ersten beiden PCs und teilen die Summe durch die totale Varianz – falls der so errechnete Anteil erfasster Varianz den Wert 0.75 überschreitet, wäre eine 2-dimensionale Darstellung im Streudiagramm der ersten beiden PCs angemessen (siehe auch Abbildung 2.3).

Die Forderung nach einer Abdeckung von mindestens ca. 80% der totalen Varianz ist grundsätzlich willkürlich, lässt sich aber durch die weit verbreitete 80/20 Regel von Pareto motivieren. In unserer Anwendung impliziert diese Regel, dass wir oft sehr viele Dimensionen sparen können, wenn wir auf die letzten 20% Varianz verzichten. Dies ist allerdings nur dann wahr, wenn die ursprünglichen Variablen relativ stark untereinander korreliert waren.

Ein weiteres, sehr beliebtes Verfahren zur Bestimmung der optimalen Anzahl der Hauptkomponenten besteht darin, die Varianzen der Hauptkomponenten $Var(Y_k)$ gegen ihre Ordnung k ($k = 1, \ldots, p$)

Abbildung 2.3: Die kumulierte Varianz wird gegen die Anzahl Hauptkomponenten abgetragen. Die Anzahl PCs, bei der die kumulierte Varianz die 80% Marke übersteigt gilt als ausreichend um die Daten zu beschreiben.

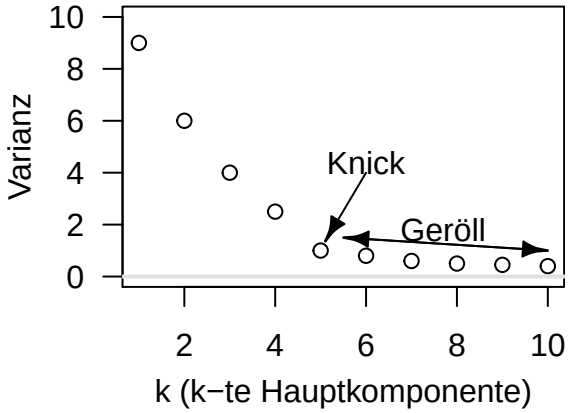

Abbildung 2.4: Im Scree-Diagramm wird die Varianz der einzelnen Hauptkomponenten gegen die Ordnungsnummer der jeweiligen Hauptkomponente abgetragen. Dort wo ein Knick zu sehen ist, ist eine ausreichende Zahl von PCs erreicht, da die nachfolgenden Hauptkomponenten nur noch einen kleine Teil zur Varianzen beitragen.

im so genannten Scree-Diagramm aufzutragen (das engl. Wort “scree” bedeutet “Geröllhang” oder das “Geröll” am Fusse einer Felswand). Ein typisches Muster, das man sich erhofft zu sehen, ist in Abbildung 2.4 gezeigt. Hier gilt als Faustregel, dass die optimale Anzahl k der PCs dort gewählt werden soll, wo der Knick im Scree-Diagramm auftritt (der Knick selber gehört hier zum “Geröll”"). Es ist dabei öfters Ermessensache, wo man die Schwelle setzt - in Figure 2.4 würde man vermutlich versuchen mit 5 PCs weiterzuarbeiten,

2.4 Projektion

Zur Analyse der Ergebnisse werden die Projektionen der Daten auf die Hauptkomponenten visualisiert. In der Regel werden dazu die ersten beiden Hauptkomponenten in einem Streudiagramm gegeneinander aufgetragen (Score-Plot), um Strukturen in den Daten sichtbar zu machen. Das macht

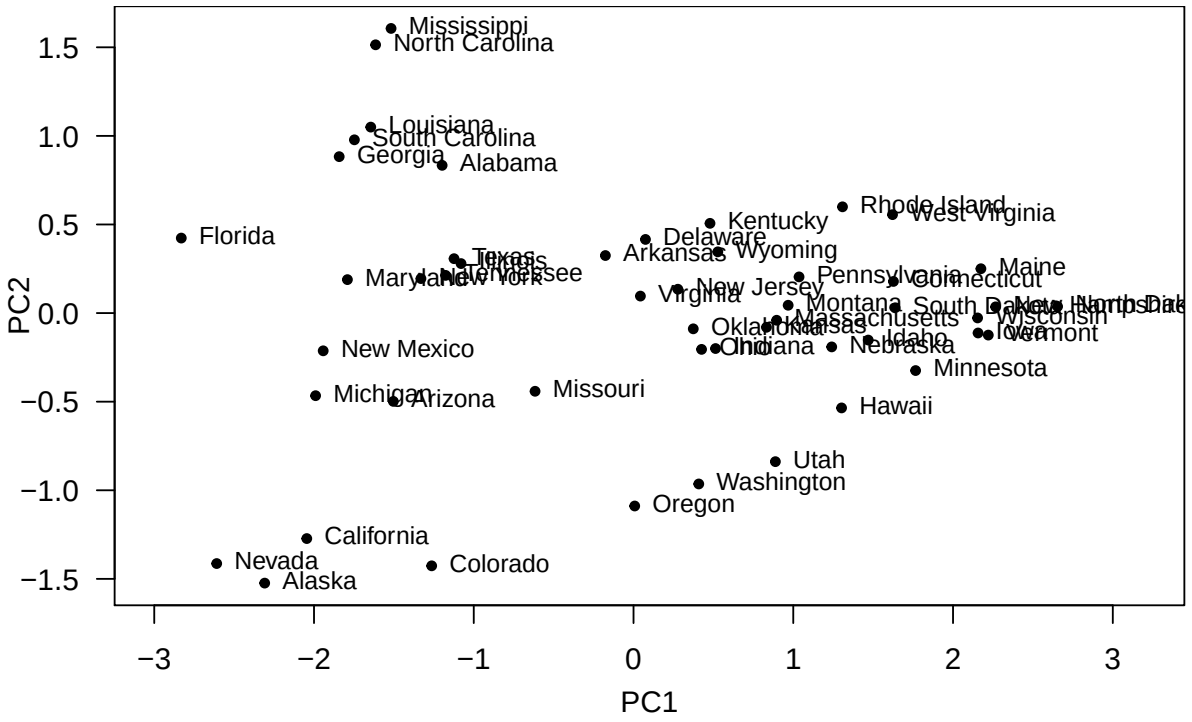

Abbildung 2.5: Verhaftete Mörder, Gewaltverbrecher und Vergewaltiger pro 100'000 Einwohner in den US-Staaten, dargestellt in den ersten beiden Hauptkomponenten (Score-Plot)

insbesondere dann Sinn, wenn durch die ersten zwei Komponenten ein grosser Teil der Varianz erklärt wird. Strukturen können ansonsten auch noch in den weiteren Hauptkomponenten versteckt sein.

Als Beispiel wurde eine Hauptkomponentenanalyse mit dem R Datensatz `USArrests` durchgeführt. Er enthält die Anzahl Gewaltverbrecher, Mörder und Vergewaltiger in den Gefängnissen in den 50 US-Staaten bezogen auf die Bevölkerung. In der Abbildung 2.5 sind die ersten beiden Hauptkomponenten aufgetragen. Anhand der Verteilung können die Staaten gemäss ihren Verbrechensraten separiert werden. Man sieht, welche Staaten ähnlich Probleme mit schweren Verbrechen haben.

2.5 Skalierung?

Wir haben diskutiert, dass die erste Hauptkomponente die Linearkombination der ursprünglichen (zentrierten) Variablen ist, welche unter allen möglichen Linearkombinationen die grösste Varianz besitzt.

Falls wir davon ausgehen können, dass alle gemessenen Variablen die gleichen Masseinheiten haben, kann es immer noch sein, dass einige Variablen viel mehr streuen als andere. Die ersten Hauptkomponenten werden dort durch die Variablen, die eine viel grössere Varianz aufweist als die anderen, dominiert. Die ersten Hauptkomponenten sind in diesem Fall hauptsächlich eine

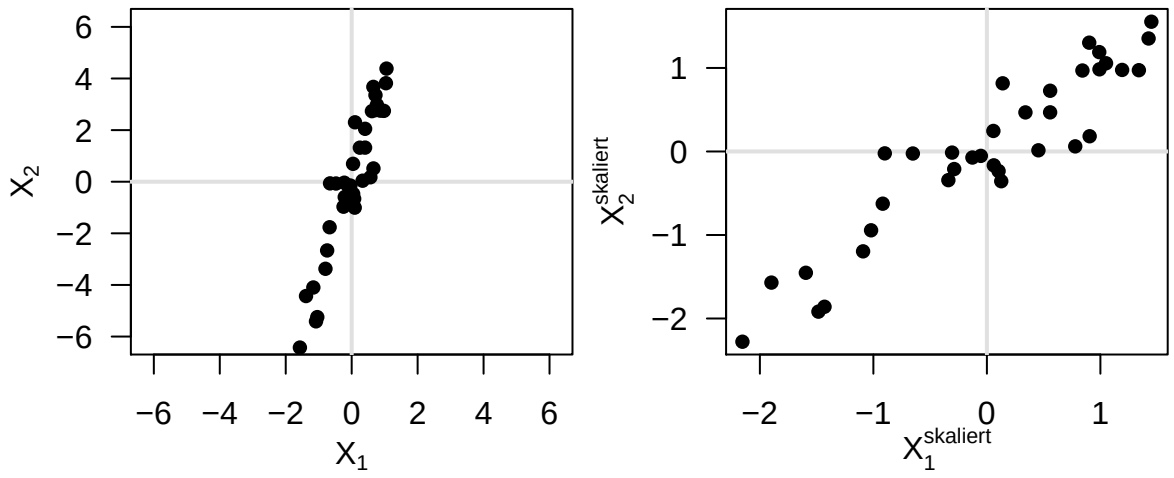

Abbildung 2.6: Skalieren der Variablen anhand eines Beispiels in 2-dimensionalen Raum – links sind die Daten vor dem Skalieren dargestellt und rechst sieht man die Daten nach dem Skalieren der Variablen.

Kombination der ursprünglichen Variablen, die stark streuen. D.h. die erste Hauptkomponente entspricht in etwa der ursprünglichen Variable, die stark streut. Will man einen solch dominierenden Effekt vermeiden, müssen alle Variablen vorgängig skaliert werden, so dass alle Variablen eine Varianz von 1 haben. So wird die erste Hauptkomponente meist nicht mehr entlang derjenigen ursprünglichen Variable liegen, die vor der Skalierung die grösste Varianz gezeigt hat.

Man könnte den Eindruck bekommen, dass es gar keine Richtung maximaler Varianz mehr gibt, nachdem alle Variablen auf die gleiche Varianz von 1 gebracht wurden. Dies ist jedoch ein Trugschluss, da es durchaus eine Richtung geben kann, entlang der die Varianz der projizierten Daten am grössten ist. Dies wird in Abbildung 2.6 demonstriert, in der die gleichen Daten vor (linker Graph) und nach (rechter Graph) der Skalierung der Variablen dargestellt sind. Vor der Skalierung ist die Varianz der Variable x_2 sehr viel grösser als die Varianz der Variable x_1; entsprechend ist die Richtung maximaler Varianz fast parallel zu x_2 und bei einer PCA würde sich als PC1 eine Linearkombination ergeben, bei der x_2 ein sehr viel grösseres Gewicht (loading) bekommt als x_1. Nachdem beide Variablen x_1 und x_2 skaliert wurden, liegt die Richtung maximaler Varianz entlang der Diagonalen und die erste Hauptkomponente würde sich als Linearkombination aus x_1 und x_2 ergeben, bei der beide Variablen mit einem ähnlichen Gewicht beitragen.

Halten wir fest: **Die Ergebnisse der PCA ändern sich i.a. wenn einzelne oder alle ursprünglichen Variablen einer Transformation unterzogen werden.** Dies gilt auch schon für lineare Transformationen. D.h. wenn ein Merkmal, welches eine Zeit beschreibt einmal in Millisekunden und einmal in Minuten angegeben wird, dann erhalten wir unterschiedliche Ergebnisse. Werden z.B. die Zeiten von verschiedenen Reaktionstests in verschiedenen Einheiten gemessen, so werden die Beträge der Millisekunden über einen grösseren Wertebereich streuen als dieselben Zeiten, wenn sie in der Einheit Minuten angegeben werden. Werden also verschiedene Reaktionstests mittels der

ersten Hauptkomponente zu einem Reaktion-Score kombiniert, dann würde der Reaktionstest, bei dem Millisekunden Werte aufgenommen werden, mit einem ungebührlich hohen Gewicht zur ersten Hauptkomponente beitragen und damit das Score stärker beeinflussen. Daher wäre es in diesem Beispiel der Reaktionstests angebracht zu skalieren.

Wann sollte man allgemein die Variablen vor der PCA standardisieren?

Als Faustregel gilt:

- Standardisieren, wenn die beobachteten Messgrössen in verschiedenen Einheiten (cm, m, kg, ...) vorliegen. Auch bei sehr grossen Unterschieden in den Varianzen der Variablen sollte man über deren Ursache nachdenken und ggf. auch dann standardisieren, wenn alle Variablen dieselbe Einheit haben.
- Nicht standardisieren, wenn die gemessenen Variablen vergleichbar sind bezüglich ihrer Einheiten und ihrer Variabilität.

Es ist in vielen Fällen also tatsächlich besser, mit standardisierten Variablen (d.h. der Mittelwert ist Null und die Standardabweichung Eins) zu arbeiten. Dabei wird jede Variable einzeln standardisiert. Es sei an dieser Stelle aber darauf hingewiesen, dass bei der PCA die ursprünglichen Variablen in jedem Fall zentriert werden, d.h. sie haben sowieso schon den Mittelwert Null, so dass es nur noch darum geht, die Varianzen anzupassen.

Bei der praktischen Durchführung der PCA in R ist es jedoch nicht nötig, die Variablen "von Hand" vor der PCA zu standardisieren, da dies über Argumente der verwendeten PCA-Funktion einfach gesteuert werden kann (siehe Kapitel PCA in R 2.8).

2.6 Interpretation der Hauptkomponenten

Die hergeleiteten Hauptkomponenten sind Linearkombinationen der ursprünglichen Merkmale. Die Hauptkomponenten sind also Funktionen der ursprünglichen Merkmale und könnten auch eine sinnvolle Interpretation haben. Das kann zu einem tieferen Verständnis der Resultate der Analyse führen und einen wertvollen Einblick in den Mechanismus oder in die Struktur der zu studierenden Situation geben.

Bei der Interpretation werden die Ladungen (Koeffizienten in der Linearkombination) betrachtet. Zuerst wird darauf geachtet, welche Variablen eine positive und welche eine negative Ladung besitzen. In zweiter Linie wird auf den Betrag der Ladung geachtet. Diejenigen Variablen, die einen sehr kleinen Betrag haben, werden zur Interpretation weggelassen.

In vielen Fällen ist es jedoch heikel, die einzelnen PCs zu interpretieren. Zwar stimmt es häufig, dass in den ersten wenigen Komponenten die wesentlichen Strukturen eingefangen werden, aber eben nur in der Kombination dieser ersten paar Hauptkomponenten.

Falls es Variablenblöcke gibt, die innerhalb eines Blocks eigentlich dasselbe messen sollen – z.B. Frageblöcke zur mathematischen Intelligenz oder sprachlichen Intelligenz,– so ergibt sich bei der

PCA häufig, dass die Fragen eines Blocks in einer PC kombiniert sind. Dies wird auch häufig als Qualitätsmerkmal für die Frageblöcke verwendet.

Auch im Fall, dass es verschiedene Scores gibt, welche z.B. die Kaufkraft eines Haushalts messen sollen, bietet die erste Hauptkomponente oft eine bessere Möglichkeit der Kombination aus all den Einzelscores, als wenn ein einfacher Mittelwert gebildet wird. D.h., dass im Fall von positiv korrelierten Daten die erste Hauptkomponente oft eine Art Mittelwertmessung ist, welche geeignet ist, um die gröbsten Unterschiede zwischen den Beobachtungen wiederzugeben. Werden beispielsweise verschiedene Kennzahlen einer Firma gesammelt (Umsatz, Anzahl Kunden, ...), so wird die PC1 vermutlich schlicht die Grösse der Firmen wiederspiegeln. Im Fall eines Patienten der Psychiatrie (Variablen: Konzentrationsstörungen, Arbeitsausfälle, depressive Phasen ...) könnte die erste Hauptkomponente evtl. die Schwere der psychischen Störung repräsentieren. Die weiteren Hauptkomponenten beinhalten dann oft die wirklich interessanten Charakteristika der untersuchten Beobachtungen, wie beispielsweise das Risikoverhalten einer Firma oder die Symptome der psychischen Störung.

2.7 PCA und Eigenwerte/Eigenvektoren

Wir haben hier weitgehend auf die der PCA zugrunde liegende Mathematik verzichtet und uns auf eine anschauliche Interpretation der Hauptkomponenten beschränkt. Schlägt man jedoch ein Lehrbuch zum Thema PCA auf, so wird oft von *Eigenwerten* und *Eigenvektoren* gesprochen. Für eine mathematischere Herleitung sei auf Kapitel A im Anhang verwiesen.

Um die Verbindung der bisherigen Abhandlung zum Thema PCA mit Eigenvektoren und Eigenwerten zu verstehen und auch das Kapitel 2.8 mit der Umsetzung der PCA in R nachvollziehen zu können, sollte man zumindest das Folgende wissen:

- Die **Eigenvektoren** $a_1 = (a_{11}, \ldots, a_{p1})$, ..., $a_p = (a_{1p}, \ldots, a_{pp})$ der $(p \times p)$ Kovarianz-Matrix der ursprünglichen p Variablen enthalten gerade die **Koeffizienten a_{jk}, die zur Bildung der Linearkombination der Hauptkomponenten** $Y_1, \ldots, Y_p$ benötigt werden (siehe Kapitel 2.2).
- Werden die Eigenvektoren spaltenweise zu einer Matrix zusammengefügt, so ergibt sich die Rotationsmatrix, durch welche das ursprüngliche Koordinatensystem ins Hauptkomponentensystem gedreht wird.
- Die zugehörigen **Eigenwerte** $\lambda_1, \ldots . \lambda_p$ geben gerade die **Varianzen der einzelnen Hauptkomponenten wieder**:

$$Var(Y_1) = \lambda_1, \ldots, Var(Y_p) = \lambda_p$$

2.8 PCA in R

Wir betrachten den Datensatz *dat*, welcher die Grösse, das Gewicht und die Schuhgrösse von 100 Personen enthält.

Wir führen die Hauptkomponentenanalyse zum besseren Verständnis erst Schritt für Schritt aus, bevor wir die in R vorimplementierten Funktionen benutzen.

Da die Koeffizienten der Linearkombination der ursprünglichen (zentrierten) Variablen zu den neuen Hauptkomponenten durch die Einträge der Eigenvektoren der Kovarianz Matrix gegeben sind, berechnen wir zunächst die Kovarianz Matrix:

```
round(cov(dat),2)
```

```
##         Groesse Gewicht Schuh
## Groesse   76.04  125.83 29.79
## Gewicht  125.83  257.72 49.22
## Schuh     29.79   49.22 16.36
```

Mit der Funktion `eigen` berechnen wir dann die Eigenwerte und Eigenvektoren. Wir erhalten zunächst unter `$values` die, der Grösse nach geordneten, Eigenwerte der Kovarianz Matrix:

```
Eigenwerte <- eigen(cov(dat))$values
round(Eigenwerte,2)
```

```
## [1] 332.50  13.77   3.86
```

Unter `$vectors` erhalten wir die Eigenvektoren, welche die Koeffizienten für die Linearkombination bei der Hauptachsentransformation enthalten (siehe Kapitel 2.2).

```
round(eigen(cov(dat))$vectors,2)
```

```
##       [,1]  [,2]  [,3]
## [1,] -0.45  0.75 -0.48
## [2,] -0.87 -0.48  0.07
## [3,] -0.18  0.45  0.87
```

Die Hauptkomponenten sind dann also:

$$PC1 = -0.45 \cdot Groesse - 0.87 \cdot Gewicht - 0.18 \cdot Schuhgrösse$$
$$PC2 = 0.75 \cdot Groesse - 0.48 \cdot Gewicht + 0.45 \cdot Schuhgrösse$$
$$PC3 = -0.48 \cdot Groesse + 0.07 \cdot Gewicht + 0.87 \cdot Schuhgrösse$$

Die durch die einzelnen Hauptkomponenten erklärten Anteile der Varianz können wir wie folgt bestimmen:

```
round(Eigenwerte/sum(Eigenwerte),2)
```

```
## [1] 0.95 0.04 0.01
```

Die erste Hauptkomponente erklärt also 95% der Totalvariation, die zweite 4%, die dritte 1%. Die

kumulierten Anteile sind:

```
round(cumsum(Eigenwerte)/sum(Eigenwerte),2)
```

```
## [1] 0.95 0.99 1.00
```

Die ersten beiden Hauptkomponenten erklären also zusammen 99% der Totalvariation.

R-Funktionen

In R gibt es in der Grundversion für die Hauptkomponentenanalyse die zwei Funktionen `princomp` und `prcomp`. `prcomp` ist numerisch stabiler und darum die bevorzugte Version. Falls es mehr Variablen als Beobachtungen gibt, kann nur mit der Funktion `prcomp` gearbeitet werden. `princomp` hat dafür etwas mehr Optionen: so kann zum Beispiel auch nur eine Korrelations- oder Kovarianzmatrix eingeben werden (`princomp(..., covmat=...)`).

Mit dem Befehl `prcomp` erhalten wir die folgende Ausgabe:

```
prcomp(dat)
```

```
## Standard deviations (1, .., p=3):
## [1] 18.234452  3.710686  1.964629
##
## Rotation (n x k) = (3 x 3):
##                PC1        PC2         PC3
## Groesse -0.4500603  0.7527626  0.48041047
## Gewicht -0.8749472 -0.4793353 -0.06859415
## Schuh   -0.1786426  0.4512053 -0.87435726
```

Vergleichen wir diese Ausgabe mit der Ausgabe der Eigenwerte und Eigenvektoren oben, so stellen wir fest, dass hier unter Rotation die Eigenvektoren stehen. Unter Standard deviations stehen die Standardabweichungen der Hauptkomponenten. Da die Varianzen der Hauptkomponenten den Eigenwerten entsprechen, sind die Standardabweichungen gerade die Quadratwurzeln aus den Eigenwerten.

In der Hilfe zu `prcomp` erfahren wir die weiteren Argumente dieser Funktion: `prcomp(x, retx=TRUE, center=TRUE, scale.=FALSE, tol=NULL)`

Das Argument `center=TRUE` bedeutet, dass standardmässig die Daten am Mittelwert zentriert werden, d.h. die Daten werden nach Null verschoben (ohne Änderung der Kovarianzstruktur). Setzt man das Argument `scale.=TRUE`, so werden die Daten so skaliert, dass sie danach eine Varianz von 1 haben, d.h. es wird mit der Korrelationsmatrix gerechnet.

Falls das Argument `retx=TRUE` (retx steht für return x) werden die rotierten Daten, d.h., die Hauptkomponenten berechnet.

Die Ergebnisse der prcomp-Funktion werden in eine Liste geschrieben. Diese Liste hat in diesem Fall die Komponenten `sdev`, `rotation` und `x`. Die Namen dieser Komponenten erfährt man in der R-Hilfe der Funktion unter Value. Nicht alle Komponenten der Liste werden in der Standardausgabe angezeigt. Es empfiehlt sich diese Liste in einem R-Objekt abzuspeichern, z.B.:

```
resultat = prcomp(dat)
```

Wenden wir die `summary`-Funktion auf das Objekt result an, so erhalten wir eine Übersicht, zu den Varianzen der einzelnen PCs sowie zum Anteil, den sie an der Totalvarianz haben:

```
summary(resultat)
```

```
## Importance of components:
##                             PC1     PC2     PC3
## Standard deviation     18.2345 3.71069 1.96463
## Proportion of Variance  0.9496 0.03933 0.01102
## Cumulative Proportion   0.9496 0.98898 1.00000
```

Diese Zahlen besagen, dass bereits mit der ersten Hauptkomponente über 90% der Varianz erklärt wird, mit den ersten beiden PCs fast 100% und mit allen drei PCs, wie es sein muss, genau 100% der Varianz abgedeckt ist.

