Ludwig-Maximilian-Universität
München

Institut für Statistik

Bachelorarbeit

Analyse des Verhaltens verschiedener Clusterverfahren nach Imputation fehlender Werte

---

Autor:
Johannes Wunder

Betreuer:
Prof. Dr. Christian Heumann

Datum:
10. Februar 2014

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Bergkirchen, 10.Februar 2014

Johannes Wunder

Zusammenfassung

Diese Arbeit befasst sich mit der Frage, wie sich verschiedene Clusterverfahren im Falle imputierter fehlender Werte verhalten. Um diese Frage beantworten zu können, wird eine Simulationsstudie mit verschiedenen Datensätzen durchgeführt. Die Datensätze sind jeweils aus verschiedenen Gruppen zusammengesetzt, die durch die Clusterverfahren gefunden werden sollen. In diesen Datensätzen werden fehlende Werte erzeugt und anschließend mithilfe der multiplen Imputation wieder imputiert. Auf Grundlage des ursprünglichen Datensatz bzw. der imputierten Datensätze werden verschiedene Clusterverfahren angewendet und die Veränderungen der Ergebnisse betrachtet. Als primäres Bewertungskriterium dient dabei die Genauigkeit, die sich durch den Anteil der richtig klassifizierten Objekte widerspiegelt. Dabei stellt sich heraus, dass partitionierende und modellbasierte Clusterverfahren im Falle imputierter Datensätze besser für eine Clusteranalyse geeignet sind als hierarchische Clusterverfahren. Die Imputation führt aber stets zur Abnahme der Genauigkeit. Enthält der Datensatz gemischt skalierte Variablen, so scheint bei keinen der angewandten Clusterverfahren eine Clusteranalyse Sinn zu machen, da die