Wir können auf die einzelnen Komponenten des PCA Ergebnisses mithilfe der Namen zugreifen. Z.B. erhalten wir die Standardabweichung der drei Hauptkomponenten wie folgt:

```
resultat$sdev
```

```
## [1] 18.234452  3.710686  1.964629
```

Mit `resultat$rotation` werden die Eigenvektoren der Kovarianz Matrix der ursprünglichen Feature ausgegeben, während wir mit `resultat$x` die Koordinaten der Beobachtungen bzgl. der Hauptkomponenten erhalten – also die score-Matrix Y. Wir geben hier nur die Scores der ersten fünf Beobachtungen wieder.

```
resultat$x[1:5,]
```

```
##               PC1        PC2        PC3
## [1,] -15.9231261  0.4076211 -1.0912546
## [2,]  -2.6052586  3.0141809  0.3304985
## [3,] -15.7179717 -1.7396384 -0.6044232
## [4,]  -0.9050592  4.2251432 -0.9368910
## [5,] -22.0869586 -8.4252362 -2.6693138
```

Diese Hauptkomponenten sind zentriert (haben jeweils einen Mittelwert 0) - das Argument center ist per default auf TRUE gesetzt.

Für die Analyse mit den skalierten Daten muss das Argument `scale` explizit auf `TRUE` gesetzt werden.

```
res_stand <- prcomp(dat, scale.=TRUE)
res_stand
```

```
## Standard deviations (1, .., p=3):
## [1] 1.6336556 0.4978775 0.2885953
##
## Rotation (n x k) = (3 x 3):
##                PC1        PC2        PC3
## Groesse -0.5942606  0.1486975 -0.7904071
## Gewicht -0.5753354  0.6081260  0.5469662
## Schuh   -0.5619996 -0.7797896  0.2758344
```

```
sum(res_stand$sdev^2)
```

```
## [1] 3
```

Die Summe der quadrierten Standardabweichungen ergibt hier direkt die Anzahl der Variablen. Da die originalen Daten unterschiedliche Einheiten aufweisen, ist die Analyse mit standardisierten Daten vorzuziehen.

Um die Anzahl der relevanten Hauptkomponenten zu analysieren, kann man den kumulierten Anteil an der totalen Varianz gegen die Anzahl Hauptkomponenten plotten oder das Scree-Diagramm (Varianzen/Eigenwerte in abnehmender Ordnung) verwenden.

```
par(mfrow=c(1,2), las=1, mar=c(4,4,2,1))
plot(cumsum(res_stand$sdev^2)/sum(res_stand$sdev^2), type="b",
     ylab="kumulativer Varianzanteil", xlab="Anzahl Hauptkomponenten",
     main="kumulierte Varianz", ylim=c(0.5,1))
abline(h=0.8, lwd=2, col="gray")
plot(res_stand$sdev^2, type="b",  ylim=c(0, 3), ylab="Varianz",
     xlab="k (k-te Hauptkomponente)", las=1, main="Scree-Plot")
abline(h=0, lty=2)
```

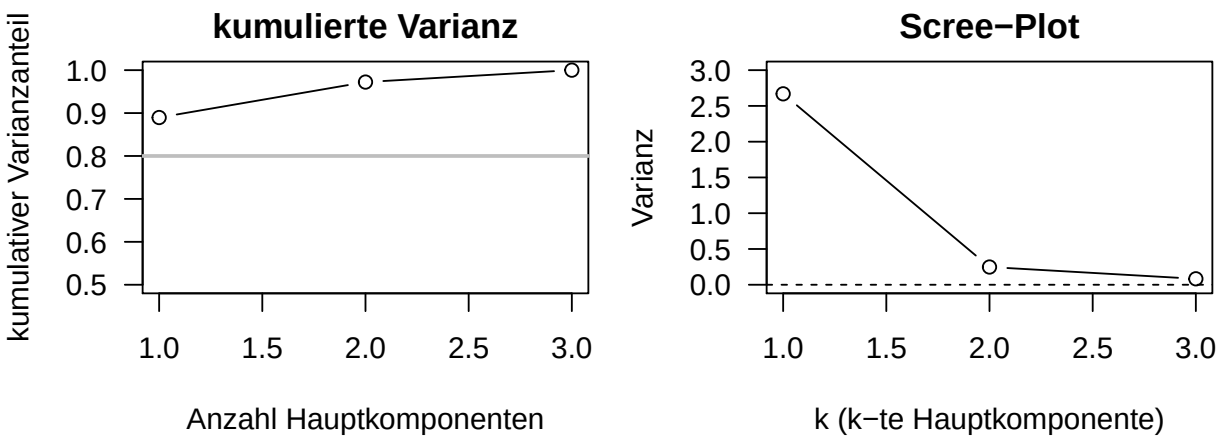

Abbildung 2.7: Kumulierter Anteilt an der totalen Varianz gegen die Anzahl Hauptkomponenten und Scree-Plot.

Darstellung der Hauptkomponenten und Biplot

Wollen wir die Beobachtungen im Koordinatensystem der ersten beiden Hauptkomponenten darstellen, so können wir dies mit folgendem plot-Befehl tun:

```
plot(resultat$x[,1], resultat$x[,2], xlab="PC1", ylab="PC2", las=1)
```

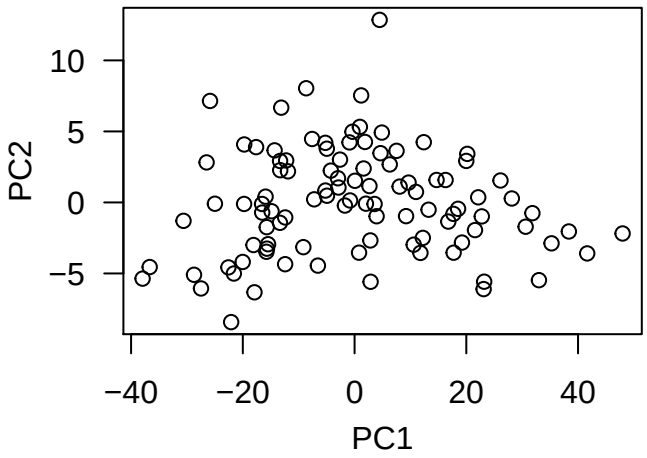

Abbildung 2.8: Scatterplot der ersten beiden Hauptkomponenten.

Eine alternative Darstellung bietet der so-genannte **Biplot**. Er beruht auf dem Streudiagramm der ersten beiden Hauptkomponenten. Standardmässig werden die Hauptkomponenten mit der Standardabweichung der entsprechenden Variable und der Wurzel der Anzahl Beobachtungen standardisiert. Die entsprechenden Skalen sind unter und links im Plot wiedergegeben. Eine Folge der Skalierung der Hauptkomponenten ist, dass der geometrische Abstand zweier Punkte im Streudia-

gramm proportional zur Mahalanobis-Distanz[1] der ursprünglichen Objekte ist. Die Skalierung ist aber nicht so entscheidend.

Diese Darstellung wird nun mit Vektoren (d.h. Pfeilen) ergänzt, die die Projektionen der ursprünglichen Merkmale in die Ebene der beiden Hauptkomponenten anzeigen. Wenn die ursprünglichen Variablen angemessen in die Ebene produziert werden können, ist die Länge eines Vektors proportional zur Varianz des entsprechenden Merkmals. D.h. wenn die Analyse mit standardisierten Daten durchgeführt wurde, sollten alle Vektoren ungefähr 1 lang sein. In diesem Fall kann auch der Winkel zwischen zwei Vektoren interpretiert werden. Dieser entspricht der Korrelation der beiden entsprechenden Merkmale: je kleiner der Winkel desto grösser die Korrelation. Bei einer schlechten Projektion der ursprünglichen Variablen, können auch die Winkel nicht wirklich interpretiert werden.

Für weitere technische Details wird auf die Literatur (z.B. Everitt und Hothorn, 2011) verwiesen.

Der biplot kann mit folgender R-Funktion erzeugt werden:

```
biplot(res_stand,  pic.biplot=TRUE, cex=0.7)
```

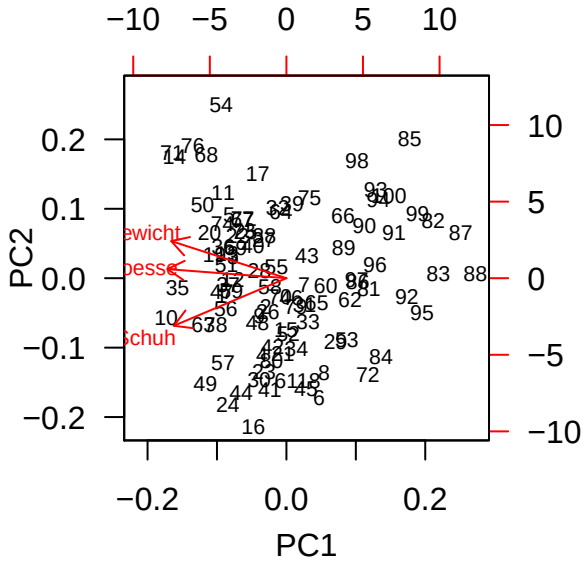

Abbildung 2.9: Biplot der standardisierten Daten.

Dem Biplot in Abbildung 2.9 können wir demnach entnehmen, dass die PC1 eine starke und negative Korrelation mit allen drei Variablen (Gewicht, Grösse, Schuhgrösse) hat – d.h. sie spiegelt so etwas wie die negative Gesamt-Grösse der Person wieder. Die PC2 ist positiv mit dem Gewicht korreliert, aber negativ mit Schuhgrösse.

[1] Distanzmass zwischen Punkten in einem mehrdimensionalen Raum. Dabei werden die unterschiedlichen Standardabweichungen entlang der Achsen des n-dimensionalen Raumes und auch die Korrelationen zwischen den einzelnen Achsen berücksichtigt.

2.9 Alternative Standardisierung und robuste Hauptkomponentenanalyse zur Detektion von Ausreissern

Da die PCA auf der Kovarianz- oder Korrelation-Matrix beruht und Varianzen und Korrelationen sensitiv bezüglich Ausreisser sind, kann **das Ergebnis einer PCA von Ausreissern stark verfälscht werden.**

Obwohl in der Praxis die Standardisierung meistens mit dem Mittelwert und der Standardabweichung durchgeführt werden, sollten auch robuste Versionen der Hauptkomponenten in Betracht gezogen werden. Z.B. der Median als Lagemass und der MAD als Skalenmass. Damit werden Ausreisser in den einzelnen Variablen besser sichtbar gemacht. Allerdings muss dann die Standardisierung selber vorgenommen und anschliessend eine Hauptkomponenten-Analyse auf der Kovarianzmatrix der robust standardisierten Variablen durchgeführt werden.

Ein zweiter Ansatz ist es, die Hauptkomponentenanalyse direkt mit robusten Schätzern durchzuführen. In R gibt es Pakete mit entsprechenden Funktionen, die das für uns tun.

Der Effekt einer robusten Hauptkomponentenanalyse kann an einem 2-dimensionalen Datensatz mit Ausreissern demonstriert werden (Abbildung 2.10). Während die erste Hauptkomonenten beim klassischen Ansatz stark durch die Ausreisser beeinflusst wird, wird beim robusten Ansatz nur die Streuung der Daten ohne Ausreisser berücksichtigt.

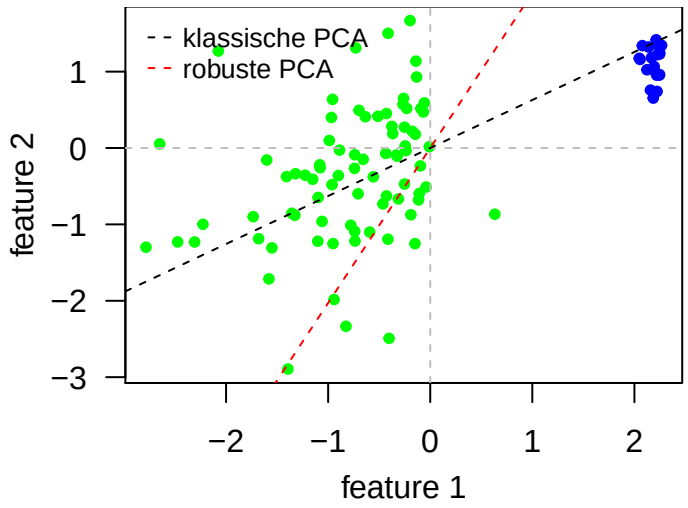

Abbildung 2.10: Hauptkomponentenanalyse mit klassischem und robustem Ansatz an einem 2-dimensionalen Datensatz mit Ausreisser (blaue Punkte). Zusätzlich zu den Beobachtungen sind jeweils die ersten Hauptkomponenten eingetragen.

Die Unterschiede sieht man auch, wenn man die ersten beiden Hauptkomponenten darstellt (Abbildung 2.11). Während in der robusten Version der Ursprung (0-Punkt) im Zentrum der Hauptgruppe liegt, liegt das Zentrum bei der klassischen Version zwischen den Datengruppen.

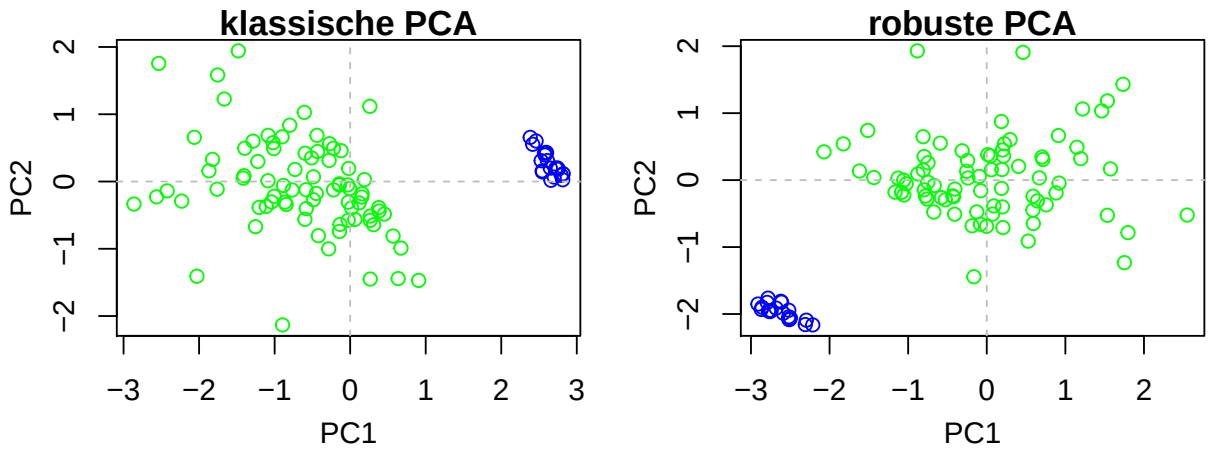

Abbildung 2.11: Score-Plot mit den ersten beiden Hauptkomponenten für klassische (links) und robuste (rechts) Hauptkomponentenanalyse.

Ausreisser-Detektion

Die robuste Analyse kann man verwenden, um Ausreisser in hochdimensionalen Daten zu detektieren. Dabei gibt es zwei Arten von Ausreissern: die Ausreisser im Hauptkomponenten-Raum und die orthogonalen Ausreisser.

Ausreisser in den Hauptkomponenten sind Beobachtungen, welche eine grosse Distanz zum Ursprung der Hauptkomponentenanalyse der berücksichtigten Komponenten haben. Diese Distanz nennt man Score-Distanz. Die Socre-Distanz SD für eine Beobachtungen i ergibt sich aus der Summe der quadrierten Score-Werte y der Komponenten, standardisiert mit den robust geschätzten Eigenwerten (λ).

$$SD_i = \sqrt{\sum_{j=1}^{k} \frac{y_{ij}^2}{\lambda_j}}$$

Die Anzahl der berücksichtigen Komponenten k muss dabei, zum Beispiel über den Scree-Plot, manuell festgelegt werden.

Je grösser die Distanz für eine Beobachtung ist, desto weiter weg liegt sie vom Zentrum, d.h. desto eher ist die Beobachtung ein Ausreisser. Projiziert man die Score-Distanz aller Beobachtungen gegen den entsprechenden Index, kann man die Ausreisser gut erkennen (Abbildung 2.12). Als Hilfe wird in die Abbildung standardmässig noch eine Cutoff-Line bei der Wurzel des 0.975-Quantil der Chi-Quadrat-Verteilung mit k Freiheitsgraden ($ch = \sqrt{\chi^2_{k,0.975}}$) eingezeichnet. Das muss aber nicht zwingend die beste Abgrenzung sein.

Orthogonale Ausreisser sind Punkte die im Raum der berücksichtigen Hauptkomponenten nicht richtig erfasst werden. Das heisst die originalen Punkte liegen weit entfernt von der modellierten

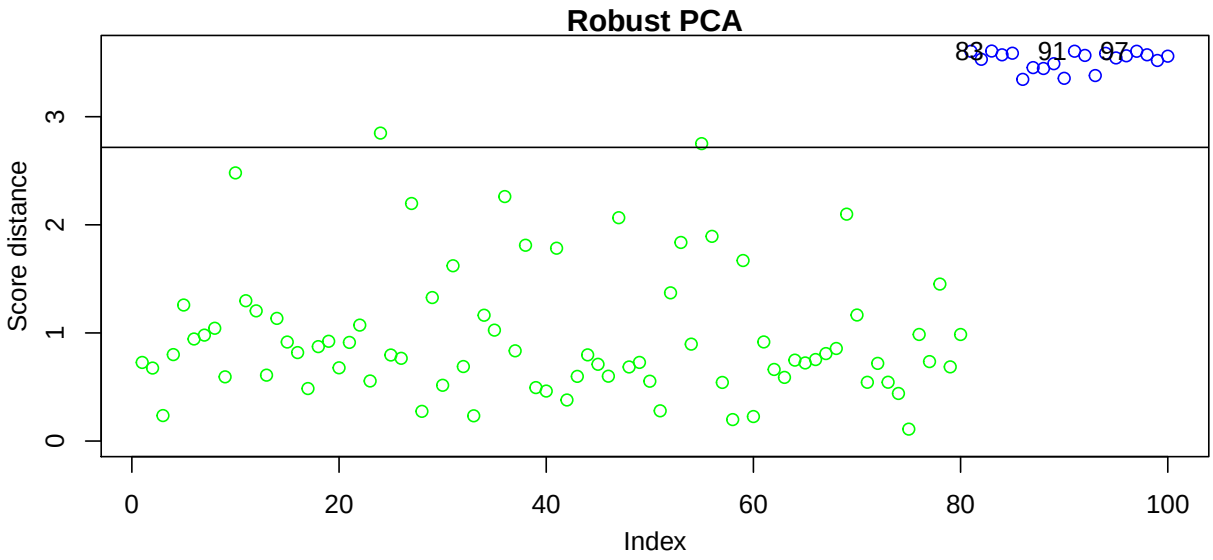

Abbildung 2.12: Score-Distanz aufgetragen gegen die Beobachtungen.

Hyperebene. Diese Punkte sind häufig fast noch interessanter. Die entsprechende Distanz bezeichnet man als orthogonale Distanz (Abbildung 2.13). In den Hauptkomponenten sind diese Punkte nicht unbedingt auffällig. Die orthogonale Distanz OD für eine Beobachtung i entspricht der Norm der ursprünglichen Koordinaten minus den robust geschätzten Mittelwerten $\hat{\mu}$, minus der Matrix der Loadings, mal die transponierte Score-Matrix y_i der berücksichtigten Hauptkomponenten:

$$OD_i = ||x_i - \hat{\mu} - P \cdot y_i^T||$$

In unserem 2-dimensionalen Beispiel ist die orthogonale Distanz bei 2 Hauptkomponenten gleich 0, da die Projektion keinen Fehler hat. Verwendet man nur eine Hauptkomponente, kann man die orthogonale Distanz berechnen und visualisieren (Abbildung 2.14).

Mit der klassischen Hauptkomponentenanalyse sind die entsprechenden Auswertungen weniger aussagekräftig, da die Ausreisser die Ergebnisse der zugrundeliegenden Hauptkomponentenanalyse stark verzehren können.

R-Befehle

Die Hauptkomponentenanalyse mit robusten Schätzern kann in R zum Beispiel mit der Funktion `PcaHubert` aus dem Paket `rrcov` durchgeführt werden. Als wichtiges Argument muss in dieser Funktion die Anzahl der zu berücksichtigen Hauptkomponenten k festgelegt werden. Zudem kann auch festgelegt werden, ob die Variablen zuerst standardisiert werden sollen oder nicht. Das PcaHuber-Objekt ist von der Klasse S4, d.h. Sie können verschiedene Attributen mit `@` ansprechen, ähnlich zu `$` in einem data.frame Objekte.

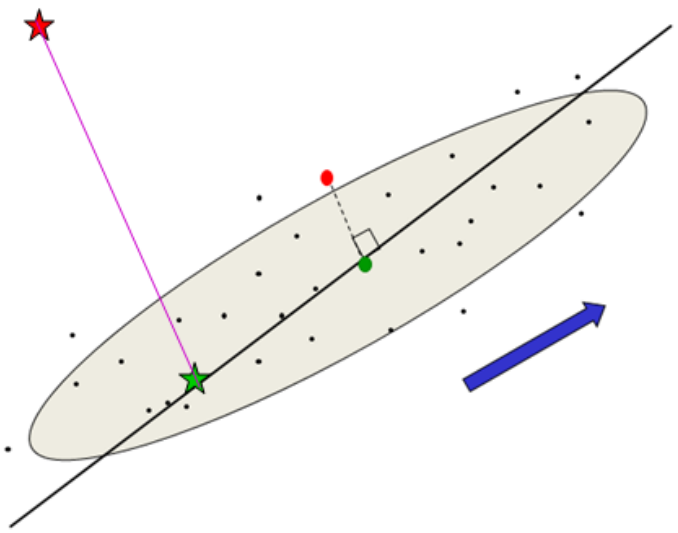

Abbildung 2.13: Die violette Linie zeigt die orthogonale Distanz zwischen dem ursprünglichen Datenpunkt (rot) und seiner Projektion (grün) in der Hauptkomponentenebene.

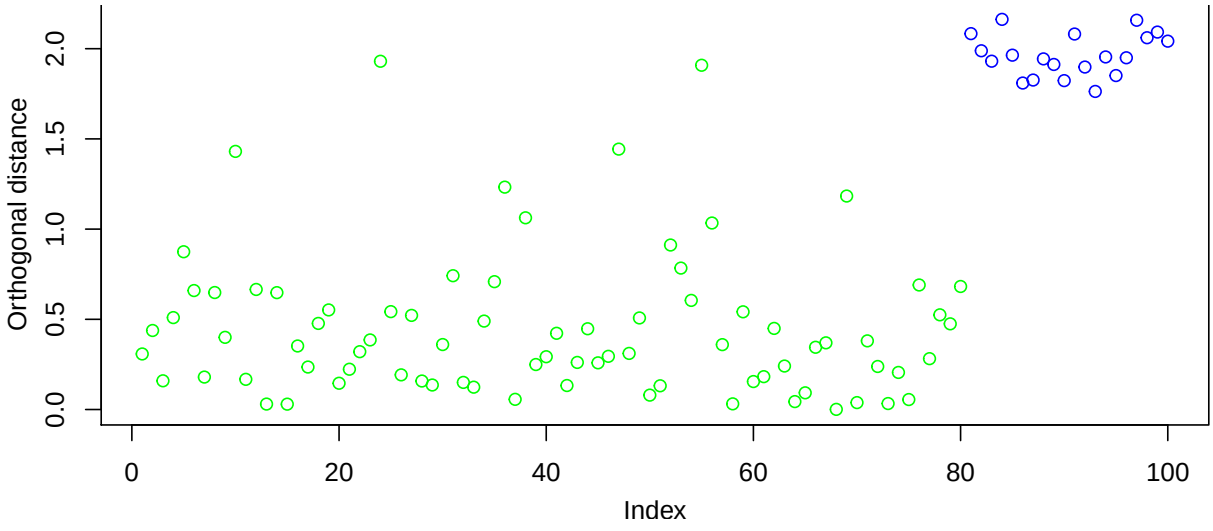

Abbildung 2.14: Orthogonale-Distanz zur ersten Hauptkomponente aufgetragen gegen die Beobachtungen.

```
library("rrcov")
pca_robust<-PcaHubert(dat,k=2, scale=FALSE)
pca_robust
```

```
## Call:
## PcaHubert(x = dat, k = 2, scale = FALSE)
##
## Standard deviations:
## [1] 1.2336941 0.7001735
```

```
summary(pca_robust)
```

```
##
## Call:
## PcaHubert(x = dat, k = 2, scale = FALSE)
## Importance of components:
##                            PC1    PC2
## Standard deviation     1.2337 0.7002
## Proportion of Variance 0.7564 0.2436
## Cumulative Proportion  0.7564 1.0000
```

```
plot(pca_robust@scores)
```

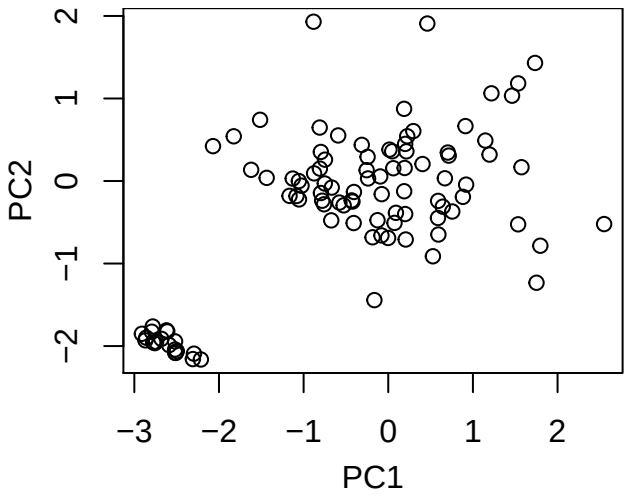

Abbildung 2.15: Score-Plot der ersten zwei robust berechneten Hauptkomponenten.

Auf die Score-Distanz kann man mit dem PCA-Objekt und `@sd`, auf die orthogonale Distanz mit `@od` zugreifen. Enthält das Datenobjekt mehr Dimension als in der robusten PCA berücksichtigt werden, erhält man mit der `plot()`-Funktion direkt den Scatterplot aus Score und orthogonale Distanz.

```
plot(pca_robust@sd)
```

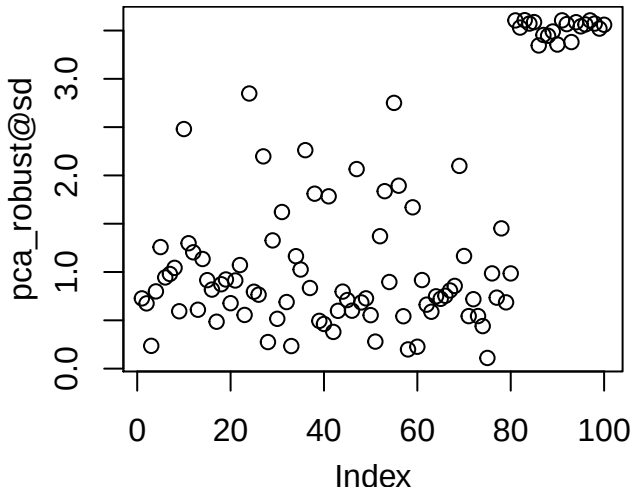

Abbildung 2.16: Score-Distanz aufgetragen gegen die Beobachtungen.

```
pca_robust2<-PcaHubert(dat,k=1, scale=FALSE)
plot(pca_robust2@od)
```

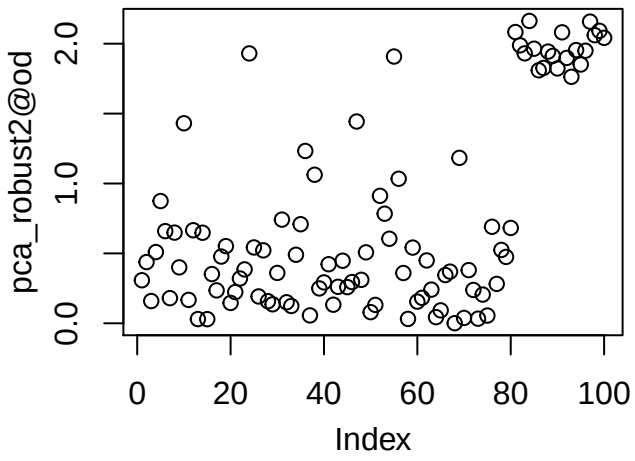

Abbildung 2.17: Orthognonale-Distanz zur ersten Hauptkomponente aufgetragen gegen die Beobachtungen.

```
plot(pca_robust2)
```

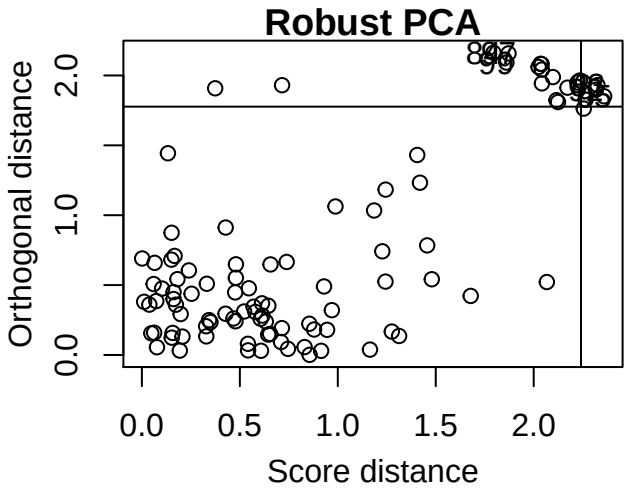

Abbildung 2.18: Score-Distanz gegen Orthognonale-Distanz.

2.10 Zusammenfassung

Eine Hauptkomponentenanalyse ist eine Drehung (orthogonale Transformation) des p-dimensionalen Raums der zentrierten Originalvariablen in das Hauptkomponentensystem. Die Hauptkomponenten werden so konstruiert, dass sie untereinander unkorreliert sind und in absteigender Weise einen möglichst grossen Teil der Totalvarianz abdecken. Damit wird erreicht, dass oft schon mit den ersten wenigen Hauptkomponenten ein Grossteil der Varianz abgedeckt ist und die wesentlichen Strukturen (wechselseitige Abstände bzw. Ähnlichkeiten der Beobachtungen sowie mögliche Gruppierungen) der Daten schon in der 2-dimensionanlen Darstellung im Koordinatensystem der ersten beiden Hauptkomponenten sichtbar werden.

Es ist oft sinnvoll, die wichtigsten Hauptkomponenten in weiterführenden Analysen (z.b. Regression) zu benutzen. Die robuste Hauptkomponentenanalyse kann auch zur Detetkion von Ausreissern in mulitvariaten Daten verwendet werden.

Es müssen keinerlei Voraussetzungen erfüllt sein, um die PCA anwenden zu können. Der Nutzen bzgl. einer Dimensionsreduktion ist aber besonders hoch, wenn die Originalvariablen stark korreliert sind. (In der Literatur finden Sie häufig eine Diskussion der PCA für den Fall von multivariat normalverteilten Daten. Für diesen Fall vereinfacht sich manches, aber die multivariate Normalverteilung ist keine Voraussetzung für die PCA!)

Die Hauptkomponentenanalyse hat einige Nachteile. Die Ergebnisse sind abhängig von der Skalierung und daher nicht eindeutig. Die Hauptkomponenten sind schwierig zu interpretieren. Die Interpretation ist daher subjektiv. Auch die Anzahl der auszuwählenden bedeutenden Hauptkomponenten ist (trotz Hilfsmittel wie der kumulierten Varianz und dem Scree-Plot) nicht eindeutig.

3 Ähnlichkeits- und Distanzmasse für Multidimensionale Skalierung

Multidimensionale Skalierung (MDS) ist der übergeordnete Begriff für Verfahren zur Entdeckung von Strukturen innerhalb von Beobachtungen. Dabei werden die Beobachtungen so dargestellt, dass die Distanz zwischen je zwei Punkten in der Visualisierungen einem gegebenen Nähe-, Abstands-, Ähnlichkeits- oder Unähnlichkeitsmass dieser Objekte so genau wie möglich entspricht. Je weiter die Objekte in der Visualisierung voneinander entfernt sind, desto unähnlicher sind sie und je näher sie beieinander liegen, desto ähnlicher sind sie.

Das zentrale Konzept, das dem MDS zugrunde liegt, ist die **Ähnlichkeit** respektive **Distanzen** zwischen zwei Objekten. Die entsprechenden Ähnlichkeits- und Distanzmasse werden wir auch beim Cluster brauchen (siehe Kapitel 4).

3.1 Ähnlichkeit

Ähnlichkeit ist schwer zu definieren, aber wie es die amerikanische Redensart ausdrückt, “We know it when we see it”. Schauen Sie sich als Beispiel die Ähnlichkeit von zwei “Tieren” in Abbildung 3.1 an.

Die eigentliche Bedeutung von Ähnlichkeit ist eine philosophische Frage. Im Data Mining müssen wir uns jedoch einem pragmatischen Ansatz zuwenden und bestimmen deshalb Ähnlichkeit aufgrund von (Objekt-) Merkmalen.

Abbildung 3.1: Ähnlichkeit ist schwer zu definieren, aber “We know it when we see it”. Quelle: Soman et al. (2006)

Diese Bestimmung erfolgt über so genannte Ähnlichkeits- bzw. Distanzmasse. Der Unterschied der beiden Masse liegt in der Interpretation: zwei Objekte sind umso ähnlicher, je grösser der Wert des Ähnlichkeitsmasses beziehungsweise je kleiner der Wert des Distanzmasses ist.

Ein **Ähnlichkeitsmass** s_{rt} zwischen den Objekten r und t quantifiziert die Übereinstimmung zwischen diesen beiden Objekten. Dabei fordert man, dass einerseits die Ähnlichkeit zwischen den Objekten r und t gleich der Ähnlichkeit zwischen den Objekten t und r ist (d.h. $s_{rt} = s_{tr}$) und andererseits die Ähnlichkeit zwischen den Objekten r und t kleiner gleich der Ähnlichkeit zu sich selber ist (d.h. $s_{rt} \leq s_{rr}$).

Technisch geläufiger sind uns **Distanzmasse**. Das wohl bekannteste Mass ist der euklidische Abstand zwischen zwei (geografischen) Punkten. Wie wir sehen werden, gibt es noch viele weitere sinnvolle Möglichkeiten, Distanzmasse zwischen Objekten zu definieren.

Ein Distanzmass $d(\dots)$, oder auch Distanzfunktion genannt, ist eine reellwertige Funktion mit den drei Eigenschaften

1. $d_{rt} \geq 0$ (d ist eine nicht-negative Funktion),
2. $d_{rr} = 0$ (die Distanz zu sich selber ist 0), und
3. $d_{rt} = d_{tr}$ (d ist symmetrisch)

Viele der verwendeten Distanzmasse haben die zusätzliche Eigenschaft, **metrische Distanzmasse** (auch Metrik genannt) zu sein. Diese erfüllen zusätzlich die zwei Bedingungen:

4. $d_{rt} = 0$ genau dann, wenn $r = t$, und
5. $d_{rt} \leq d_{rs} + d_{st}$ (Δ-Ungleichung).

Folglich ist die *metrische Distanz* restriktiver definiert als die *Distanz*; d.h. nicht jede Distanz ist auch eine metrische Distanz! Metrische Distanzmasse werden bevorzugt, weil sie unserer räumlichen Vorstellung entsprechen.

Distanzen und Ähnlichkeiten können sowohl direkt erhoben, als auch indirekt aus einer Datenmatrix berechnet werden. Eine direkte Erhebung liegt beispielsweise vor, wenn Personen Urteile über die Ähnlichkeit von Objektpaaren abgeben oder wenn die Ähnlichkeit zweier Objekte mit der Häufigkeit gleichgesetzt wird, mit der die Objekte verwechselt werden. Bei direkten Erhebungen genügen die Ähnlichkeiten und Distanzen oft nicht den obigen Eigenschaften: Ähnlichkeiten und Distanzen sind nicht immer symmetrisch, oder die Δ-Ungleichung ist verletzt.

Wenn eine Funktion verwendet wird, die aufgrund der Merkmale der Objekte Distanz- oder Ähnlichkeitswerte berechnet, spricht man von einer **indirekten Erhebungen** des Distanz- oder Ähnlichkeitsmasses. Die indirekt erhobenen Distanzen oder Ähnlichkeiten erfüllen die Eigenschaften (1) bis (3), wenn die entsprechenden Ähnlichkeits- respektive Distanzfunktionen geeignet gewählt werden.

Bei der Wahl des Distanzmasses sollte man die Datentypen, die zur qualitativen und quantitativen Beschreibung der Merkmale benutzt werden, berücksichtigen. Wir unterscheiden folgende Merkmalstypen

- **Kategorielle** (nominalskalierte) Variablen sind Merkmale, bei denen sich die verschiedenen möglichen Ausprägungen *nicht* in eine sinnvolle Reihenfolge bringen lassen. Beispiele aus dem Bereich der demographischen Variablen wären: Geschlecht, Haarfarbe, Muttersprache ..... Hat das Merkmal nur zwei Ausprägungen, so spricht man auch von einem binären Merkmal.
- **Ordinalskalierte** Merkmale sind kategorielle Merkmale, bei denen sich die verschiedenen Ausprägungen in eine sinnvolle Reihenfolge bringen lassen. Beispiele wären: Schulnoten, die verschiedenen Komfort-Klassen bei Hotels oder Flugtickets. Ordinale Daten entstehen auch, wenn man bei numerischen Ausprägungen Klassen bildet und nur die Klasse sich merkt. Entsprechend werden solche Daten auch klassierte Daten genannt. Beispiel: Variable `Einkommen` mit den Kategorien `CHF 500-999`, `CHF 1000-1499` usw.
- Bei **quantitativen** (metrischen) Merkmalen geben die Ausprägungen eine Intensität bzw. ein Ausmass wieder. Mit diesen Merkmalen kann man "rechnen". Liegen die Werte von quantitativen Merkmalen zwischen 0 und 1, oder zwischen 0% und 100%, so spricht man von *Anteilen* oder *Prozentzahlen*. Handelt es sich um nicht-negative ganze Zahlen, so nennt man diese *Anzahlen* oder *Zähldaten*. Für die Distanzmasse kann es manchmal wichtig sein, wenn die Untertypen berücksichtigt werden, obwohl es technisch nicht immer zwingend ist.

3.2 Quantititative Merkmale

Euklidische Distanz

Die Objekte r und t lassen sich durch numerische Vektoren $\vec{x}_r$ und $\vec{x}_t$ beschreiben. Die bekannteste Metrik zwischen den Objekten r und t ist die **euklidische Distanz** (auch L_2-Metrik genannt).

$$d_{rt} = \sqrt{\sum_{j=1}^{p} \left(x_r^{(j)} - x_t^{(j)}\right)^2}$$

.

Eine Erweiterung ist die **skalierte euklidische Distanz**.

$$d_{rt} = \sqrt{\sum_{j=1}^{p} w_j^2 \left(x_r^{(j)} - x_t^{(j)}\right)^2}$$

Dabei ist w_j ein geeignet gewähltes Gewicht (Skalierung) für jede Dimension (welche einer Variable entspricht). Häufig wird für w_j entweder der Kehrwert der empirischen Standardabweichung der Variablen ($w_j = \frac{1}{sd(\vec{x}^{(j)})}$) oder eine der robusten Formen (wie z.B. $w_j = \frac{1}{MAD(\vec{x}^{(j)})}$) verwendet.

Manchmal sieht man auch den Kehrwert des Bereichs, den die Variable abdeckt $w_j = 1/(\max(\vec{x}^{(j)}) - \min(\vec{x}^{(j)}))$. Ziel ist es dabei, alle Variablen bezüglich der Streuung gleich wichtig zu machen.

Sobald die Skala in mindestens einer Variablen geändert wird, wenn z.B. von der Einheit *Metern* auf die Einheit *Kilometer* gewechselt wird, wird sich die Rangordnung der Beobachtungen bezüglich ihrer (unskalierten) euklidischen Distanzen ändern. Bsp.: Seien die Daten für die Objekte A, B und C gegeben durch

$$\begin{array}{c} \\ A \\ B \\ C \end{array} \overbrace{\begin{pmatrix} 4 & 50 & 12 \\ 6 & 20 & 19 \\ 3 & 70 & 17 \end{pmatrix}}^{x^{(1)}\ x^{(2)}\ x^{(3)}}$$

Dann ist die Distanz zwischen A und B 30.87, zwischen A und C 20.64 und zwischen B und C 50.13. Dividiert man die mittlere Spalte durch 100, so erhalten wir für die Distanz zwischen A und B 7.29, zwischen A und C 5.10 und zwischen B und C 3.64. Wir haben also nicht nur unterschiedliche Zahlen sondern auch eine unterschiedliche Rangordnung erhalten.

Um solche Abhängigkeiten von Skalenänderungen zu verhindern, wurde die skalierte euklidische Distanz vorgeschlagen. Allgemeiner ausgedrückt: Falls eine Variable eine viel grössere Streuung als alle anderen hat, so wird diese Variable (oder Dimension) wie oben gesehen die (unskalierten) euklidischen Distanzen wesentlich bestimmen. Manchmal ist dies ein gewünschter Effekt und manchmal ist dies störend. Trifft das zweite zu, so wechselt man zu den skalierten euklidischen Distanzen. Falls die Streuung einer Variablen durch Ausreisser dominiert ist, wird $w_j = 1/sd(\vec{x}^{(j)})$ die Ausreisser "verstecken". Wenn das nicht erwünscht ist (wohl meisten), dann sollte man die robusten Versionen, wie $1/MAD(\vec{x}^{(j)})$ oder 1 über die Quartilsdifferenz von $\vec{x}^{(j)}$, verwenden.

Manhattan Distanz

Eine weitere Möglichkeiten ist die Manhattan-Metrik.

$$d_{rt} = \sum_{j=1}^{p} |x_r^{(j)} - x_t^{(j)}|$$

oder die standardisierte Form

$$d_{rt} = \sum_{j=1}^{p} w_j \cdot |x_r^{(j)} - x_t^{(j)}|$$

Für hoch-dimensionale Daten funktioniert diese Distanz allenfalls besser. Da hier einzelne grosse Abstände nicht quadratisch in die Distanz eingehen.

{Bemerkung zu korrelierten Merkmalen:} Sind zwei Merkmale hoch korreliert (grösser als 0.9), dann wird die Charaktereigenschaft der Objekte, die durch diese zwei Merkmale beschrieben wird, in der Distanz stärker berücksichtigt, weil sie eben *zweimal* gemessen wurde. Das kann ein gewünschter

Effekt sein oder eben nicht. Auf jeden Fall muss man sich dessen bewusst sein!

Von Ähnlichkeiten zu Distanzen

Die in diesem Kurs vorgestellten Verfahren basieren alle auf Distanzangaben. Haben wir Ähnlichkeitsangaben, so ergibt sich die Herausforderung, diese in Distanzangaben zu überführen. Dafür gibt es verschiedenste Vorschläge, wie z.B.

- $d_{rt} = 1 - s_{rt}$, sofern $s_{rt} \in [-\infty, 1]$
- $d_{rt} = \sqrt{2(1 - s_{rt})}$, sofern $s_{rt} \in [-1, 1]$
  Dieser Vorschlag kann verallgemeinert werden falls alle $s_{rt} \geq 0$: $d_{rt} = \sqrt{s_{rr} - 2s_{rt} + s_{tt}}$
- $d_{rt} = \sqrt{2(1 - s_{rt}^2)}$, sofern $s_{rt} \in [-1, 1]$

3.3 Kategorielle und ordinalskalierte Merkmale

Binäre (dichotome) Daten

Angenommen, die p Merkmale von zwei Objekten r und t können je nur zwei Werte (**binäre Merkmale**) annehmen wie z.B. ja - nein, + - -, 0 - 1, usw. Wie ähnlich sind dann die beiden folgenden binären Sequenzen

$$\begin{aligned} r &= 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ t &= 0\,0\,0\,0\,0\,0\,1\,0\,0\,1 \end{aligned}$$

Ähnlichkeitsmasse zwischen den zwei Objekten r und t beruhen auf folgenden vier Auszählungsmöglichkeiten, die in einer 2×2 Tabelle zusammengestellt werden kann:

		Objekt r		
		0	1	
Objekt t	0	$M_{00} = 7$	$M_{01} = 1$	$M_{0\cdot} = 8$
	1	$M_{10} = 2$	$M_{11} = 0$	$M_{1\cdot} = 2$
		$M_{\cdot 0} = 9$	$M_{\cdot 1} = 1$	$M_{\cdot\cdot} = 10$

Die Gesamtanzahl $M_{\cdot\cdot}$ ist gerade identisch zur Anzahl betrachteten Merkmale p.

Die beiden bekanntesten Vorschläge für Ähnlichkeits-, respektive Distanzangaben für binäre Daten sind der **Simple-Matching-** und der **Jaccard-Koeffizient**.

Der **Simple-Matching-Koeffizient** ist der Anteil der Übereinstimmung in den Variablen, also $s(r,t) = (M_{00} + M_{11})/M_{\cdot\cdot}$. Die Distanz $d(r,t) = 1 - s(r,t) = (M_{01} + M_{10})/M_{\cdot\cdot}$ ist unter dem Namen **Canberra-Metrik**, ein metrisches Distanzmass, bekannt. Es spielt keine Rolle was mit 1 und was mit 0 codiert wird. Das Mass ist dann sinnvoll, wenn beide Levels ähnlich häufig sind. Wir sprechen deshalb von einer symmetrischen binären Variablen.

In unserem Beispiel mit dem Vergleich von zwei binären Sequenzen erhalten wir einen Simple-Matching-Koeffizienten von

$$s(r,t) = \frac{7+0}{10} = 0.7$$

oder eine Canberra-Metrik von

$$d(r,t) = \frac{2+1}{10} = 0.3$$

Der **Jaccard-Koeffizient** ist $s(r,t) = M_{11}/(M_{01} + M_{10} + M_{11})$. Die Distanz $d(r,t) = 1 - s(r,t) = (M_{01} + M_{10})/(M_{01} + M_{10} + M_{11})$ wird **Jaccard-Metrik** genannt und ist ebenfalls ein metrisches Distanzmass.

Im Gegensatz zum Simple-Matching-Koeffizienten wird beim Jaccard-Koeffizient die binären Werte asymmetrisch behandelt, indem nur der Anteil der 1-1 Übereinstimmungen berücksichtigt werden. Dieses Ähnlichkeitsmass wird angewandt, wenn nur eine 1-1 Übereinstimmung sinnvoll ist, hingegen eine 0-0 Übereinstimmung schlecht oder gar nicht interpretiert werden kann. Zum Beispiel wenn Menschen gruppiert werden sollen, ist das Merkmal, dass zwei Personen altgriechisch lesen können, ein starker Hinweis auf ihre Ähnlichkeit im Gegensatz zur Abwesenheit dieses Merkmals. Das Vorhandensein eines solchen Merkmals wird immer mit 1 codiert. Beim Jaccard-Koeffizienten wird an Stelle von $M_{..}$ nur durch $M_{..} - M_{00}$ dividiert.

In unserem Beispiel erhalten wir einen Jaccard-Koeffizienten von

$$s(r,t) = \frac{0}{2+1+0} = 0$$

oder eine Jaccard-Metrik von

$$d(r,t) = \frac{2+1}{2+1+0} = 1$$

Gemäss diesem Mass sind die beiden binären Sequenzen also völlig verschieden.

Mehrstufig nominal- und ordinalskalierte Daten

Falls die kategoriellen Daten aus mehr als zwei ungeordneten Kategorien bestehen (nominale Daten), dann ist eine nahe liegende Definition für Ähnlichkeit die Anzahl der Übereinstimmungen in den p Merkmalen (Dimensionen) dividiert durch p. Dieses Mass ist unter dem Namen **Sneath-Übereinstimmungskoeffizient** bekannt.

Der Sneath-Übereinstimmungskoeffizient wird üblicherweise mit $(1 - s_{rt})$ in eine Distanz überführt. Diese misst die Anzahl der Nichtübereinstimmungen. In der Nachrichtentechnik wird dieses Mass ebenfalls zur Bestimmung der Unterschiedlichkeit von gleichlangen Zeichenketten eingesetzt und ist unter dem Namen **Hamming-Abstand** bekannt.

Falls die Kategorien geordnet sind, kann man den Kategorien zum Beispiel die Werte 1 bis K, K ist

die Anzahl der Kategorien, zuordnen (Wert in $x_r^{(j)}$) und mit den neuen Variablen

$$z_r^{(j)} \stackrel{def}{=} \frac{x_r^{(j)} - 1}{K - 1} \in [0, 1]$$

wie mit quantitativen Daten arbeiten. Mit dieser Transformation lassen sich ordinal-skalierte Variablen mit verschiedener Anzahl Stufen problemlos vergleichen.

3.4 Mischungen von Variablentypen

In praktischen Problemstellungen wird man oft sowohl qualitative als auch quantitative Merkmale antreffen. Obwohl dieser Fall sehr wichtig ist, sind die in dieser Situation angebotenen Distanz- oder Ähnlichkeitsmasse (noch) sehr rudimentär. Ein möglicher Ansatz ist, für jedes Merkmal einen separaten Distanzwert zu bestimmen und dann den Mittelwert als Distanzmass für zwei Objekte zusammenzufassen. Wird auch die Möglichkeit berücksichtigt, dass sich zwei Merkmale nicht vergleichen lassen[2], so kann man folgendermassen vorgehen:

- Bestimme die Distanz zwischen Objekt r und t bezüglich ihres j-ten Merkmals: $d_{rt}^{(j)}$. Stellen Sie dabei sicher, dass $0 \leq d_{rt}^{(j)} \leq 1$.
- Halte in $\delta_{rt}^{(j)}$ fest, ob sich die Objekt r und t bezüglich des j-ten Merkmals vergleichen lassen:

$$\delta_{rt}^{(j)} = \begin{cases} 1 & \text{falls der Vergleich möglich ist} \\ 0 & \text{sonst.} \end{cases}$$

- Die verschiedenen Distanzmasse für jedes Merkmal werden nun mit

$$d(r, t) = \frac{\sum_{j=1}^{p} \delta_{rt}^{(j)} d_{rt}^{(j)}}{\sum_{j=1}^{p} \delta_{rt}^{(j)}}$$

  zu einer Distanzangabe zwischen den Objekten r und t zusammengeführt. Durch diese Konstruktion liegt $d(r, t)$ zwischen 0 und 1.

Dieser Ansatz ist auch unter dem Namen **Distanzmass von Gower** bekannt. Der Ansatz kann noch weiter verallgemeinern werden und Gewichte für jedes Merkmal zulassen.

3.5 R-Funktion

Um die Distanzmatrix eines Datensatz mit numerischen Grössen in R zu berechnen, gibt es die Funktion `dist()`. Mit dem Argument `method` kann man dabei verschiedene Distanzmasse berechnen (z.B. `euclidean`, `manhattan` oder `binary`). Für Details zur Funktion wird auf die R-Hilfe verwiesen.

[2] Ein Vergleich kann nicht möglich sein, wenn Informationen fehlen oder im Fall von dichotomen Variablen eine Charakteristik nicht bei beiden Objekten vorhanden ist, wie z.B. die Flügellänge bei Albatrossen, respektive bei Elefanten.

```
A <- data.frame(V1=c(5,3,1), V2=c(1, 0, 4), V3=c(2, 4, 2))
A
```

```
##   V1 V2 V3
## 1  5  1  2
## 2  3  0  4
## 3  1  4  2
```

```
dist(A, method="euclidean")
```

```
##          1        2
## 2 3.000000
## 3 5.000000 4.898979
```

```
dist(A, method="manhattan")
```

```
##   1 2
## 2 5
## 3 7 8
```

```
dist(A, method="manhattan", diag=TRUE, upper=TRUE)
```

```
##   1 2 3
## 1 0 5 7
## 2 5 0 8
## 3 7 8 0
```

Um Distanzen von Objekten mit verschiedenen Variablentypen zu berechnen, gibt es die R-Funktion `daisy()` aus dem Paket `cluster`. In diese ist auch der Ansatz von Gower implementiert (`metric="gower"`). Für die die einzelnen Merkmalstypen stehen dabei folgende Ansätze verwendet:

- Merkmal (Variable) ist symmetrisch binär oder nominal:

$$d_{rt}^{(j)} = \begin{cases} 0 & \text{falls } x_r^{(j)} = x_t^{(j)} \\ 1 & \text{sonst.} \end{cases}$$

- Merkmal (Variable) ist numerisch:

$$d_{rt}^{(j)} = \frac{\left| x_r^{(j)} - x_t^{(j)} \right|}{\max_s(x_s^{(j)}) - \min_s(x_s^{(j)})}$$

- Merkmal ist ordinal: Bildet

$$z_r^{(j)} = \frac{x_r^{(j)} - 1}{K^{(j)} - 1}$$

wobei $K^{(j)}$ die Anzahl Kategorien im j-ten Merkmal ist und behandelt die transformierten Grössen anschliessend wie numerische Merkmale: $d_{rt}^{(j)} = |z_r^{(j)} - z_t^{(j)}|$.

Einzelne Variablen können dabei zusätzlich mit dem Argument `type = list()` in der Form einer Liste angepasst werden (siehe Beispiel im R-Code unten). So können binäre Merkmale als asymmetrisch definiert werden, dadurch werden 0-0 Übereinstimmungen nicht berücksichtigt (vgl. Jaccard-Koeffizient).

Bei einer numerischen Variable kann man entweder verlangen, dass sie zuerst logarithmiert wird (`logratio`) oder dass sie wie eine ordinale Variable behandelt werden soll (`ordratio`). Der zweite Ansatz wird u.a. dann eingesetzt, wenn nicht klar ist, ob das Merkmal intervall- oder verhältnisskaliert ist oder wenn über die Qualität der Messung Zweifel bestehen.

Als Beispiel betrachten wir den Datensatz `flower` aus dem R-Paket `cluster`. Darin sind von 18 populären Blumenarten acht Eigenschaften festgehalten. Es handelt sich dabei um:

Name	Typ	Kurzbeschreibung
V1	binär	Ist die Pflanze winterhart oder nicht?
V2	binär	Braucht die Pflanze einen Schattenplatz?
V3	binär	Hat die Pflanze eine Knollenwurzel oder nicht?
V4	nominal	Farbe der Blütenblätter: (1 = weiss, 2 = gelb, 3 = pink, 4 = rot, 5 = blau)
V5	ordinal	Wächst die Pflanze an trockenen (1), normalen (2) oder feuchten (3) Standorten
V6	ordinal	Vorlieben einer Testgruppe; rangiert von 1 bis 18
V7	numerisch	Höhe der Pflanze in cm.
V8	numerisch	Empfohlene Setzdistanz zwischen Pflanzen in cm.

Die ersten beiden Merkmale können als symmetrisch binär angesehen werden. Das dritte Merkmal ist asymmetrisch binär, da zwei Pflanzen mit Knollenwurzeln sicher eine Gemeinsamkeit haben, jedoch wenn beide keine Knollenwurzeln haben, dann können sie ganz unterschiedlich wachsen. Das vierte Merkmal ist offensichtlich nominal mit fünf Stufen. Beim fünften Merkmal kann man die Stufen als geordnet gemäss der bevorzugten zunehmenden Bodenfeuchtigkeit sehen. Die Testgruppe hat im sechsten Merkmal ihre Vorliebe in einer Rangierung ausgedrückt, was identisch zu einer ordinalen Skalierung ist. Die beiden letzten Merkmale sind numerisch oder, etwas genauer, verhältnisskaliert.

```
library(cluster)
str(flower)
```

```
## 'data.frame':    18 obs. of  8 variables:
##  $ V1: Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 2 2 ...
##  $ V2: Factor w/ 2 levels "0","1": 2 1 2 1 2 2 1 1 2 2 ...
##  $ V3: Factor w/ 2 levels "0","1": 2 1 1 2 1 1 1 2 1 1 ...
##  $ V4: Factor w/ 5 levels "1","2","3","4",..: 4 2 3 4 5 4 4 2 3 5 ...
```

```
##  $ V5: Ord.factor w/ 3 levels "1"<"2"<"3": 3 1 3 2 2 3 3 2 1 2 ...
##  $ V6: Ord.factor w/ 18 levels "1"<"2"<"3"<"4"<..: 15 3 1 16 2 12 13 7 4 14 ...
##  $ V7: num  25 150 150 125 20 50 40 100 25 100 ...
##  $ V8: num  15 50 50 50 15 40 20 15 15 60 ...
```

```
f.dis <- daisy(flower, type=list(asymm=3))
str(f.dis)
```

```
##  'dissimilarity' num [1:153] 0.888 0.527 0.352 0.412 0.227 ...
##  - attr(*, "Size")= int 18
##  - attr(*, "Metric")= chr "mixed"
##  - attr(*, "Types")= chr [1:8] "N" "N" "A" "N" ...
```

```
## Reduktion des Datensatzes zur Visualisierung
f.dis1 <- daisy(flower[1:5,], type=list(asymm=3))
f.dis1
```

```
## Dissimilarities :
##           1         2         3         4
## 2 0.9701923
## 3 0.6118590 0.5904762
## 4 0.4169872 0.5698718 0.5865385
## 5 0.4256410 0.7952381 0.5095238 0.7176282
##
## Metric :  mixed ;  Types = N, N, A, N, O, O, I, I
## Number of objects : 5
```

```
f.dis2 <- daisy(flower[1:5,], type=list(asymm=3, ordratio=c(7,8)))
f.dis2
```

```
## Dissimilarities :
##           1         2         3         4
## 2 0.9437500
## 3 0.5854167 0.5904762
## 4 0.3833333 0.5770833 0.5937500
## 5 0.4520833 0.7952381 0.5095238 0.7104167
##
## Metric :  mixed ;  Types = N, N, A, N, O, O, T, T
## Number of objects : 5
```

3.6 Metrische multidimensionale Skalierung

In der **metrischen MDS** wollen wir aufgrund der gegebenen Distanzen zwischen den Objekten eine Konfiguration von Punkten in der zweidimensionale Ebene suchen, sodass die Punkte die Objekte repräsentieren und der Abstand (d.h. die euklidischen Distanzen) möglichst nahe bei den gegebenen Distanzen liegen. Das geschieht mittels linearer Algebra. Für Details wird auf weiterführende Literatur verwiesen: z.B. Everitt und Hothorn (2011).

Wenn metrische Distanzen verwendet werden, ist die metrische MDS äquivalent zur Hauptkomponentenanalyse. Im Unterschied zur Hauptkomponentenanalyse kann MDS auch für kategorielle Variablen verwendet werden. Das kann aber zu gewissen Verzerrungen führen.

In R kann man dazu die Funktion `cmdscale()` verwenden. Das Resultat-Objekt kann anschliessend mit der `plot`-Funktion visualisiert werden. Für weitere Optionen wird auf die Hilfefunktion verwiesen.

```
res.cmd <- cmdscale(f.dis)
plot(res.cmd)
text(res.cmd, labels=1:18, pos=4)
```

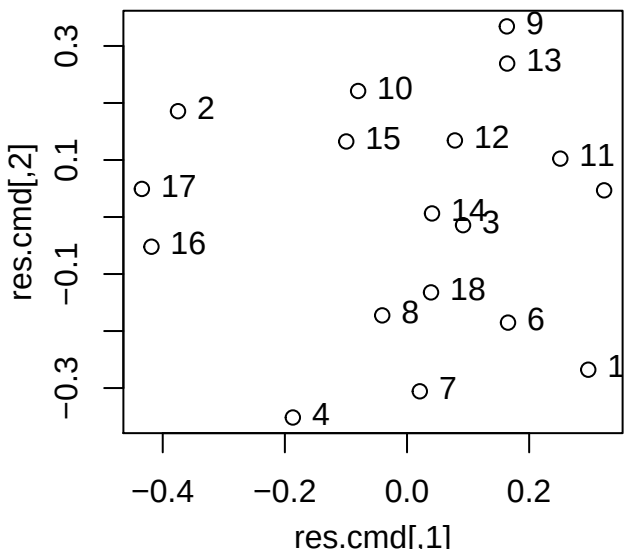

Abbildung 3.2: Darstellung der Flower-Daten mittels metrischer MDS.

Die Konfiguration kann verschoben, rotiert oder gespiegelt werden ohne dass sich die Distanzen ändern.

3.7 Ordinale multidimensionale Skalierung

Bei der **ordinalen multidimensionalen Skalierung** wird, im Unterschied zur metrischen MDS, nicht versucht, Distanzen in einer niedrigen Dimension darzustellen, sondern nur die **Ränge**. Das

ist insbesondere dann sinnvoll, wenn die Distanzen nicht metrisch sind und die absoluten Werte nicht bedeutungsvoll sind.

In R kann man dazu die Funktion `isoMDS()` aus dem Paket `MASS` verwenden. Das Resultat kann anschliessend visualisiert werden. Details zur Funktion siehe R-Hilfe.

```
library(MASS)
res.iso <- isoMDS(f.dis)
plot(res.iso$points)
text(res.iso$points, labels=1:18, pos=4)
```

```
## initial  value 28.042747
## iter   5 value 21.179372
## final  value 20.995880
## converged
```

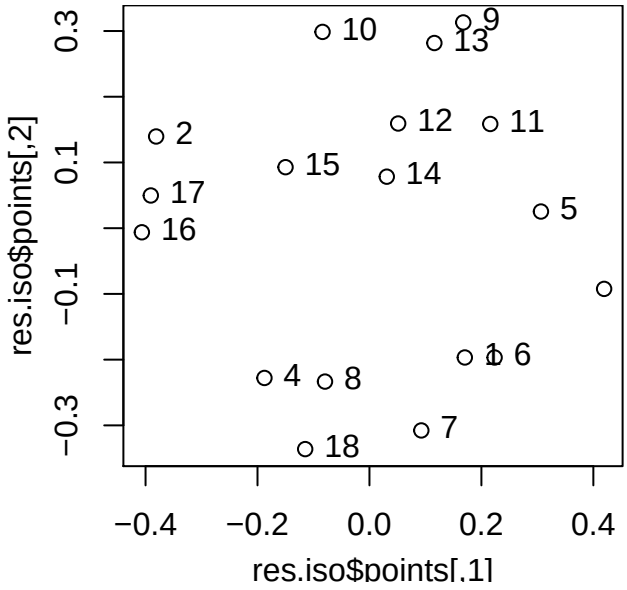

Abbildung 3.3: Darstellung der Flower-Daten mittels ordinaler MDS.

3.8 t-distributed Stochastic Neighbour Embedding (t-SNE)

T-distributed Stochastic Neighbor Embedding (t-SNE) ist ein weiterer Ansatz zur Dimensionsreduktion. Entwickelt wurde er von Laurens van der Maaten und Geoffrey Hinton (2008). Im Unterschied zur PCA und zur multidimensionalen Skalierung liegt der Fokus von t-SNE auf der Erhaltung der lokalen Strukturen.

Wichtig ist zu verstehen, dass bei **t-SNE nur Nachbarschaften modelliert werden, Distanzen zwischen Clustern haben keine Bedeutung**. Der Ansatz eignet sich auch, wenn die hochdimensionalen Daten nicht in einer Ebene liegen.

Konkret wird jedes hochdimensionale Objekt durch eine zwei- (oder drei-) dimensionalen Projektion so modelliert, dass ähnliche Objekte mit hoher Wahrscheinlichkeit durch benachbarte Punkte und ungleiche Objekte durch entfernte Punkte dargestellt werden. Dazu werden für ein Objekt die Distanz zu allen anderen Punkten im hochdimensionalen Raum bestimmt. Diese Distanzen werden in einer Normalverteilung, zentriert um das Zentrum, aufgetragen. Die Varianz der Normalverteilung ist dabei abhängig von der Dichte der Punkte um das Objekt. Die Dichtewerte bei den entsprechenden Distanzen entsprechen den unskalierten Ähnlichkeiten. Anschliessend werden die Ähnlichkeiten noch skaliert, so dass sie zusammen 1 ergeben. Allgemein, je ähnlicher ein Punkt ist, desto grösser ist der Ähnlichkeitswert. Diese Schritte werden für alle Objekte durchgeführt. Die Ähnlichkeit zwischen zwei Objekten muss dabei nicht in beide Richtungen gleich sein. Für das Erstellen der Ähnlichkeitsmatrix wird jeweils der Mittelwert der beiden Richtungen verwendet. Bei der Projektion werden die Punkte zuerst zufällig in die niedrige Dimension projiziert. Darauf basierend wird die Ähnlichkeit der Punkte in der niedrigen Dimension berechnet. Im Unterschied zu den hochdimensionalen Daten werden die Distanzen dabei in eine t-Verteilung (etwas breiter als eine Normalverteilung) eingezeichnet. Die t-Verteilung hilft Cluster von Daten besser aufzutrennen. In einem iterativen Prozess werden die Punkte in der niedrigen Dimension dann so verschoben, dass die Kullback-Leibler-Divergenz [3] zwischen den beiden Verteilungen minimiert wird, d.h. die Unterschiede in den Ähnlichkeitsmatritzen der originalen Daten und der Projektion werden minimiert.

In R gibt es die Funktion `Rtsne` aus dem Paket `Rtsne`. Standardmässig wird t-SNE auf die ersten 50 Hauptkomponenten der Daten angewendet (`PCA=TRUE` und `initial_dims=50`), man kann die Funktion aber auch direkt auf die Daten oder eine Distanzmatrix (`is_distance=TRUE`) anwenden. Wichtig sind insbesondere die Argumente `max_iter` (Anzahl Iterationen) und `perplexity` (siehe auch http://distill.pub/2016/misread-tsne/). Die Anzahl Iterationen sollte genügend gross sein, damit wir eine stabile Konfiguration erhalten. Das Argument `perplexity` entspricht grob der Anzahl zu erwartenden Nachbaren. Die Standardeinstellung für `perplexity` ist 10. Allgemein ist bei einem grösseren / dichteren Datensatz eine grössere Perplexity erforderlich. Typische Werte für die Perplexity liegen zwischen 3 und 50. Entsprechende Werte kann man testen. Wenn man zum Beispiel einen seltsamen "Ball" mit gleichmässig verteilten Punkten erhält, deutet das darauf hin, dass die Perplexity zu hoch gewählt wurde. Alle Punkte streben dann nach einem äquidistanten Abstand.

Standardmässig wird eine schneller Barnes Hut Implementation (Argument `theta=0.5`) verwendet. Für eine exakte Berechnung kann `theta=0` gesetzt werden.

```
library(Rtsne)
r_tsne <- Rtsne(flower, PCA=TRUE, perplexity=3, max_iter=3000)
d_tsne <-  as.data.frame(r_tsne$Y)
names(d_tsne) <-  c("tsne1","tsne2")
d_tsne$label <-  1:18
```

[3] Mass für den Unterschiedlich zweier Wahrscheinlichkeitsverteilungen

```
plot(x=d_tsne$tsne1, y=d_tsne$tsne2,pch=20, xlab="tsne1", ylab="tsne2")
text(x=d_tsne$tsne1, y=d_tsne$tsne2, labels = d_tsne$label, pos=3, cex=0.7)
```

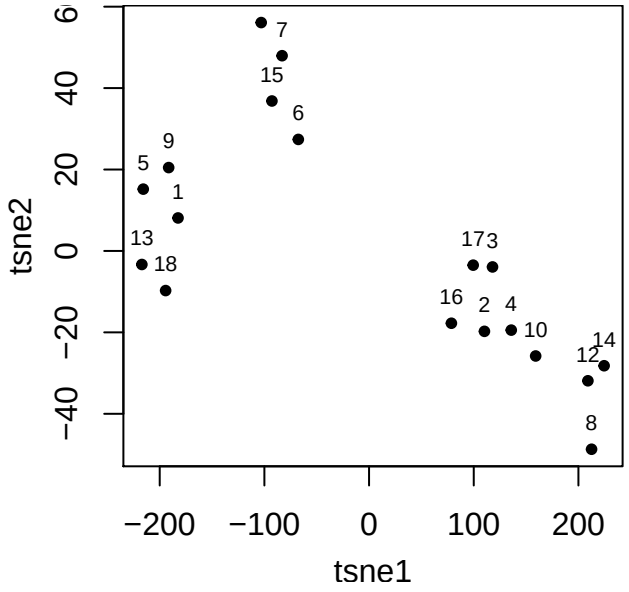

Abbildung 3.4: Darstellung der Flower-Daten mittels t-SNE.

Der Datensatz `flower` lässt sich mit t-SNE ziemlich schön in drei Gruppen unterteilen (Abbildung 3.4). Die Perplexity ist hier tief, da der Datensatz klein ist.

Die Zielfunktion wird bei t-SNE durch eine gradient descent Optimierung minimiert, die zufällig initialisiert wird. Dadurch ist es möglich, dass verschiedene Läufe zu unterschiedlichen Lösungen führen. Es ist vollkommen in Ordnung, t-SNE mehrmals auszuführen (mit den gleichen Daten und Parametern) und die Visualisierung mit dem niedrigsten Wert der Zielfunktion (`itercosts`) als endgültige Visualisierung auszuwählen.

t-SNE lernt ein nichtparametrisches Mapping, d.h. es gibt keine explizite Funktion, die die Projektion durchführt. Daher ist es nicht möglich, Testpunkte in eine bestehende Projektion einzubetten. Man kann aber t-SNE auf dem erweiterten Datensatz erneut ausführen.

4 Cluster-Analyse

In vielen Anwendungen hätte man gerne ein allgemeines Rechenschema, das Objekte aufgrund ihrer Merkmale in Gruppen unterteilt. Je nach Fragestellung möchte man in jedem Fall eine Einteilung in Gruppen ähnlicher Objekte erhalten, oder man möchte "Anhäufungen" (englisch clusters) finden. In beiden Situationen spielt das Konzept "Ähnlichkeit" von zwei Objekten eine zentrale Rolle.

Die Begriffe "Ähnlichkeit" und "Anhäufungen" können in unterschiedlichsten Arten präzisiert werden, was zu vielen Ansätzen und Methoden zur Bildung von Gruppen führt. Diese Methoden werden und wurden nicht nur im Umfeld der Statistik entwickelt, sondern auch in ganz anderen Fachgebieten. Entsprechend finden Sie die Methoden zur Gruppierung von Objekten unter ganz verschiedenen Stichwörtern. Wir nennen das Gebiet in diesem Modul **Cluster-Analyse**. In Marketing (oder Customer Analytics) nennt man Techniken, die Kunden mit ähnlichem Verhalten und Merkmalen in klar unterscheidbare Gruppen einteilen, **Segmentationstechniken** (segmentation techniques). Aber auch Begriffe wie **unsupervised learning**, **unsupervised pattern recognition** und **data mining** (im engeren Sinn) sind heute gebräuchlich. Seinen Ursprung hat das Gebiet in der **numerischen Taxonomie**, ein Gebiet der Biologie.

Synthetisches Beispiel

Ziel in der Cluster-Analyse ist es, die Objekte zu gruppieren. Im Allgemeinen wird dazu für jedes Objekt eine Reihe von Merkmalen erhoben (gemessen). Die Gruppen sollen dann hinsichtlich dieser Merkmale in sich "homogen" und voneinander hingegen deutlich unterscheidbar sein. Das synthetische Beispiel in Abbildung 4.1 zeigt, dass es oft nicht klar ist, was "in sich homogen" bedeutet und welche Gruppen von Objekten (hier Punkte) sich deutlich von den anderen Objekten unterscheidet. Wie viel in sich homogene Gruppen hat es denn in diesem synthetischen Beispiel? Sind es fünf plus ein einzelner Punkt oder sind es nur drei plus ein einzelner Punkt? Und der einzelne Punkt, zählt er als Gruppe oder doch nicht?

Wie dieses synthetische Beispiel weiter zeigt, ist der einfachste Weg Cluster zu bilden, die Daten mit grafischen Mitteln zu visualisieren und damit die Gruppeneinteilung vorzunehmen. Da es sehr schwierig ist, hochdimensionale Daten als solche grafisch darzustellen, müssen wir auf geeignete niedrigdimensionale (am liebsten zweidimensionale) Darstellungen ausweichen, wie wir das z.B. in den letzten beiden Kapiteln gemacht haben. Wie weit die Gruppenstrukturen dabei erhalten bleiben, hängt vom konkreten Datensatz und der verwendeten "Projektionsmethode" ab.

Die Cluster-Analyse ist grundsätzlich eine explorative Datenanalyse-Methode. Folglich können keine statistisch formale Schlussfolgerungen aus der Analyse gezogen werden. Die *Cluster-Analyse dient vielmehr der Erzeugung von neuen Arbeitshypothesen*, welche durch anschliessende Untersuchungen erhärtet werden sollten.

Eine wesentliche Eigenschaft der Cluster-Analyse ist, dass eine aus der Analyse resultierende Gruppeneinteilung nicht als "richtig" oder "falsch" beurteilt werden kann, sondern nur als "brauchbar"

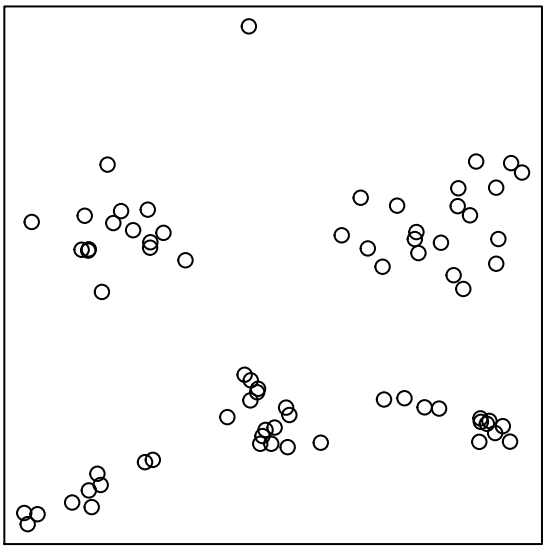

Abbildung 4.1: Synthetisches Beispiel: Deutlich sichtbar sind Anhäufungen (Clusters). Doch wie viele Clusters sind es?

oder "unbrauchbar" *in Hinsicht auf einen bestimmten Zweck*. Für diese Beurteilung sind nicht so sehr formale, sondern vor allem inhaltliche Kriterien von Bedeutung.

Umgangssprachlich lässt sich das zu behandelnde Problem auch so formulieren: Man möchte "Anhäufungen" (engl. clusters) finden, sofern solche vorhanden sind. Eine Hauptschwierigkeit dabei ist, nur mit Hilfe von erhobenen Merkmalen zu definieren, was eine Gruppe, eine Klasse, oder eben eine "Anhäufung" (cluster) sein soll. Daraus ergibt sich ein weiteres Problem: Wie soll man die Anzahl der "Anhäufungen" festlegen (vgl. synthetisches Beispiel in Abbildung 4.1).

Im Wesentlichen können die Cluster-Methoden in drei Typen unterteilt werden: Methoden, die

- den **Raum unterteilen** (**Partitionsverfahren**); d.h. die Daten werden in nicht überlappende Cluster unterteilt.
- **hierarchisch unterteilen**; d.h. die Cluster werden innerhalb von grösseren Clustern unterteilt und bilden so einen Baum von Clustern. Hierarchische Cluster-Verfahren sind spezielle Partitionsverfahren.
- **überlappende Cluster** ergeben.

Im Gegensatz zu den hierarchischen Verfahren kann es bei Partionsverfahren bei jeder Anzahl von Clustern zu einer neuen Unterteilung des Raumes kommen ohne Rücksicht auf die Aufteilungen bei kleinerer oder grösserer Anzahl von Clustern. Überlappende Cluster werden hier nicht behandelt.

4.1 Partitionsverfahren K-Means

Partitionierende Cluster-Verfahren wurden entwickelt, um Objekte bestmöglich in K Cluster zu gruppieren, wobei die Anzahl der Cluster K im Voraus festgelegt werden muss.

Ein beliebtes Rechenverfahren, um die Objekte in Gruppen aufzuteilen, ist das **K-Means-Verfahren**. Beim K-Means-Verfahren wird eine vorgegebene Anzahl K von Gruppenzentren so positioniert, dass die Summe der quadrierten Objektsabstände aller Beobachtungen zum nächstgelegenen Gruppenzentrum minimiert wird:

$$\sum_{i=1}^{n} |x_i - z(i)|^2 = minimal$$

Dabei bezeichnet $z(i)$ das Gruppenzentrum, welches am nächsten zum Objekt i liegt. Diesem Kriterium liegt die Vorstellung zugrunde, dass eine Gruppe ähnlicher Objekte eine kleinere Streuung (konkretisiert durch die empirische Varianz der Gruppenmitglieder) aufweist als eine Gruppe unähnlicher Objekte. Das Gütekriterium kann man auch als totale Variation innerhalb der Cluster (WCV, within-cluster variation) über alle K Cluster beschreiben, welche minimiert werden soll.

$$\underset{C_1,\ldots,C_K}{min} \left\{ \sum_{k=1}^{K} WCV(C_k) \right\}$$

Wobei die Variation innerhalb eines Cluster k (C_k), wie folgt bestimmt wird:

$$WCV(C_k) = \frac{1}{\#C_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

Wobei $\#C_k$ der Anzahl Beobachtungen im Cluster k entspricht und p die Anzahl Features(Dimension) der Objekte ist. $(x_{ij} - x_{i'j})^2$ entspricht der quadrierten euklidischen Distanz zwischen den Objekten i und $i\prime$. Entsprechend setzt dieses Verfahren voraus, dass alle Merkmale mit quantitativen Grössen beschrieben sind. Eine Distanzmatrix ist nicht ausreichend. Der Algorithmus umfasst folgende Schritte:

i) Generiere eine Starteinteilung, in der die Objekte zufällig einer der K Cluster zugeordnet werden.

ii) Berechne die Clustermittelwerte.

iii) Ordne jedes Objekt jenem Cluster zu, zu dessen Mittelwert es am nächsten liegt. Man erhält also eine neue Clustereinteilung.

Die Schritte 2 und 3 werden wiederholt bis es keine Änderungen mehr gibt.

Weil das zu optimierende Kriterium nicht konvex ist, hängt die Lösung von den Startwerten ab und ist demzufolge nicht eindeutig. Es empfiehlt sich also, das Verfahren mit mehreren Startpartitionen (oder äquivalent mit verschiedenen Startkonfigurationen der Clusterzentren) durchzuführen und dann die Partition mit dem kleinsten Wert des Gütekriteriums (total WCV) als die beste Lösung herauszuziehen.

Der Algorithmus wird direkt auf den quantitativen Merkmalen angewendet. Das Verfahren eignet sich deshalb auch für riesige Datenmengen (d.h. grosse Anzahl von Objekten), weil die Daten nicht notwendigerweise im Hauptspeicher vorhanden sein müssen.

Häufig wird man die Vorstellung haben, dass sich die Objektmenge in eine Anzahl von "natürlichen" Gruppen aufteilen lässt. Die genaue Anzahl der Gruppen ist jedoch nicht bekannt. Ein plausibler Vorschlag, die Gruppengrösse K zu bestimmen, beruht auf der Eigenschaft des K-Means-Verfahrens, dass der minimale Wert des Gütekriteriums mit wachsender Gruppenzahl K nicht grösser wird. Bezeichnet h_K den minimalen Wert des Gütekriteriums bei K Gruppen, dann wählt man als "natürliche" Gruppenanzahl jenen Wert von K, für den $h_{K-1} - h_K$ deutlich grösser ist als $h_K - h_{K+1}$. Hinter diesem Vorschlag steht folgende Annahme: Ein relativ grosser Wert $h_{K-1} - h_K$ deutet darauf hin, dass in der Partition mit $K-1$ Gruppen noch eine heterogene Gruppe enthalten ist, die sich in zwei homogene (natürliche) Gruppen aufspalten lässt. Ein kleiner Wert von $h_K - h_{K+1}$ deutet hingegen darauf, dass bei der Erhöhung der Gruppenanzahl von K auf $K+1$ eine homogene Gruppe aufgespalten wurde (Abbildung 4.2).

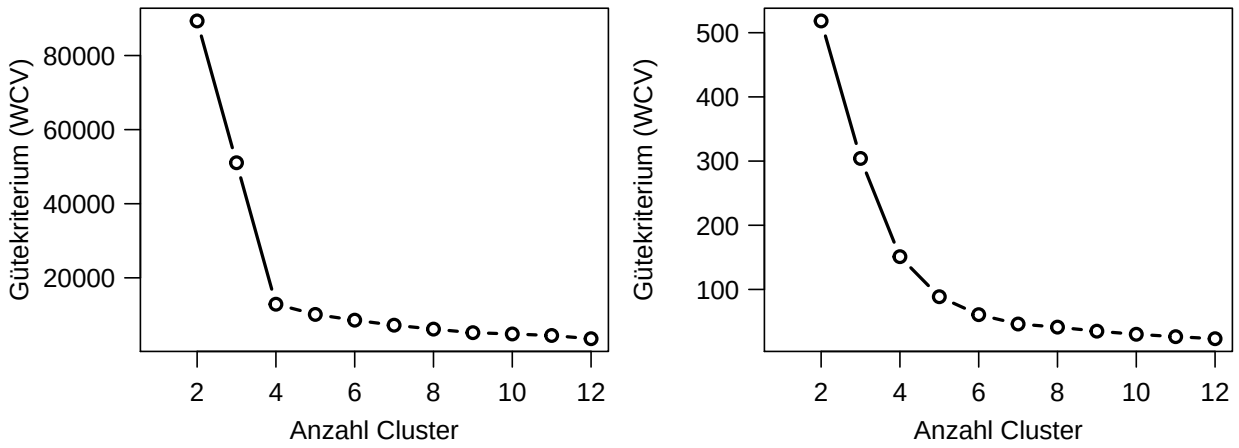

Abbildung 4.2: Gütekriterium gegen Anzahl Cluster. Im linken Beispiel scheint das Festlegen von drei Gruppen ein vernünftiger Vorschlag zu sein. Mit der Erhöhung von zwei auf drei Gruppen kann noch eine homogene Gruppe separiert werden. Bei der Erhöhung auf vier Gruppen deuten die Daten eher darauf hin, dass eine homogene Gruppe aufgespaltet wird. Im rechten Beispiel für die synthetischen Daten aus Abbildung 4.1 ist hingegen kein Knick sichtbar ist, entsprechend kann man auch keine optimale Anzahl Cluster finden.

Das K-Means-Verfahren hat die Tendenz, "kugelförmige""Gruppen mit gleich vielen Objekten zu bilden. Deshalb können natürliche Gruppen mit länglicher Ausdehnung oder verschiedener Streuungen schlecht erkannt werden (vgl. Abbildung 4.3 links). Die durch das K-Means-Verfahren vorgeschlagenen Gruppierungen sind dann in der Regel unbrauchbar. Sind Ausreisser vorhanden, können auch komische Effekte auftreten, falls nicht für jeden Ausreisser (oder für die Anhäufung von Ausreissern) ein separater Cluster gebildet wird (vgl. Abbildung 4.3 rechts).

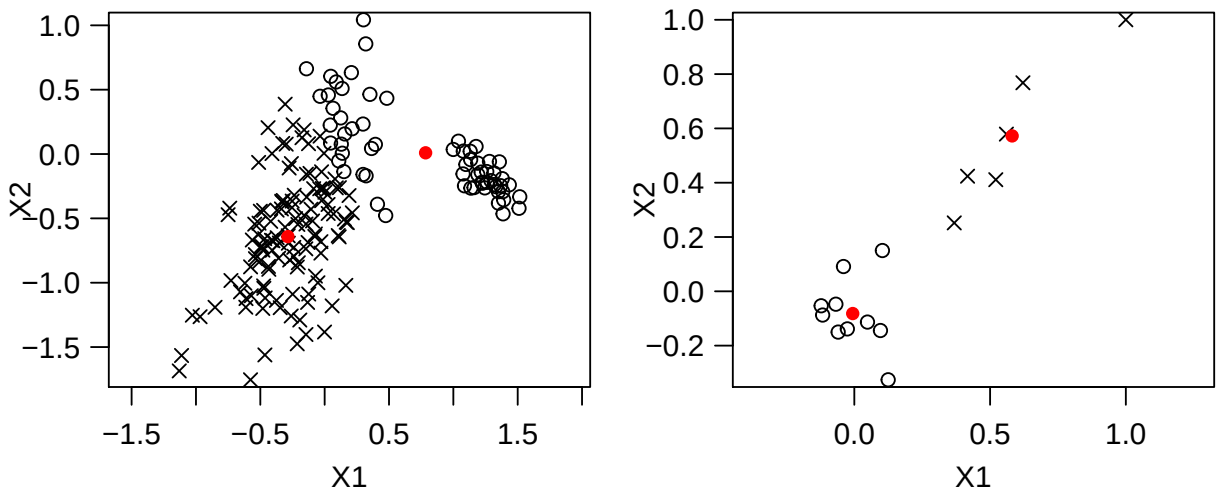

Abbildung 4.3: Resultate des K-Means-Verfahrens bei ungleicher Streuung (links) und bei Ausreissern (rechts). Die beiden ausgefüllten Kreisscheiben zeigen das jeweilige Clusterzentrum an. Die Beobachtungen mit gleichem Symbol gehören gemäss dem K-Means-Verfahren dem gleichen Cluster an.

R-Befehle

In R gibt es für das K-Means Clustering die Funktion `kmeans`. Als Argumente sind dazu die Datenmatrix (x) und die Anzahl Cluster (*centers*) erforderlich. Im folgenden ist die Analyse am Beispiel eines zweidimensionalen Datensatz wiedergegeben.

```
set.seed(32)
dat <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 1.6, sd = 0.3), ncol = 2))
colnames(dat) <- c("x", "y")
```

```
cl <- kmeans(x=dat, centers=3)
```

Wichtige Outputgrössen sind die zugewiesenen Cluster (`...$cluster`), die Zentren der Cluster (`...$centers`), sowie die Within-Cluster Variation (WCV) der einzelnen Cluster (`...$withinss`), sowie die aller Cluster zusammen (`...$tot.withinss`).

```
cl$cluster
```

```
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2
##  [71] 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2
## [106] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 2 1 1 1 1 1
```

```
cl$centers
```

```
##            x           y
## 1 1.69231733  1.63991493
## 2 0.98455282  1.04765117
## 3 0.02081891 -0.04047304
```

```
cl$withinss
```

```
## [1] 6.805419 9.008258 6.998124
```

```
cl$tot.withinss
```

```
## [1] 22.8118
```

Das Ergebnis einer Clusteranalyse wird häufig in einem 2D-Plot visualisiert. Bei einem 2-dimensionalen Datensatz kann man die Daten direkt plotten. Ansonsten kann man auch auf Visualisierungstechniken für hochdimensionale Daten zurückgreifen (Hauptkomponentenanalyse, MDS der t-SNE).

```
plot(dat, col = cl$cluster, las=1)
points(cl$centers, col = 1:3, pch = 8, cex = 2)
```

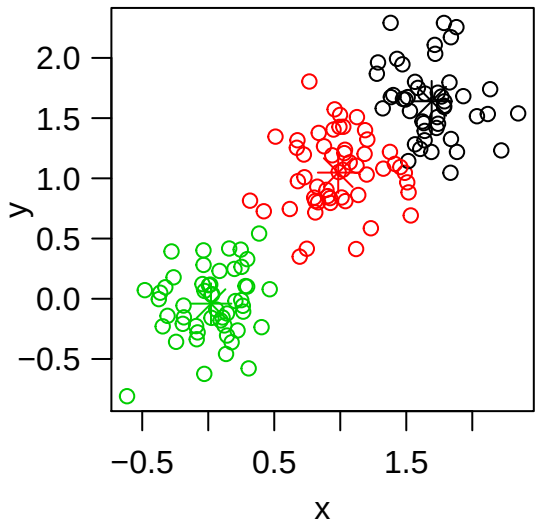

Abbildung 4.4: Visualisierung K-Means

Eine Abbildung der Within-Cluster Variation gegen die Anzahl Cluster zur Bestimmung der optimalen Anzahl Cluster kann manuell oder sehr komfortabel mit der Funktion `fviz_nbclust` (`method='wss'`) aus dem Paket `factoextra` generiert werden.

```
library(factoextra)
fviz_nbclust(dat, kmeans, method='wss')
```

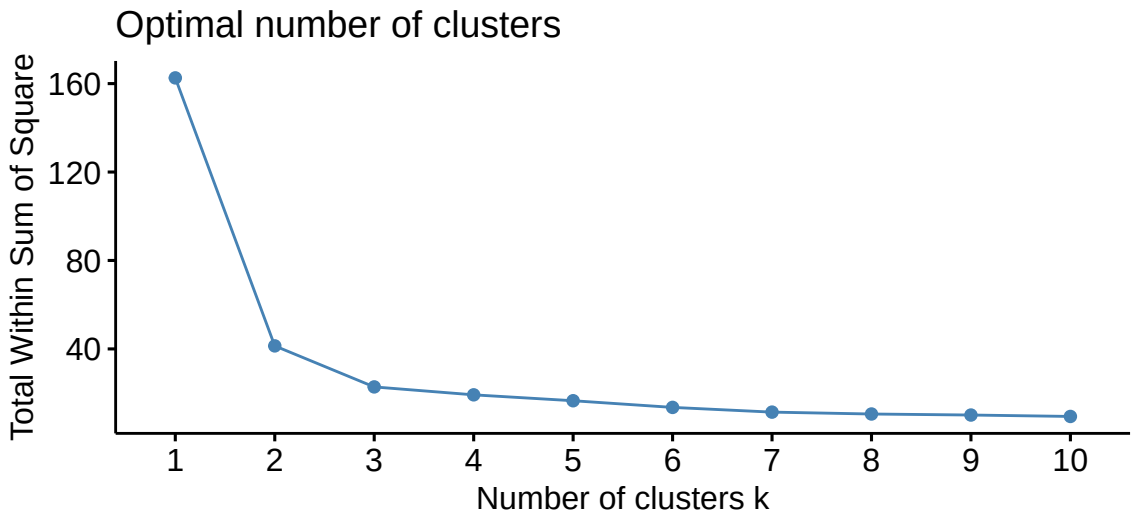

Abbildung 4.5: Abbildung der Within-Cluster Variation gegen die Anzahl Cluster zur Bestimmung der optimalen Anzahl Cluster.

4.2 Silhouettenanalyse

Zur Analyse des Clusterergebnisses werden häufig die Silhoutten verwendet. Die Silhouette gibt an, wie gut die Zuordnung für eine Beobachtung hinsichtlich der beiden nächstgelegenen Clustern ist. Der Silhouetten-Koeffizient einer Beobachtung i ist definiert als:

$$sil_i = \frac{b_i - a_i}{max(a_i, b_i)} \in [-1, 1]$$

a_i ist dabei die mittlere Distanz zwischen dem Punkt i und allen anderen Punkten aus dem gleichen Cluster. b_i ist die mittlere Distanz zwischen i und allen Punkten aus dem benachbarten Cluster. Ist $b_i \gg a_i$ liegt der benachbarte Cluster weit weg und der Silhouetten-Koeffizient ist nahe bei 1. Ist $b_i \approx a_i$ ist der benachbarte Cluster ähnlich nahe (Silhouetten-Koeffizient ≈ 0). Einzelne Koeffizienten können auch negativ sein. Das bedeutet, dass die Beobachtungen im Mittel näher bei den Beobachtungen im benachbarten Cluster, als zu denen im Eigenen liegen. Das Clustering muss dadurch nicht zwingend schlecht sein.

Zur Beurteilung des Clusterergebnisses wird die mittlere Sihouettenbreite über alle Beobachtung betrachtet. Die Ergebnisse können dabei gemäss folgender Faustregel interpretiert werden:

$0.70 < \text{sil} < 1.00$:	Gute Struktur
$0.50 < \text{sil} < 0.70$:	Vernünftige Struktur
$0.25 < \text{sil} < 0.50$:	Schwache Strukturen, weitere Analysen notwenig
$-1 < \text{sil} < 0.25$	Forget it!

In R gibt es für die Silhouettenanalyse die Funktion `silhouette` aus dem Paket `cluster`. Als Input wird die Clustereinteilung und die Distanzmatrix benötigt. Das Ergebnis kann direkt geplottet werden.

```
library(cluster)
plot(silhouette(x=cl$cluster, dist=dist(dat)))
```

Abbildung 4.6: Silhouetten-Plot mit den mittleren Koeffizienten für die einzelnen Cluster und der ganzen Analyse.

Die mittlere Silhouttenbreite kann auch zur Wahl der optimalen Anzahl Cluster verwendet werden. Dazu sucht man die Clusteranzahl mit dem höchsten Koeffizienten. In R kann man dazu in der Funktion `fviz_nbclust` die Methode `method='silhouette'` wählen.

```
library(factoextra)
fviz_nbclust(dat, kmeans, method='silhouette')
```

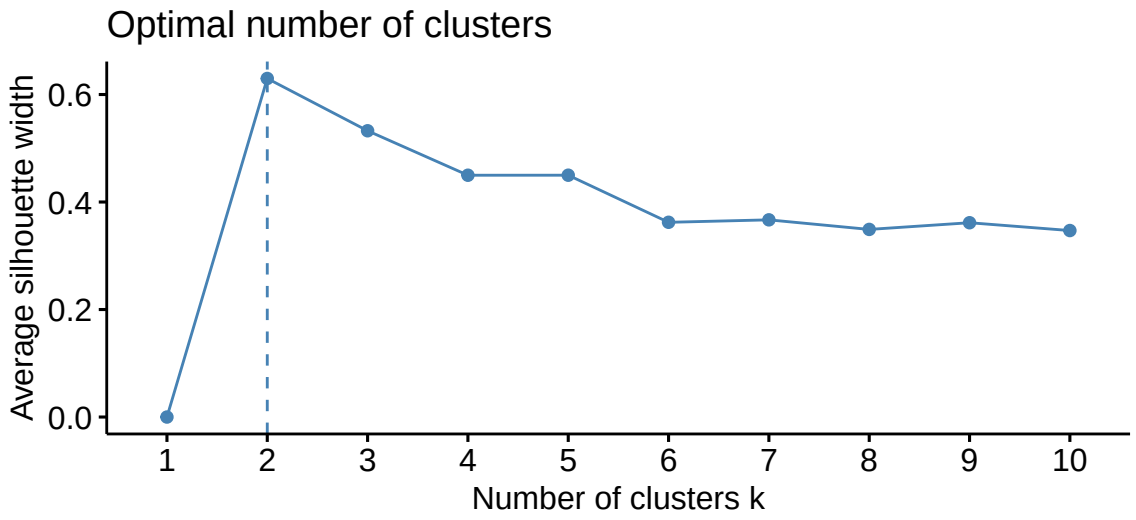

Abbildung 4.7: Abbildung der mittleren Silhouettenbreite gegen die Anzahl Cluster zur Bestimmung der optimalen Anzahl Cluster.

Diese Analyse kann auch zu einem anderen Ergebnis als der Ansatz mit der Within-cluster Variation (WCV) führen.

4.3 Partitioning Around Medoids (PAM)

Das **Partitioning Around Medoids** (**PAM**) Verfahren, das von Kaufmann und Rousseeuw (2005) entwickelt wurde, ist ebenfalls ein Partitionierungsverfahren und sehr ähnlich zum K-Means Verfahren. Beide Methoden gehen von einer festen Anzahl K von Clustern aus und minimieren den Abstand zum jeweiligen Clusterzentrum. Im Gegensatz zu K-Means, wo die Clusterzentren (Centroids genannt) künstlich kreierte "Objekte" sind, werden im PAM-Verfahren reale Objekte (Medoids genannt), d.h. spezifische Beobachtungen, als Clusterzentren gewählt. Da man hier nur die Distanzen zwischen den Objekten betrachtet, ist es, im Gegensatz zu K-Means, ausreichend, wenn nur die Distanzen bekannt sind. Die Methode kann dabei mit allen Distanzmassen umgehen und ist daher nicht auf quantitative Daten beschränkt.

Ausgehend von einer Distanzmatrix d_{ij} wird die Gesamtdistanz der Objekte zu den Repräsentanten der jeweiligen Cluster minimiert:

$$F(x) = \min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \cdot z_{ij}$$

Die Variable z_{ij} stellt sicher, dass nur die Distanzen zwischen Objekten von demselben Cluster berechnet werden.

Der Algorithmus zur Minimierung dieser Kostenfunktion sieht wie folgt aus:

i) k Objekte werden zufällig als Clusterzentrum gewählt.

ii) Jedes Objekt wird dem nächsten Clusterzentrum zugewiesen.

iii) Die Kostenfunktion wird evaluiert.

iv) Iterativ wird ein Medoids mit einem nicht Medoids-Objekt vertauscht und die Schritte ii) und iii) wiederholt. Wenn sich dadurch die Kostenfunktion verringert, wird der neue Punkt als Medoid fixiert. Es wird getauscht bis keine Verbesserung mehr auftritt.

Im Gegensatz zum K-Mean Verfahren, wo die quadrierten Distanzen eingehen (d.h. es entspricht einer Varianzbetrachtung), werden hier nur die Distanzen betrachtet und das zu minimierende Kriterium entspricht folglich dem Gesamtweg von den Zentren zu den Objekten. Deshalb haben Ausreisser im PAM-Verfahren auch einen viel geringeren Einfluss.

In R gibt es dafür die Funktion `pam()` aus dem Paket `cluster`. Als Alternative für grosse Datensätze ist die Funktion `clara` aus dem gleichen Paket geeignet. Dies wird erreicht in dem der pam-Algorithmus auf Teildatensätze angewendet wird. Clusterzentren sind dabei immer Beobachtung auf welche mit `$medoids` zugegriffen werden kann.

```
library(cluster)
cl_pam <- pam(dat, k=3)
plot(dat, col = cl_pam$clustering, main="pam")
points(cl_pam$medoids, col = 1:3, pch = 8, cex = 2)
cl_clara <- clara(dat, k=3)
plot(dat, col = cl_clara$clustering, main="clara")
points(cl_clara$medoids, col = 1:3, pch = 8, cex = 2)
```

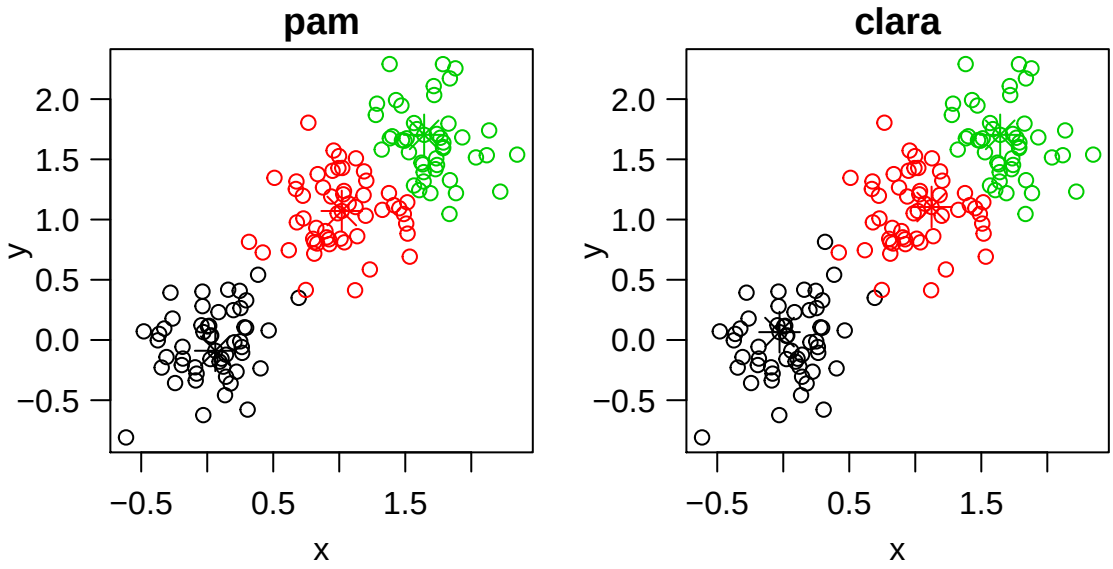

Abbildung 4.8: Clustering mit PAM (links mit der Funktion pam, rechts mit der Funktion clara

Die Bestimmung der Anzahl Cluster über das Gütekriterium ist nicht möglich, da dieses nicht monoton abnimmt. Das Festlegen der Anzahl Cluster über die mittlere Silhouttenbreite ist aber weiterhin möglich.

```
library(factoextra)
fviz_nbclust(dat, kmeans, method="silhouette")
```

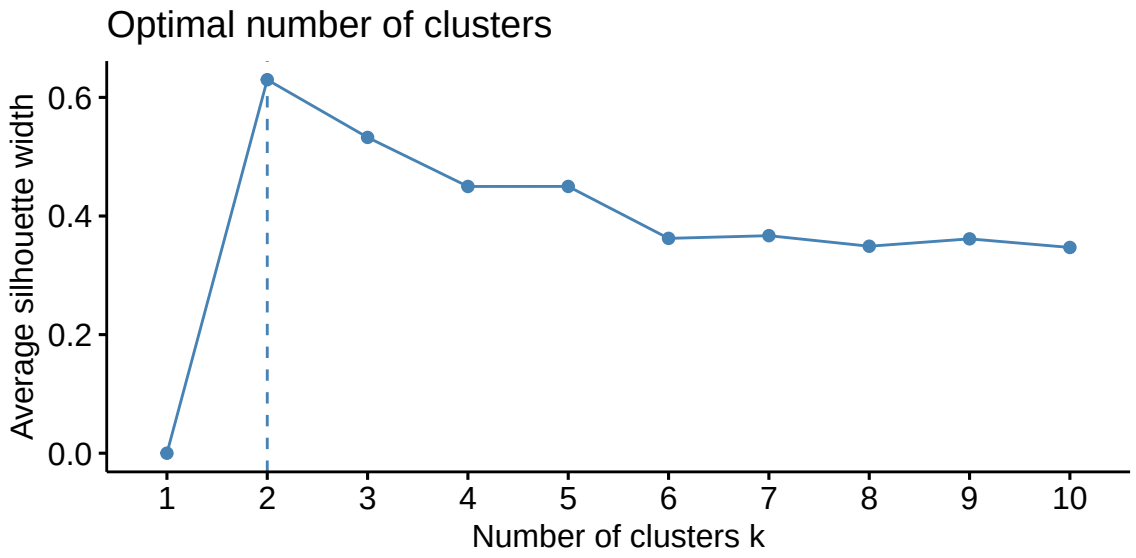

Abbildung 4.9: Mittlere Shilhouttenbreite gegen die Anzahl Cluster zur Bestimmung der optimalen Anzahl Cluster.

4.4 Hierarchische Clusteranalyse

Bei den **hierarchischen Cluster-Analyse-Verfahren** muss, im Gegensatz zu den vorgestellten Partitionsverfahren, die Anzahl der Cluster nicht von Beginn an festgelegt werden. Bei diesen Verfahren, wird zuerst eine Folge von Unterteilungen (Partitionen) der beobachteten Objekte konstruiert und dann unter Berücksichtigung der Unterteilungen ein geeigneter Grenzwert für die Anzahl Klassen festgelegt. Dabei wird die Anforderung an die Homogenität der Gruppen entweder schrittweise erhöht (**divisive** Verfahren) oder schrittweise verringert (**agglomerative** Verfahren). Bei divisiven Verfahren wird eine höhere Homogenität dadurch erreicht, dass Clusters sukzessive in Teilclusters zerlegt werden. Die Klassenzahl nimmt folglich im Laufe des Verfahrens zu. Bei agglomerativen Verfahren wird die Homogenität so verringert, dass Clusters sukzessive vereinigt werden. Die Klassenzahl wird deshalb im Laufe des Verfahrens kleiner.

Im Folgenden werden wir die wichtigsten Schritte in einem hierarchischen Cluster-Analyse-Verfahren beschreiben und einige der beliebtesten Methoden vorstellen. Vom Rechenaufwand kommen nur agglomerative Verfahren in Frage kommen, weil bei grösseren Objektanzahlen der Rechenaufwand

bei den divisiven Methoden zu gross ist.[4]

Die hierarchische Clusteranalyse basiert auf Distanzmassen. Wir gehen davon aus, dass alle Distanzen zwischen je zwei Objekten in einer Distanzmatrix $\mathcal{D}$ abgelegt wurden. Das Aufstellen der Hierarchie besteht dann bei der Anwendung eines **agglomerativen Verfahrens** aus vier Schritten.

i) Jedes einzelne Objekt wird als Gruppe betrachtet. Somit gibt es zu Beginn n Gruppen.

ii) Suche das kleinste Element in der Distanzmatrix. Angenommen es sei d_{km}. Dann werden die Gruppen k und m zu einer Gruppe zusammengefasst. Der Wert d_{km} wird notiert. Falls es kein eindeutig kleinstes Element in der Distanzmatrix gibt, muss ein verfeinertes Verfahren angewandt werden. In den meisten Fällen genügt es aber, wenn willkürlich eines ausgewählt wird.

iii) Berechne die Distanzen von der neuen Gruppe zu allen anderen, lösche die k-te und m-te Zeilen und Spalten in der Distanzmatrix $\mathcal{D}$ und füge eine Spalte/Zeile mit den neu berechneten Distanzwerten hinzu. Die Distanzmatrix hat nun eine Zeile und eine Spalte weniger als zuvor.

vi) Wiederhole die Schritte 2 und 3 bis nur noch eine Gruppe übrig bleibt. Folglich besteht die Distanzmatrix nur noch aus einem Wert.

Für den dritten Schritt, der Bestimmung der Distanzen von der neuen Gruppe zu allen anderen, gibt es verschiedene Verfahren.

Single-Linkage-Verfahren

Beim **Single-Linkage-Verfahren** ist die Distanz zwischen zwei Gruppen U und V gleich der kleinsten Distanz zwischen einem Objekt aus U und einem aus V:

$$d(U,V) := min\{d(u,v) : \ u\ in\ U,\ v\ in\ V\}$$

Beispiel: Nehmen wir an, dass es fünf Objekte A, B, C, D und E gibt. Die Distanzen zwischen diesen Objekten sind in der Matrix $\mathcal{D}$ festgehalten:

$$\mathcal{D} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & & & & \\ 9 & 0 & & & \\ 3 & 7 & 0 & & \\ 6 & 5 & 9 & 0 & \\ 11 & 10 & 2 & 8 & 0 \end{pmatrix} \end{array}$$

[4] agglomerativ: falls k Cluster schon bekannt sind, müssen $\binom{k}{2} = \frac{1}{2}(k)(k-1)$ Fälle untersucht werden, um herauszufinden, welche zwei Cluster vereint werden sollen; divisiv: Müssen wir einen Cluster von n Objekte in 2 Cluster aufteilen, heisst das, $2^{n-1} - 1$ Kombinationen von Aufteilungen zu untersuchen, um die Beste zu finden!

Im ersten Schritt betrachten wir jedes Objekt als einen Cluster; d.h. wir haben fünf Cluster. Die kleinste Distanz zwischen den Clustern ist jene zwischen C und E: $d_{CE} = 2$. Somit verschmelzen wir den Cluster $\{C\}$ mit dem Cluster $\{E\}$ und erhalten insgesamt die vier Cluster $\{A\}$, $\{B\}$, $\{D\}$ und $\{C, E\}$. Gemäss der Nächste-Nachbarmethode ist die Distanz zwischen $\{C, E\}$ und den anderen Cluster gleich

$$
\begin{aligned}
d(\{C,E\},A) &= min\{d(C,A),\ d(E,A)\} = min\{3,11\} = 3\\
d(\{C,E\},B) &= min\{d(C,B),\ d(E,B)\} = min\{7,10\} = 7\\
d(\{C,E\},D) &= min\{d(C,D),\ d(E,D)\} = min\{9,8\} = 8
\end{aligned}
$$

Damit erhalten wir folgende Distanzmatrix:

$$
\mathcal{D} = \begin{array}{c} \\ A \\ B \\ D \\ \{C,E\} \end{array}
\begin{array}{c}
\begin{array}{cccc} A & B & D & \{C,E\} \end{array} \\
\begin{pmatrix} 0 & & & \\ 9 & 0 & & \\ 6 & 5 & 0 & \\ 3 & 7 & 8 & 0 \end{pmatrix}
\end{array}
$$

Als nächstes werden also Cluster $\{C, E\}$ und $\{A\}$ bei einer Distanz von 3 zusammengelegt, danach Cluster $\{B\}$ und $\{D\}$ bei einer Distanz von 5 und zu guter Letzt die Cluster $\{A, C, E\}$ und $\{B, D\}$ bei einer Distanz von 6.

Um zwei Gruppen fusionieren zu können, genügt es, dass ein Objekt aus der einen Gruppe und ein Objekt aus der anderen Gruppe nahe beieinander liegen (im Sinne des Distanzmasses). An die Distanzen zwischen den übrigen Objekten werden keine Forderungen gestellt, so dass diese Objekte durchaus sehr weit auseinander liegen können. Aufgrund dieser Eigenschaft ist das Single-Linkage-Verfahren am ehesten von den hier vorgestellten Verfahren in der Lage, Gruppen von beliebiger Form zu bilden. Gruppen, die durch "Brücken" verbunden sind, werden, selbst wenn sie ansonsten deutlich getrennt sind, jedoch gerne in eine Gruppe fusioniert. Man spricht in diesem Zusammenhang auch von der **Verkettungseigenschaft**. Diese Eigenschaft kann durchaus seine Berechtigung haben, vor allem in Anwendungsgebieten wie Taxonomie, wo die evolutionäre Entwicklung eine Rolle spielt. (Das Single-Linkage-Verfahren hat übrigens seinen Ursprung in der Taxonomie.)

Das Single-Linkage-Verfahren eignet sich auch, um multivariate Ausreisser aufzudecken, weil es diese oft in Gruppen der Grösse 1 zuordnet und diese Gruppen grosse Distanzen zu den anderen Objekten aufweisen. Der Aufwand zur Berechnung ist sehr vorteilhaft: man braucht sowohl wenig Zeit wie auch wenig Speicherplatz.

Complete-Linkage-Verfahren

Beim **Complete-Linkage-Verfahren** ist die Distanz zwischen zwei Gruppen U und V gleich der grössten Distanz zwischen einem Objekt aus U und einem aus V:

$$d(U,V) := max\{d(u,v) : \ u\,in\,U,\ v\,in\,V\}.$$

Im **Beispiel** aus dem Single-Linkage-Verfahren wird der Schritt drei nun ersetzt durch die Berechnung der Distanzen zwischen den Clustern gemäss des Complete-Linkage-Verfahrens:

$$\begin{aligned}
d(\{C,E\},A) &= max\{d(\{C,A\},\ d\{E,A\}) = max\{3,11\} = 11\\
d(\{C,E\},B) &= max\{d(\{C,B\},\ d\{E,B\}) = max\{7,10\} = 10\\
d(\{C,E\},D) &= max\{d(\{C,D\},\ d\{E,D\}) = max\{9,8\} = 9
\end{aligned}$$

Damit erhalten wir folgende Distanzmatrix:

$$\mathcal{D} = \begin{array}{c} \\ A \\ B \\ D \\ \{C,E\} \end{array} \begin{array}{c} \begin{array}{cccc} A & B & D & \{C,E\} \end{array} \\ \begin{pmatrix} 0 & & & \\ 9 & 0 & & \\ 6 & 5 & 0 & \\ 11 & 10 & 9 & 0 \end{pmatrix} \end{array}$$

Als nächstes werden nun Cluster $\{B\}$ und $\{D\}$ bei einer minimalen Distanz von 5 zusammengelegt, danach Cluster $\{A\}$ und $\{B,\ D\}$ bei einer Distanz von 9 und zu guter Letzt die Cluster $\{A,\ B,\ D\}$ und $\{C,\ E\}$ bei einer Distanz von 11.

Werden zwei Gruppen, die einen Distanzwert von d_{km} haben, zusammengeführt, so ist die maximale Distanz zwischen zwei Objekten in der neuen Gruppe auch gerade d_{km}. In diesem Sinn erzeugt das Complete-Linkage-Verfahren homogene Gruppen. Im Gegensatz zu den möglicherweise "länglichen" Gruppierungen, die das Single-Linkage-Verfahren generiert, liefert das Complete-Linkage-Verfahren eher "kugelförmige" Gruppierungen.

Average-Linkage-Verfahren

Beim **Average-Linkage-Verfahren** ist die Distanz zwischen zwei Gruppen U und V gleich der mittleren Distanz zwischen allen Objekt aus U und V:

$$d(U,V) := \frac{1}{\#(U)\,\#(V)} \sum_{u\in U} \sum_{v\in V} d(u,v)\,.$$

Wir betrachten noch einmal unser **Beispiel**. Der Schritt drei wird nun ersetzt durch die Berechnung der Distanzen zwischen den Clustern gemäss des Average-Linkage-Verfahrens:

$$d(\{C,\, E\}, A) = \frac{1}{2}(d(C,A) + d(E,A)) = \frac{1}{2}(3+11) = 7$$
$$d(\{C,\, E\}, B) = \frac{1}{2}(d(C,B) + d(E,B)) = \frac{1}{2}(7+10) = 8.5$$
$$d(\{C,\, E\}, D) = \frac{1}{2}(d(C,D) + d(E,D)) = \frac{1}{2}(9+8) = 8.5$$

Damit erhalten wir folgende Distanzmatrix:

$$\mathcal{D} = \begin{array}{c} \\ A \\ B \\ D \\ \{C,E\} \end{array} \begin{array}{c} \begin{array}{cccc} A & B & D & \{C,E\} \end{array} \\ \begin{pmatrix} 0 & & & \\ 9 & 0 & & \\ 6 & 5 & 0 & \\ 7 & 8.5 & 8.5 & 0 \end{pmatrix} \end{array}$$

Als nächstes werden nun Cluster $\{B\}$ und $\{D\}$ bei einer Distanz von 5 zusammengelegt, danach Cluster $\{A\}$ und $\{C,\, E\}$ bei einer Distanz von 7 und zu guter Letzt die Cluster $\{B,\, D\}$ und $\{A,\, C,\, E\}$ bei einer Distanz von 8.167.

Damit zwei Gruppen fusioniert werden, genügt es beim Average-Linkage-Verfahren, dass deren Objekte im Mittel hinreichend ähnlich sind. Objekte, die weit voneinander entfernt sind, können durch Objekte, die nahe beieinander liegen, kompensiert werden. So bilden sich ähnlich wie beim Complete-Linkage-Verfahren eher "kugelförmige" Gruppierungen.

Ward-Verfahren

Das **Ward-Verfahren** beruht auf der Minimierung des "Verlusts an Informationen" bei der Verschmelzung von zwei Gruppen beruhen. Die Methode wird in der Regel so implementiert, dass der Verlust an Informationen durch die Zunahme der Summe der quadrierten Abweichungen von den Gruppenzentren gemessen wird. Sei beim Cluster k ESS_k die Summe der quadrierten Abweichungen jedes Objektes im Cluster k vom Cluster-Mittelwert (Schwerpunkt). Falls es K Cluster gibt, ist ESS die Summe aller ESS_k, d.h. $\text{ESS} = \text{ESS}_1 + \text{ESS}_2 + \ldots + \text{ESS}_K$. Bei jedem Schritt im Cluster-Verfahren wird die Vereinigung aller möglichen Paare von Cluster betrachtet. Jene beiden Cluster werden verschmolzen, deren Vereinigung den geringsten Anstieg in ESS ergibt. Anfangs besteht jeder Cluster aus einem einzigen Objekt. Wenn es N Objekte hat, so gilt $\text{ESS}_k = 0$, $k = 1, \ldots, N$ und damit auch $\text{ESS} = 0$. Am Ende des Verfahrens sind alle Objekte in einem einzigen Cluster und der Wert für ESS ist gegeben durch

$$\text{ESS} = \sum_{k=1}^{N} (x_k - \bar{x})^T (x_k - \bar{x}) \,,$$

wobei der Vektor x_k die Merkmale des Objekts k beschreibt und der Vektor $\bar{x}$ die Mittelwerte der einzelnen Merkmale enthält.

Das Verfahren von Ward liefert ebenfalls eher kugelförmige Gruppierungen und ist sicher geeignet, wenn den Clustern sphärische Normalverteilungen zugrunde liegen.

R-Befehle

In R kann für das hierarchische Clustering die Funktion `hclust()` verwendet werden. Als Input muss eine Distanzmatrix bereitgestellt werden. Mit dem Argument `method`, kann man dort, unter anderem, die Linkage-Methoden `'ward.D2'`, `'single'`, `'complete'`, `'average'` auswählen.

```
h_cluster <- hclust(dist(dat), method = 'single')
```

4.5 Dendrogramm

Die hierarchische Clusterstruktur kann durch ein so genanntes **Dendrogramm** (dendron: griechisch für Baum) visualisiert werden. Das ist ein auf den Kopf gestellter Baum, der die hierarchische Zerlegung der Objektmenge in immer kleinere Teilmengen darstellt (vgl. Abbildung 4.10). Die Wurzel (zuoberst im Dendrogramm) repräsentiert den Cluster, der alle Objekte enthält. Die Blätter des Baumes (ganz unten im Dendrogramm) repräsentieren Cluster, in denen sich nur noch je ein einzelnes Objekt befindet. Die Verzweigungen dazwischen repräsentieren die entsprechende Aufteilung des Clusters in je zwei Teil-Clusters. Auf der vertikalen Achse sieht man, bei welcher Distanz der Cluster aufgeteilt wird.

Das geschilderte Vorgehen bei der hierarchischen Clusteranalyse führt nur zu einer sukzessiven Unterteilung der Objekte. Jedoch ist noch nicht klar, welche Cluster als in sich homogen gelten können und welche nicht.

Um einen Schwellwert festzulegen, der die homogenen von den inhomogenen Clustern trennt, geht man im Dendrogramm von oben nach unten durch und betrachtet die Differenzen der Cluster-Distanzen zwischen den Aufsplitterungen. Üblicherweise sind die Differenzen zu unterst am kleinsten und oben oft gross. Meistens gibt es dazwischen eine Zone, wo sich die Differenzen plötzlich stark verkleinern. Oberhalb davon ist ein geeigneter Ort, um den Schwellwert zu setzten.

Manchmal ist es jedoch sinnvoller, keinen globalen Schwellwert für die Clusteridentifikation festzulegen, sondern einen separaten für jeden “Hauptast”. Dies kann helfen, Clusterbildungen bei Anhäufungen mit sehr unterschiedlichen internen Streuungen zu identifizieren. Der Nachteil ist, wenn man so stark auf die Daten eingeht, dass die Entscheidungen kaum objektiv nachvollziehbar sind und eher willkürlich erscheinen können.

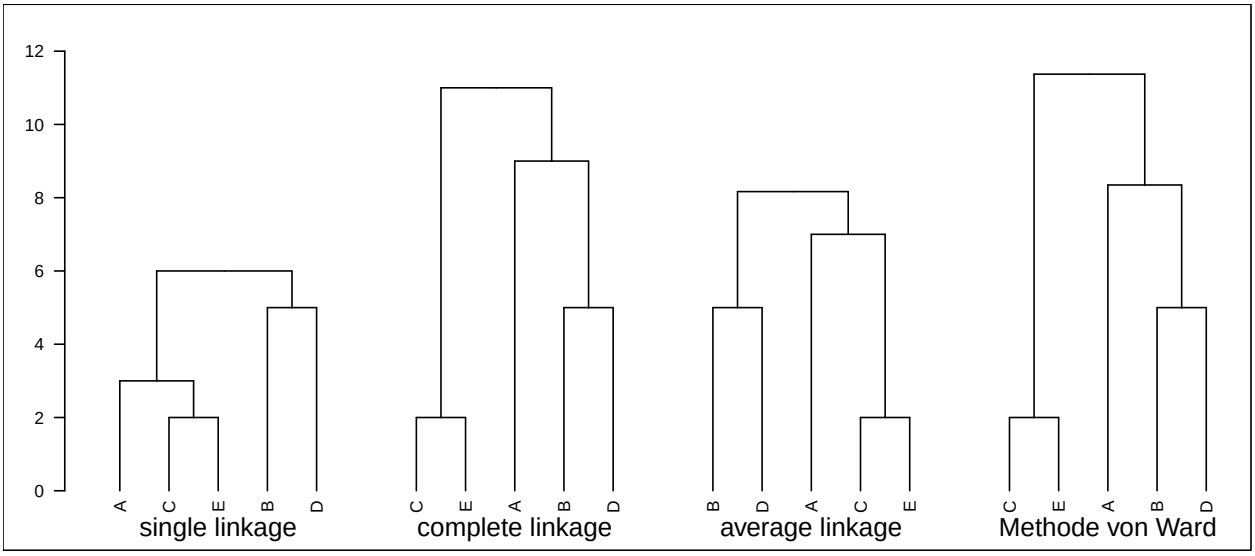

Abbildung 4.10: Dendrogramme der Resultate aus dem Single-Linkage-, Complete-Linkage- und dem Average-Linkage-Verfahren sowie der Methode von Ward.

R-Befehle

Um eine Dendrogramm zu erzeugen, muss einfach das Objekt aus `hclust` geplottet werden (`plot(res)`). Hilfreich kann hier das Argument `hang` sein, es gibt an, wieviel die Labels untern den Ästen hängen. Ein negativer Wert führt dazu, dass die Labels gleichmässig unterhalb von 0 angeordnet sind. Um eine Aufteilung in Cluster vorzunehmen, gibt es die Funktion `cutree`. Als Input wird wiederum das Resultat aus `hclust` benötigt. Unterteilen kann man das Dendrogramm in eine fixe Anzahl Cluster (Argument `k`) oder man kann es in einer spezifischen Höhe aufteilen (Argument `h`). Graphisch können diese Gruppen mit der Funktion `rect.hclust` in das Dendrogramm eingezeichnet werden.

```
h_cluster <- hclust(dist(dat), method = 'ward.D2')
plot(h_cluster, las=1, hang=-1)
klassen <- cutree(h_cluster, k=3)
rect.hclust(h_cluster, k=3, border="red")
```

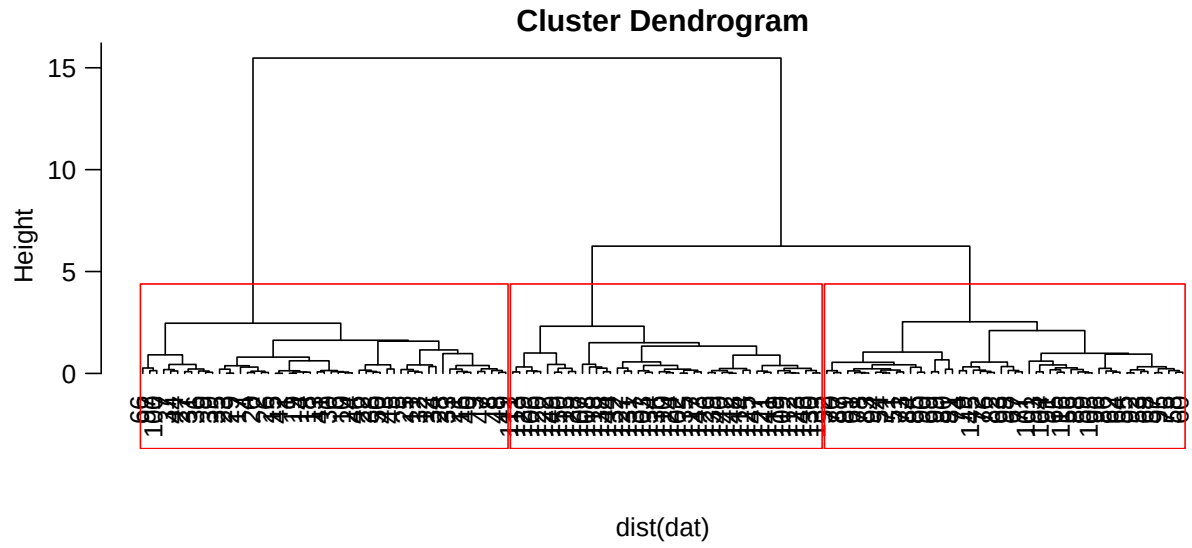

Abbildung 4.11: Dendrogramme mit Klassenunterteilung.

4.6 Heatmaps

Eine **Heatmap** dient der Visualisierung einer Datenmatrix. Die Visualisierung beruht auf zwei Grundelementen. Einerseits werden die Werte in den Zellen der Datenmatrix durch farbige Quadrätchen representiert. Die Farbabstufung ist fein und geht von dunkel (für kleine Werte) zu hell (für grosse Werte), üblicherweise in einem Farbschema wie z.B. dunkelrot-rot-orange-gelb-weiss. Anderserseits werden die Zeilen und Spalten so permutiert, dass ähnliche Zeilen respektive Spalten nahe beieinander platziert werden. Da die Permutation durch hierarchische Clusteranalyseverfahren ermittelt werden, wird die Darstellung am Rande mit den entsprechenden Dendrogrammen ergänzt.

Heatmaps erlauben, besser zu verstehen, in welchen Merkmalen sich die resultierenden Gruppen in einer hierarchischen Clusteranalyse unterscheiden. Allerdings sind die Resultate wie bei allen in diesem ersten Teil vorgestellten Methoden sehr sensitiv auf die Datenaufbereitung (Transformationen), die gewählten Distanzmasse und das verwendete hierarchische Clusteranalyseverfahren.

In R gibt es verschiedene Funktionen, mit denen Heatmaps erzeugt werden können. Eine davon, `heatmap`, steht schon in der Basisversion von R zur Verfügung. Über die Argumente können viele Optionen gesteuert werden. Lesen Sie dazu die Hilfe durch. Eine elegante Alternative ist auch die Funktion `pheatmap` aus dem Paket `pheatmap`.

Als Beispiel betrachten wir einen Datensatz mit den Informationen MPG (miles per gallon; Verbrauch), weight, Drive_Ratio (Achsenübersetzung), Displacement (Hubraum) und Cylinders von 38 Fahrzeugen aus verschiedenen Ländern. In der Defaultversion von `heatmap` wird für das Cluster die Funktion `hclust` mit der Complete-Linkage Methode verwendet. Standardmässig werden die Daten über die Zeilen skaliert (Argument `scale`). Das Clustering kann mit dem Argument hclustfun auch angepasst werden. Standardmässig werden die Daten über die Zeilen skaliert (Argument

`scale="row"`). Auch das Distanzmass (default `distfun=dist`). Für weitere Optionen wird auf die Hilfe verwiesen.

```
heatmap(x) # complete linkage
heatmap(x, distfun = dist, hclustfun =function(d) hclust(d, method="ward.D2"), scale="row")
```

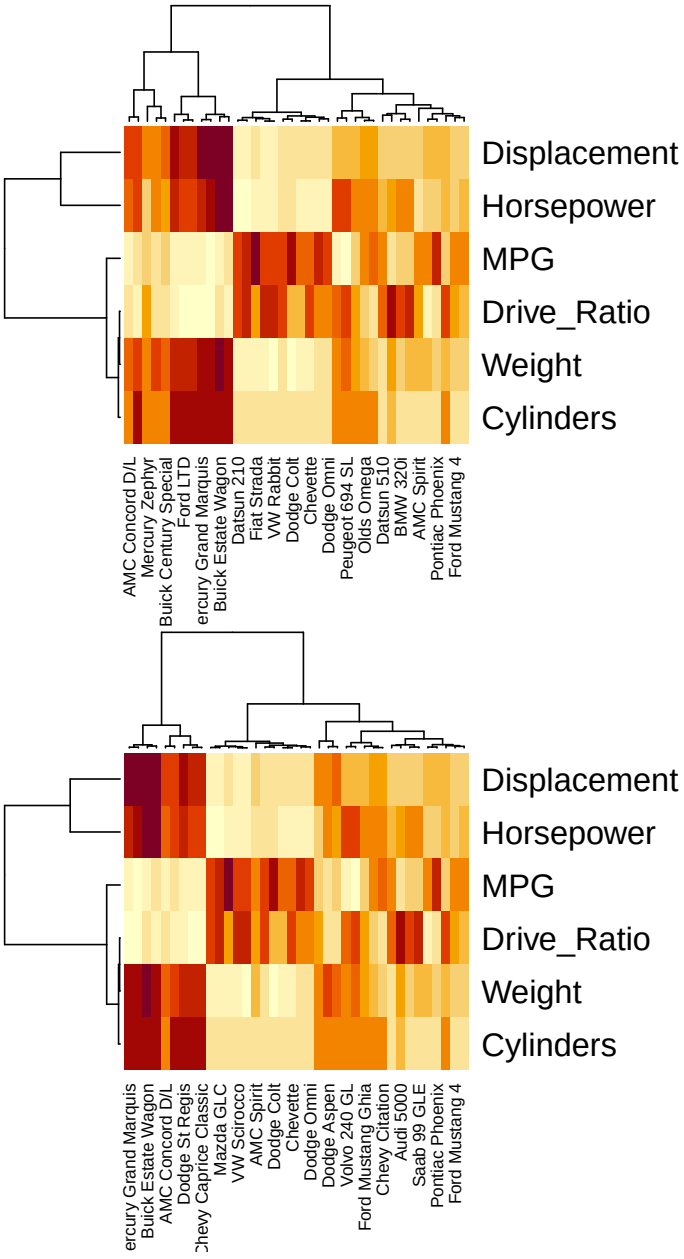

Abbildung 4.12: Heatmaps mit der Funktion headmap von Fahrzeugdaten (oben mit complete Linkage Clustering, unten mit der Methode von Ward.

Mit der Funktion `pheatmap` lässt sich das Ergebnis noch besser darstellen. Die Skalierung muss bei dieser Funktion explizit angegeben werden (`scale="row"`). Die Clusteringmethode kann auch

gewählt werden (`clustering_method`). Zudem gibt es weitere interessante Optionen, z.B. kann zusätzlich eine kategorielle Variable, hier das Produktionsland, mitabgebildet werden (Argument `annotation_col`). Das hilft Assoziationen zwischen Clustern und der externen kategorialen Variablen zu analysieren.

```
library(pheatmap)
rownames(country) <- colnames(x) #Verknüpfung mit colnames von x
pheatmap(x, scale="row", clustering_method = "average", annotation_col=country)
```

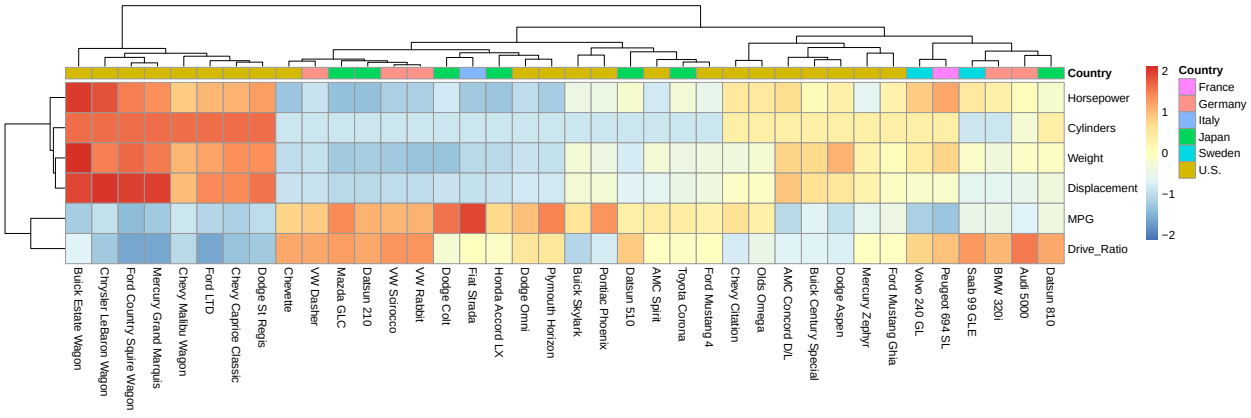

Abbildung 4.13: Heatmaps mit der Funktion pheatmap von Fahrzeugdaten.

4.7 Modellbasiertes Clustern

Siehe z.B. Everitt & Hothorn (2011) oder Scrucca et al. (2016).

Teil II

Überwachtes Lernen

5 Klassifikation am Beispiel des Nächste-Nachbarn-Klassifizierer

Bei der **Klassifikation** gehen wir davon aus, dass eine Grundgesamtheit aus mehreren Teilgesamtheiten, Klassen genannt, besteht, so dass jedes Objekt der Grundgesamtheit genau einer Klasse angehört. Das Ziel der Klassifikationsverfahren ist es, ein Objekt aus dieser Grundgesamtheit, dessen Klassenzugehörigkeit *unbekannt* ist, *aufgrund seiner Merkmale* derjenigen Klasse zugeordnet, der es auch entstammt. Solche Problemstellungen treten in verschiedenen Bereichen, von der Vorhersage der Kreditwürdigkeit über die Identifizierung von Schriftzeichen bis zur medizinischen Diagnostik, auf.

Gehen wir noch etwas näher auf das Beispiel der Kreditwürdigkeit ein. Bei der Kreditvergabe zerfallen die potentiellen Kunden einer Bank in natürlicher Weise in zwei Klassen, nämlich Kunden, die ihre Zahlungen ordnungsgemäss abwickeln, und solche bei denen Verzögerungen der Ratenzahlungen bis hin zum Kreditausfall auftreten. Auf der Basis einer Reihe von Merkmalen des Kreditnehmers, die seine persönliche, wirtschaftliche und rechtliche Situation kennzeichnen, wird versucht, eine Entscheidung über zukünftige Vergabe oder Ablehnung des Kredits zu treffen.

Eine ähnliche Problemstellung haben wir schon in der **Cluster-Analyse** (vgl. Kapitel 4) kennen gelernt. Dort sucht man aufgrund von Merkmalsangaben Objekte so zu *gruppieren*, dass sich die entstandenen Gruppen möglichst klar unterscheiden. Der wesentliche Unterschied besteht also darin, dass im Gegensatz zur Klassifikation die Gruppen, zu welchen die Objekte zugeordnet werden sollen, noch unbekannt sind. Beide Vorgehensarten werden im Gebiet “machine learning” unter dem Stichwort **statistical learning** zusammengefasst. Cluster-Analyse ist dann **unsupervised** learning, weil die Gruppen a priori unbekannt sind, und Klassifikation ist **supervised**, weil üblicherweise aufgrund von gemachten Erfahrungen für eine Auswahl von Objekten bereits Klassenzugehörigkeiten bekannt sind.

5.1 Nächste-Nachbarn-Klassifizierer

Wir beginnen mit einer ganz einfachen Idee, wie man einen Klassifizierer (auch Klassifikator genannt) bauen kann. Es handelt sich um die Nächste-Nachbarn Regel.

Ausgehend von einer **Lernstichprobe** (oder auch Trainingsdatensatz genannt) bestehend aus den Merkmalsvektoren $x_1, \ldots, x_n$ und der Klassenindex $k_1, \ldots, k_n$, wird eine neue Beobachtung x_o derjenigen Klasse zugeordnet, zu der das am nächsten bei x_o liegende Objekt der Lernstichprobe,

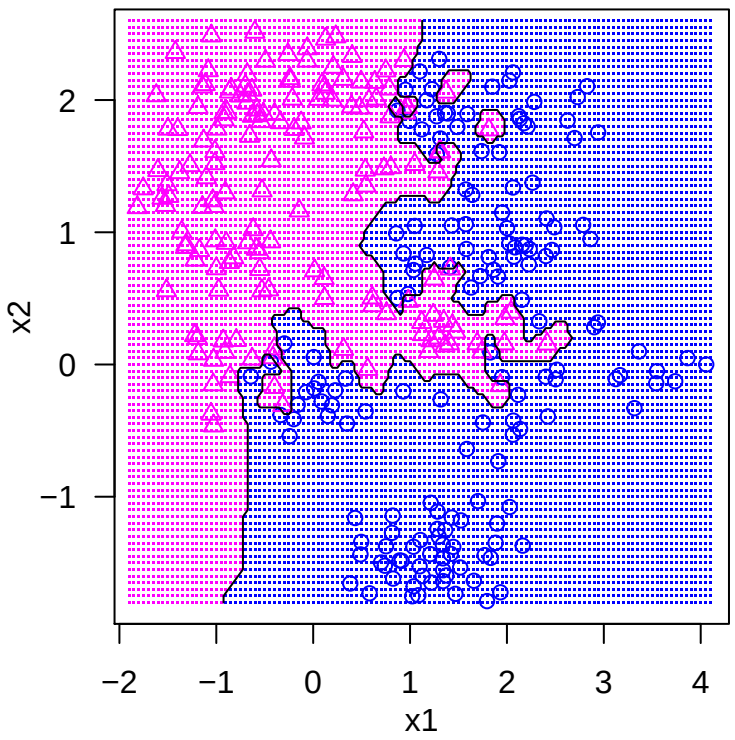

Abbildung 5.1: Synthetisches Klassifikationsbeispiel: Die beiden Klassen sind mit Kreisen und Triangel gekennzeichnet. Die Klassengrenzen ist mit der einfachen Nächste-Nachbarn-Regel bestimmt worden (ausgezogenen Linie). Die Punkte geben an, auf welchem Raster die Klassifikationsvorhersage gemacht wurde.

dem *nächsten Nachbarn*, gehört. Der nächste Nachbar x_{j_o} von x_o ist dabei bestimmt durch

$$j_o = \arg \min_{i=1,\ldots,n} d(x_o, x_i)$$

wobei $d()$ ein Distanzmass (oft die euklidische Metrik) ist.

In Abbildung **??** wird ein Streudiagramm von 320 Objekten gezeigt, die sich durch zwei intervallskalierte Merkmale beschreiben lassen. Jedes Objekt gehört entweder zur Klasse $\circ$ oder zur Klasse $\triangle$. Die Klassenzugehörigkeit eines neuen Objekts, von dem wir die beiden Merkmale kennen, jedoch nicht dessen Klassenzugehörigkeit, wollen wir mit der einfachen Nächste-Nachbarn-Regel vorhersagen. Wir können sogar die Klassengrenzen einzeichnen, falls wir ein potentiell neues Objekt über ein Raster von Merkmalspunkten laufen lassen und jeweils die Klassenzugehörigkeit schätzen.

Eine nahe liegende Erweiterung der Nächste-Nachbarn-Regel ist die **k-Nächste-Nachbarn-Regel**, bei der nicht nur der nächstgelegene Objekt berücksichtigt wird, sondern die k am nächsten bei der Beobachtung x_o liegenden Objekte der Lernstichprobe.

Sei unter den k Nächste-Nachbarn von x_o jeweils m_ℓ aus der Klasse ℓ, $\ell = 1, \ldots, g$. Also gilt $\sum_{\ell=1}^{g} m_\ell = k$. Die Beobachtung x_o wird nun derjenigen Klasse $\hat{\kappa}$ $(\in 1, \ldots, g)$ zugeordnet, für die

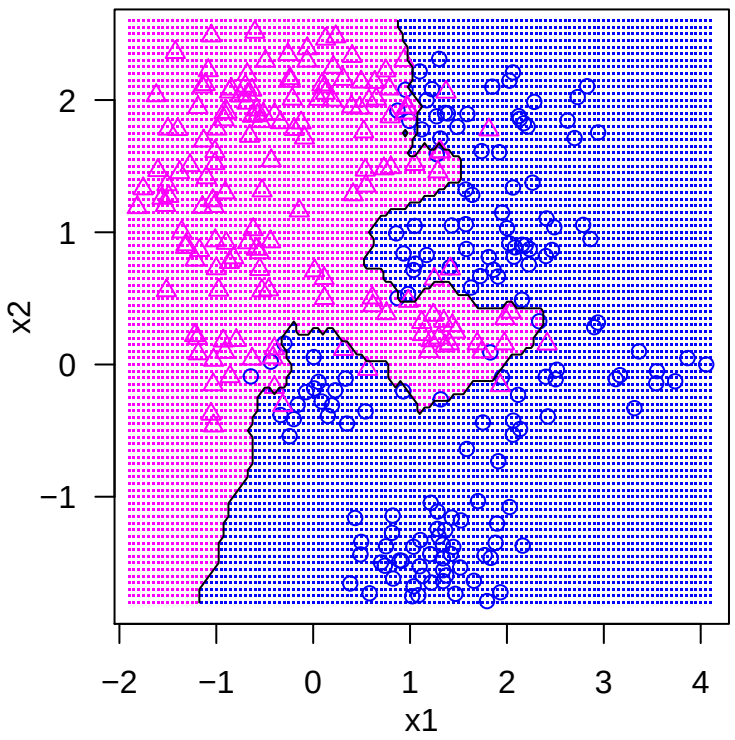

Abbildung 5.2: Synthetisches Klassifikationsbeispiel: Die beiden Klassen sind mit Kreisen und Triangel gekennzeichnet. Die Klassengrenzen ist mit den 5 Nächste-Nachbarn bestimmt worden (ausgezogenen Linie). Die Punkte geben an, auf welchem Raster die Klassifikationsvorhersage gemacht wurde.

gilt:

$$m_{\hat{\kappa}} = \max_{\ell=1,\ldots,g} m_\ell$$

Gegenüber der einfachen Nächste-Nachbarn-Regel ergeben sich mit der k-Nächste-Nachbarn-Regel deutlich besser zusammenhängende Gebiete gleicher Klassenzugehörigkeit je grösser k ist (vgl. Abbildung 5.2, wo die gleichen Daten wie in 5.1 jedoch zur Klassifikation die 5 nächsten Nachbarn verwendet wurden). Weil mehr Objekte für die Schätzung der Klassengrenzen verwendet werden können, ist die Schätzung bei grossem k mit einem kleineren statistischen Fehler verbunden. Allerdings kann bei zu grossem k die Gebiete gleicher Klassenzugehörigkeit zu grossräumig werden und echte "Buchten" und "Inseln" sind darin nicht mehr erfasst – man spricht vom **Bias des Klassifikators**. Dies wird zu Fehlklassifikationen führen. Die **Fehlklassifikationsrate** misst den Anteil an Fehlklassifikation und wird als Anzahl Fehlklassifikationen dividiert durch die Anzahl zu klassifizierenden Objekte berechnet.

Andererseits, je kleiner k desto weniger Fehlklassifikationen kommen in der Lernstichprobe vor. Bei $k = 1$ gibt es keine Fehlklassifikationen. Das ist aber nur ein Scheinerfolg, indem die Klassengrenze an die Lernstichprobe zu gut angepasst wurde. Nur mit *neuen* Stichproben zeigen sich dann in der

Regel die wahren Grössenordnung der Fehlklassifikationen.

Aus dieser Beobachtung ist zu vermuten, dass sowohl ein zu kleines wie auch ein zu grosser k ungünstig für die Bestimmung der Klassifikationsgrenzen ist und das irgendwo in der Mitte ein optimales k existiert. Mehr dazu in Abschnitt 5.2.

Praktische Hinweise: Wählen Sie bei *zwei* Klassen immer eine *ungerade* Anzahl nächste Nachbarn, da damit immer eine eindeutige Wahl gefällt werden kann. Hat man mehr als zwei Klassen, so ist es, unabhängig von der Wahl von k, nicht mehr immer möglich, eine eindeutige Zuordnung vorzunehmen. Falls sich ein Patt in der Zuordnung einstellt, wird die prognostizierte Klassenzugehörigkeit ausgewürfelt, damit nicht willkürlich eine Klasse bevorzugt wird. Dieses Vorgehen hat aber den Effekt, dass bei mehreren Durchführungen der Klassifikation sich unterschiedliche prognostizierte Klassenzugehörigkeiten ergeben und somit unterschiedliche Fehlerraten ermittelt werden. Um reproduzierbare Resultate zu erhalten, kann vor der Klassifikation der Seed im Zufallszahlengenerator auf einen festen Wert gesetzt werden. Allerdings geht damit theoretisch der Vorteil der zufälligen Wahl verloren, was praktisch jedoch leicht zu verschmerzen ist.

Der Vorteil der Nächste-Nachbarn-Zuordnungsregeln liegt in ihrer Einfachheit und darin, dass keinerlei weiteren Informationen über die Daten nötig sind. Sie haben allerdings den grossen Nachteil, dass für jede Klassifikation die gesamte Lernstichprobe zur Verfügung stehen und durchgearbeitet werden muss. In hochdimensionalen Problemen muss zudem diese Lernstichprobe extrem gross sein, damit die Klassengrenzen zuverlässig geschätzt werden können. Ein weiterer Schwachpunkt ist die Wahl der Distanzfunktion. Spätestens wenn die Objektmerkmale mit verschiedenen Variablentypen beschrieben wurden, stellt sich die Frage nach der geeigneten Wahl der Distanzfunktion. Wie wir im Unterkapitel 3.2 gesehen haben, kann dies eine echte Herausforderung sein.

In R kann der Nächste-Nachbarn-Klassifizierer mit der Funktion `knn` aus dem Paket `class` ausgeführt werden.

5.2 Performance-Evaluation

Im letzten Kapitel haben wir den Nächsten-Nachbar-Klassifizier kennengelernt. Wie gut ist dieser Klassifizier nun? Welche Anzahl an Nachbaren sollte man berücksichtigen? Sobald wir verschiedene Klassifizierer betrachten, stellt sich die Frage, welcher im konkreten Fall am besten geeignet ist.

In den folgenden Abschnitten wollen wir einige Masse zu Evaluation der Modellperformance betrachten.

Für die folgenden Ausführungen gehen wir von einem Zwei-Klassen-Klassifikationsproblems aus. Die beiden Klassen werden dabei vorzugsweise als positiv und negativ bezeichnet. Die betrachtete Stichprobe wird dann aus P positiven und N negativen Objekten bestehen. Der Klassifizierer ordnet jedem Objekt eine Klasse zu, wobei einige der Zuordnungen falsch sein können. Um den Klassifizierer zu beurteilen, wird nun die Anzahl der richtig Positiven (TP - “true positive"), richtig Negativen

(TN), falsch Positiven (FP) (das sind Negative, die durch den Klassifizierer als Positive vorhergesagt werden) und die falsch Negativen (FN) (das sind Positive, die durch den Klassifizierer als Negative vorhergesagt werden) erfasst. Diese Anzahlen können effizient in einer Kreuztabelle festgehalten werden:

		Vorhergesagte Zugehörigkeit		
		positiv	negativ	
Tatsächliche	positiv	true positive (TP)	false negative (FN)	P
Zugehörigkeit	negativ	false positive (FP)	true negative (TN)	N

Der Klassifizierer ordnet also ($TP + NP$) Objekte den Positiven zu und ($TN + FN$) Objekte den Negativen. In der statistischen Testtheorie würde man FP mit Fehler 1. Art und FN mit Fehler 2. Art bezeichnen, sofern die Null-Hypothese mit "negativ' ' ierntifiziert wird. In Data Mining wird diese ganze Kreuztabelle als **Konfusionsmatrix**[5] (*engl.* confusion matrix) bezeichnet.

Verschiedene Performance-Masse werden in der Praxis in Erwägung gezogen. So z.B.

Bezeichnung	Berechnung	weitere Bezeichnungen
FP-Rate	$\frac{FP}{N}$	Ausfallquote ("fall-out"), "false alam rate"
TP-Rate	$\frac{TP}{P}$	Sensitivität ("sensitivity"), Trefferquote ("recall", "hit rate")
TN-Rate	$\frac{TN}{N}$	Spezifität ("specificity")
Antwortrate	$\frac{TP}{TP+FP}$	(response rate), Wirksamkeit, "precision", Relevanz
accuracy	$\frac{TP+TN}{P+N}$	(Korrektklassifikationrate)
Fehlerrate	$\frac{FP+FN}{P+N}$	("error rate"), 1-accuracy

Bei einigen Performance-Masse wie FP-Rate, TP-Rate oder Antwortrate wird die Klasse "Positive" bevorzugt behandelt. Je nach Anwendungsgebiet ist dies auch in der Klassifikation zu rechtfertigen. Zum Beispiel in der "fraud-detection" will man ein besonderes Auge auf die Betrüger richten. Obwohl das Betrügen nicht positiv im eigentlichen Sinn ist, können die entsprechenden Masse doch verwendet werden.

Im Data Mining ist eine der am häufigsten verwendeten Performance-Masse die **Fehlerrate**. Um diese für einen gegebenen Datensatz und einen konkreten Klassierer zu bestimmen, wird sie aus der Konfusion-Matrix berechnet, welche wiederum aufgrund der Resultate aus der Lernstichprobe ermittelt wurde.

Leider tendiert die so bestimmte (d.h. geschätzte) Fehlerrate die wahre Fehlerrate (d.h. die Fehlklassifikationswahrscheinlichkeit) zu unterschätzen. Dies ist besonders bei kleinen Stichprobengrössen P

[5] Anmerkung: Haben wir nicht nur zwei Klassen sondern sehr viele, so wird die Dimension der Konfusionsmatrix sehr gross, wie z.B. in der Handschriftenerkennung oder in der Spracherkennung. Um dann nicht den Überblick zu verlieren, kann die Konfusionsmatrix als Ähnlichkeitsmatrix betrachtet werden und mit Methoden der multivariaten Skalierung visualisiert werden. Allerdings muss das Problem der Asymmetrie gelöst werden.

und N augenfällig. Im Wesentlichen kommt die zu optimistische Schätzung der Fehlerrate daher, dass die gleichen Objekte zur Bestimmung des Klassifizierers wie auch zu seiner Beurteilung (mittels der Fehlerrate) verwendet werden.

Wir nennen deshalb die Fehlerrate, die aufgrund der Lernstichprobe bestimmt wird, die **in-sample** Fehlerrate.

Beispiel: Die Konfusions-Matrix für das Beispiel aus dem letzten Kapitel bei dem der k-Nächste-Nachbarn-Klassifizierer mit $k = 5$ auf die Daten angewandt wurde, ist

		Vorhergesagte Zugehörigkeit	
		Klasse 1	Klasse 2
Tatsächliche	Klasse 1	145	18
Zugehörigkeit	Klasse 2	15	142

Daraus ergibt sich eine in-sample Fehlerrate von

$$\mathrm{FR}_L = \frac{18 + 15}{145 + 15 + 18 + 142} = 0.103\,,$$

oder 10.3%. Dies besagt, dass ca. jedes zehnte Objekt der falschen Klasse zugeordnet wird. Ob das hoch oder niedrig ist, hängt von den konkreten Anforderungen an den Klassifizierer ab.

Beachten Sie, dass der Nächste-Nachbarn-Klassifizierer mit $k = 1$ eine in-sample Fehlerrate von 0% hat, also bezüglich der in-sample Fehlerrate viel besser abschneidet als der Nächste-Nachbarn-Klassifizierer mit $k = 5$. }

Schätzung durch Teststichprobe: Besser ist es, wenn die Gesamtstichprobe *zufällig* in eine **Lern-** und eine **Teststichprobe** aufgeteilt wird und die Fehlerrate aus der Teststichprobe geschätzt wird, wobei die Teststichprobe keine Berücksichtigung in der "Lernphase' ' erfahren hat. Die Objekte aus der Teststichprobe werden mit dem aus der Lernstichprobe geschätzten Klassifizierers den Klassen zugeordnet und der Anteil der Fehlklassifikationen berechnet. Dieses Vorgehen liefert eine *erwartungstreue* Schätzung der Fehlklassifikationswahrscheinlichkeit. Damit weder die Lern- noch die Teststichprobe zu klein sind, ist allerdings eine grosse Gesamtstichprobenumfang nötig.

Anmerkung: Bei einer Aufteilung der Gesamtstichprobe muss man eine gewisse Vorsicht walten lassen. Achten Sie darauf, dass bei Faktorvariablen alle Faktorstufen genügend oft in der Lernstichprobe vorkommen. Ansonsten hat man für die betroffenen Faktorstufen keinen Schätzwert und somit für das betroffene Objekt auch keine geschätzte (eigentlich prognostizierte) Klassenzugehörigkeit angeben.

Kreuzvalidierung. Um einen zufälligen grossen positiven oder negativen Bias in der geschätzten Fehlerrate bei nur einmaliger Unterteilung der Stichprobe zu vermeiden, teilt man die Gesamtstichprobe besser *mehrmals* zufällig in Lern- und Teststichprobe auf und mittelt die aus den jeweiligen Teststichproben erhaltenen Fehlerraten. Dieses Verfahren wird **Kreuzvalidierung** (engl. **cross**

validation) genannt. Ein Spezialfall davon ist die **jackknife** oder **leave-one-out Methode**. Dabei werden die Klassfizierer unter Weglassen *eines* Objektes der Gesamtstichprobe geschätzt und anschliessend dieses Objekt klassifiziert. Dies wird für sämtliche Objekte durchgeführt und die Fehlerrate durch den Anteil der fehlklassifizierten Objekte geschätzt.

Die leave-one-out Kreuzvalidierung ist heutzutage sehr populär und scheint sich in der Praxis zu bewähren. Die Schätzung der Fehlerrate aus der leave-one-out Methode besitzt allerdings den Nachteil, dass (sie recht rechenintensiv und) ihre Varianz besonders für kleine Gesamtstichprobenumfänge gross ist. Im Einzelfall kann dies wiederum bedeuten, dass die Schätzung der Fehlerrate recht dürftig sein kann. Um diesen Effekt etwas abzuschwächen, sehen Varianten der Kreuzvalidierung vor, nicht nur ein Objekt wegzulassen sondern mehrere (**d.h. Kreuzvalidierung mit x%-iger Auslassung**). Da dies nicht mehr für alle Kombinationen berechenbar ist, wird eine *zufällige* Auswahl getroffen. Folglich können bei einer zu anderen Berechnung unterschiedliche Fehlerraten herauskommen, die sich eben mit der zufälligen Auswahl der ausgelassenen Werte erklären lassen.

Anmerkung: Beachten Sie bei einer Kreuzvalidierung mit x%-iger Auslassung, dass entweder die Auslassung so klein ist, dass nie eine Faktorstufe ganz weggelassen werden kann, oder die Auslassung geschichtet vorgenommen wird, die sicherstellt, dass keine Faktorstufe in der Lernstichprobe nicht vorhanden ist (vgl. auch Anmerkung in 5.2).

6 Klassifikationsbäume

Klassifikationsbäume sind intuitive und für viele Personen *das* nahe liegende Verfahren, um Objekte mittels einigen Fragen, die nur mit "ja/nein" oder "wahr/falsch" beantwortet werden müssen, einer Klasse zuzuordnen. Dabei hängt die als nächstes zu stellende Frage von der Antwort der zuvor gestellten Frage ab. Die Fragen und die Entscheidung, zu welcher Klasse man am Ende der Fragerei das Objekt zuordnen will, kann in einer Baumstruktur festgehalten werden. Diese anschauliche Darstellung des Resultats erleichtert die Kommunikation mit Fachleuten und Auftraggebern, die für statistische Details wenig Interesse zeigen.

7 Weiterführende Literatur

- Everitt, Brian, & Hothorn, Torsten. (2011). *An Introduction to applied multivariate analysis with R (Use R!).* New York: Springer.
- Hastie, Trevor, Tibshirani, Robert, & Friedman, Jerome H. (2017). *The elements of statistical learning : Data mining, inference, and prediction* (Second edition, corrected at 12th printing 2017 ed., Springer series in statistics). New York NY: Springer.
- Irizarry, Rafael A (2019). *Introduction to Data Science : Data Analysis and Prediction Algorithms with R.* Boca Raton: CRC, 2020. Print. Chapman & Hall/CRC Data Science Ser.
- James, Gareth, Witten, Daniela, Hastie, Trevor, & Tibshirani, Robert. (2013). *An Introduction to Statistical Learning: With Applications in R* (Vol. 103, Springer texts in statistics). New York, NY: Springer New York.
- Kaufman, Leonard, & Rousseeuw, Peter J. (2005). *Finding groups in data : An introduction to cluster analysis* (Wiley series in probability and statistics). Hoboken, N.J: Wiley.
- Scrucca, Luca, Michael Fop, T. Brendan Murphy, & Adrian E. Raftery. (2016). *mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models*, The R Journal 8, 289–317.
- Soman, K., Diwakar, Shyam, & Ajay, V. (2006). *Insight Into Data Mining: Theory and Practice*, PHI Learning Private Limited, Delhi.
- van der Maaten, Laurens, & Hinton, Geoffrey (2008). *Visualizing Data using t-SNE.* Journal of Machine Learning Research 9. 2579-2605.
- Varmuza, Kurt, & Filzmoser, Peter. (2009). *Introduction to multivariate statistical analysis in chemometrics.* Boca Raton: Taylor & Francis.

Anhang

A PCA: Konstruktion der Hauptkomponenten

Im Folgenden werden Matrizen dadurch gekennzeichnet, dass sie unterstrichen sind.

Bei der PCA werden neue Variablen $Y_1, Y_2, \ldots, Y_p$ - die sogenannten Hauptkomponenten - konstruiert, indem Linearkombinationen der ursprünglichen, zentrierten Variablen $X_1, X_2, \ldots, X_p$ gebildet werden. Um die k-te PC zu konstruieren (oder in Beobachtungssichtweise den Koordinatenwert einer Beobachtung bzgl. der k-ten PC) wird also eine Linearkombination der Originalvariablen gebildet.

$$Y_k = a_{1k}X_1 + a_{2k}X_2 + \ldots + a_{pk}X_p = (a_{1k}\, a_{2k}\, \ldots\, a_{pk}) \cdot \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix} = a_k^t \cdot X$$

Dies gilt für alle Komponenten $k = 1, 2, \ldots, p$ und kann durch p solcher Gleichungen ausgedrückt werden oder (einfacher wie der Mathematiker findet) durch nur eine Gleichung in Matrixschreibweis.

$$y = x^t \cdot \underline{A} = (x_1 \ldots x_p) \cdot \begin{pmatrix} a_{11} & \ldots & a_{p1} \\ \vdots & \ddots & \vdots \\ a_{p1} & \ldots & a_{pp} \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + \ldots + a_{1p}x_p \\ \vdots \\ a_{1p}x_1 + \ldots + a_{pp}x_p \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_p \end{pmatrix}$$

Dabei ergibt sich der neue p-dimensionalen Merkmalsvektors $y = (y_1, \ldots, y_p)$, indem der (zentrierte) Merkmalsvektors $x = (x_1, \ldots, x_p)$ mit der Rotationsmatrix $\underline{A}$ multipliziert wird. Im Normalfall haben wir nicht nur 1 p-dimensionale Beobachtung x, sondern wir haben n Beobachtungen, wobei für jede Beobachtung die Werte der p Merkmale gemessen wurde. Die erste Beobachtung hat den Merkmalsvektors $x^{(1)} = (x_{11}, \ldots, x_{1p})$, die zweite Beobachtung hat den Merkmalsvektors $x^{(2)} = (x_{21}, \ldots, x_{2p})$ usw. Diese Beobachtungen können in einer $(n \times p)$ Datenmatrix $\underline{X}$ zusammengefasst werden, wobei jede der n Zeilen eine p-dimensionale Beobachtung enthält (die Spalten enthalten die einzelnen Merkmale und sind auf 0 zentriert).

$$\underline{X} = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(n)} \end{pmatrix} = \begin{pmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{p1} & \ldots & x_{pp} \end{pmatrix}$$

Die Berechnung der neuen Merkmalsvektoren kann für alle n Beobachtungen gleichzeitig durchgeführt werden, indem die Datenmatrix $\underline{X}$ mit der Rotationsmatrix $\underline{A}$ multipliziert wird.

$$\underline{Y} = \underline{X} \cdot \underline{A} = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{p1} & \dots & x_{pp} \end{pmatrix} \cdot \begin{pmatrix} a_{11} & \dots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{p1} & \dots & a_{pp} \end{pmatrix}$$

Die Einträge der $(p \times p)$ Matrix $\underline{A}$ werden oft als "loadings" bezeichnet und die Einträge der $(n \times p)$ Matrix $\underline{Y}$ als "scores".

$$\underline{Y} = \begin{pmatrix} y_{11} & \dots & y_{1p} \\ \vdots & \ddots & \vdots \\ y_{p1} & \dots & y_{pp} \end{pmatrix} = \begin{pmatrix} a_{11}x_{11} + \ldots + a_{p1}x_{1p} & \dots & a_{1p}x_{11} + \ldots + a_{pp}x_{1p} \\ \vdots & \ddots & \vdots \\ a_{11}x_{n1} + \ldots + a_{p1}x_{np} & \dots & a_{1p}x_{n1} + \ldots + a_{pp}x_{np} \end{pmatrix}$$

Die Aufgabe besteht nun darin, die Einträge der Matrix $\underline{A}$ für die Rotation bzw. lineare Transformationen zu bestimmen, wobei eine Beobachtung durch die ursprünglich beobachteten (zentrierten) Merkmalen $X = (X_1, \ldots, X_p)$ beschrieben wird:

- **PC1**: finde $a^{(1)} = (a_{11}, \ldots, a_{p1})$, so dass $Y_1 = a^{(1)T}X$ maximale Varianz hat, wobei die Nebenbedingung $|a^{(1)}| = 1$ erfüllt sein muss.
- **PC2**: finde $a^{(2)} = (a_{12}, \ldots, a_{p2})$, so dass $Y_2 = a^{(2)T}X$ - unter der Nebenbedingung Y_2 ist unkorreliert mit Y_1 - maximale Varianz hat und $|a^{(2)}| = 1$ gilt.
- **PC3**: finde $a^{(3)} = (a_{13}, \ldots, a_{p3})$, so dass $Y_3 = a^{(3)T}X$ - unter der Nebenbedingung Y_3 ist unkorreliert mit (Y_1, Y_2) - maximale Varianz hat und $|a^{(3)}| = 1$ gilt.
- usw.

Um die erste Hauptkomponente zu finden müssen wir also die Varianz von Y_1, die eine Funktion der p Koeffizienten $a_{11}, \ldots, a_{p1}$ ist, maximieren und gleichzeitig die Nebenbedingung einhalten, dass der Betrag des Koeffizienten Vektors $|a^{(1)}| = 1$ ist.

Um eine Funktion unter Nebenbedingungen zu maximieren, kann die Methode der *Lagrangefunktion* verwendet werden. Hier werden die mathematischen Details nicht diskutiert, sondern nur die analytische Lösung präsentiert, die sich damit ergibt. Als Lösung ergibt sich, dass die Spalten der Rotationsmatrix $\underline{A}$ durch die Eigenvektoren der Varianz-Kovarianz-Matrix (oft auch nur Kovarianz-Matrix genannt) Σ gegeben sind. Es ist keine Überraschung, dass die Kovarianz-Matrix hier eine Rolle spielt, da es darum geht die Varianzen der Daten in geeigneter Weise "einzufangen".

Neben der Lagrange-Methode kann dieses Ergebnis auch mit etwas linearer Algebra hergeleitet werden, was im Folgenden etwas genauer skizziert werden soll.

Eine Bedingung bei der PCA ist, dass die Hauptkomponenten unkorreliert sind, d.h. die Korrelation zwischen beliebigen 2 PCs jeweils Null ist. Dies führt zu einer besonders einfachen Varianzstruktur, bei der die Kovarianz-Matrix der Hauptkomponenten (siehe unten), die bis auf die Diagonale nur mit Nullen besetzt ist, da eben alle Kovarianzen Null sind.

An dieser Stelle sei noch einmal an die Definitionen der Varianz, Kovarianz und der Korrelation ρ eines Merkmals erinnert, wobei wir weiter annehmen, dass wir n Beobachtung und p Merkmale beobachten.

Varianz des k-ten Merkmals:

$$\sigma_k^2 = var(X_k) = \frac{1}{n}\sum_{i=1}^{n}(X_{ik} - \overline{X}_{ik})^2$$

Kovarianz zwischen dem k-ten und s-ten Merkmal:

$$\sigma_{kl} = cov(X_k, X_s) = \frac{1}{n}\sum_{i=1}^{n}(X_{ik} - \overline{X}_{ik})(X_{is} - \overline{X}_{is})$$

Korrelation zwischen dem k-ten und s-ten Merkmal:

$$\sigma_{kl} = cor(X_k, X_s) = \frac{\sigma_{ks}}{\sigma_k \cdot \sigma_s} = \frac{cov(X_k, X_s)}{\sqrt{var(X_k)} \cdot \sqrt{var(X_s)}}$$

(Varianz-) Kovarianz-Matrix:

$$\underline{\Sigma}_X = \frac{1}{n}\underline{X}^t\underline{X} = \begin{pmatrix} \sigma_1^2 & \dots & \sigma_{1p} \\ \vdots & \ddots & \vdots \\ \sigma_{p1} & \dots & \sigma_p^2 \end{pmatrix}$$

(Standardabweichung-) Korrelation-Matrix:

$$\underline{\Sigma}_{X,std} = \begin{pmatrix} 1 & \dots & \rho_{1p} \\ \vdots & \ddots & \vdots \\ \rho_{p1} & \dots & 1 \end{pmatrix}$$

Da die Hauptkomponenten unkorreliert sind, also alle Kovarianzen bzw. Korrelationen zwischen je zwei Hauptkomponenten Null sind, erwarten wir, dass die Kovarianz-Matrix $\underline{\Sigma}_Y$ der Hauptkomponenten eine Diagonalmatrix ist: (Varianz-) Kovarianz-Matrix der PCs:

$$\underline{\Sigma}_Y = \frac{1}{n}\underline{Y}^t\underline{Y} = \begin{pmatrix} \sigma_{y_1}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_{y_p}^2 \end{pmatrix}$$

Nach der Wiederholung der (Ko-)Varianz-Definitionen kehren wir zur Konstruktion der Hauptkomponenten $Y_1, \dots, Y_p$ zurück, welche sich durch eine Drehung der (zentrierten) ursprünglichen Merkmale $X_1, \dots, X_p$ ergeben.

Wie oben diskutiert, wird dabei die Datenmatrix $\underline{X}$ mit der Rotationsmatrix $\underline{A}$ multipliziert. Hauptachsentransformationsgleichung: $\underline{Y} = \underline{X} \cdot \underline{A}$

Aus einem Theorem der linearen Algebra (Spektralzerlegung) ist bekannt, dass jede symmetrischen Matrix $\underline{\Sigma}$ (und eine Kovarianz-Matrix ist immer symmetrisch da $cov(a, b) = cov(b, a)$ gilt) sich darstellen lässt als $\underline{\Sigma} = \underline{E\Lambda E^t}$, wobei die Matrix $\underline{E}$ gerade spaltenweise aus den standardisierten Eigenvektoren e_i von $\underline{\Sigma}$ gebildet wird und die Diagonalmatrix Λ die dazugehörigen sortierten Eigenwerte λ_i als Hauptdiagonal-Elemente enthält.

Spektralzerlegung:

$$\underline{\Sigma} = \underline{E\Lambda E^t},\ mit\ \underline{\Lambda} = \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_p \end{pmatrix}$$

$$\underline{\Sigma} = \begin{pmatrix} e_{11} & \dots & e_{1p} \\ \vdots & \ddots & \vdots \\ e_{p1} & \dots & e_{pp} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_p \end{pmatrix} \cdot \begin{pmatrix} e_{11} & \dots & e_{p1} \\ \vdots & \ddots & \vdots \\ e_{1p} & \dots & e_{pp} \end{pmatrix}$$

Die p standardisierte Eigenvektoren der $(p \times p)$-dimensionalen Kovarianz-Matrix $\underline{\Sigma}_X$ sind p-dimensionale Vektoren $e^{(1)} = (e_{11}, \dots, e_{p1}), \dots, e^{(p)} = (e_{1p}, \dots, e_{pp})$ mit Länge 1, d.h. $|e_k| = 1$. Man spricht von einem standardisierten Eigenvektor einer Matrix, wenn sich bei der Multiplikation von Matrix und Eigenvektor $e_k (\neq 0)$ ein Vielfaches des Eigenvektor ergibt, wobei der Faktor der zugehörige Eigenwert λ_k ist und die Länge des Eigenvektor $|e_k| = 1$ ist. Zu jedem Eigenvektor e_k gibt es einen Eigenwert λ_k, wobei die folgende Eigenwertgleichung gilt:

$$\Sigma \cdot e_k = \lambda_k \cdot e_k$$

Die Eigenwerte und damit auch die zugehörigen Eigenvektoren werden so durchnummeriert, dass gilt: $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$. . Aus der Definition der Kovarianz-Matrix, der Transformationsgleichung und der Spektralzerlegung ergibt sich unmittelbar, dass $\underline{\Sigma}_Y$ durch eine Diagonalmatrix gegeben ist, welche als Diagonalelement die sortierten Eigenwerten λ_k der Kovarianz Matrix $\underline{\Sigma}_X$ enthält:

$$
\begin{aligned}
\underline{\Sigma}_Y &= \frac{1}{n}\underline{Y}^t\underline{Y} && \text{(Definition der Kovarianzmatrix)}\\
&= \frac{1}{n}(\underline{X}\cdot\underline{X})^t\cdot(\underline{X}\cdot\underline{X}) && \text{(Transformationsgleichung)}\\
&= \frac{1}{n}(\underline{A}^t\cdot\underline{X}^t\cdot\underline{X}\cdot\underline{A}) = \underline{A}^t\cdot\left(\frac{1}{n}\cdot\underline{X}^t\cdot\underline{X}\right)\cdot\underline{A} && \text{(Matrix Rechenregeln)}\\
&= \underline{A}^t\cdot(\underline{\Sigma}_Y)\cdot\underline{A} && \text{(Definition der Kovarianzmatrix)}\\
&= \underline{A}^t\cdot\left(\underline{E\Lambda E^t}\right)\cdot\underline{A} && \text{(Spektralzerlegung von } \underline{\Sigma}_Y)\\
&\overset{\text{setze } \underline{A}=\underline{E}!}{=} \underline{\Lambda} && \text{(alle Kovarianzen sind Null)}
\end{aligned}
$$

Wird $\underline{A} = \underline{E}$ gesetzt, so ergibt sich für $\underline{\Sigma}_Y = \underline{\Lambda}$ also eine Diagonalmatrix, was bedeuted, dass die Hauptkomponenten Y_k, $k = 1, \ldots, p$, unkorreliert sind.

Der obigen Umformungen kann entnommen werden, dass sich die gewünschte Kovarianz-Struktur der Hauptkomponenten ergibt, wenn als Rotationsmatrix die Eigenvektor Matrix von $\underline{\Sigma}_X$ gewählt wird. Aus obigen Umformungen wurde ausserdem klar, dass sich die Kovarianz-Matrix der Hauptkomponenten mittels der Kovarianz-Matrix der ursprünglichen (zentrierten) Merkmale und (Überraschung!) der oben diskutierten Rotationsmatrix $\underline{A}$ darstellen lässt, d.h. $\underline{\Sigma}_Y = \underline{A\Sigma}_X\underline{A}^t$.

Die Spalten der $(p \times p)$-dimensionalen Rotationsmatrix $\underline{A}$ sind also (gemäss des Theorems der Spektralzerlegung) durch die jeweils p-dimensionalen standardisierten Eigenvektoren $e^{(1)} = (e_{11}, \ldots, e_{p1}), \ldots, e^{(p)} = (e_{1p}, \ldots, e_{pp})$ der $(p \times p)$-dimensionalen Kovarianz-Matrix $\underline{\Sigma}_X$ gegeben.

$$
\Sigma_X \cdot e_k = \lambda_k \cdot e_k
$$

$$
\underline{A} = \begin{pmatrix} a_{11} & \ldots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{p1} & \ldots & a_{pp} \end{pmatrix} = \begin{pmatrix} e_{11} & \ldots & e_{1p} \\ \vdots & \ddots & \vdots \\ e_{p1} & \ldots & e_{pp} \end{pmatrix} = (e^{(1)}, \ldots, e^{(p)})
$$

Damit haben wir die Koeffizienten a_{ij}, auch Ladungen genannt, gefunden, die zur Konstruktion der Hauptkomponenten benötigt werden. Sie sind gerade durch die Einträge der Eigenvektoren der Kovarianz Matrix der Originalvariablen gegeben.

$$
\underline{Y} = \underline{X}\cdot\underline{A} = \begin{pmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{p1} & \ldots & x_{pp} \end{pmatrix} \cdot \begin{pmatrix} a_{11} & \ldots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{p1} & \ldots & a_{pp} \end{pmatrix}
$$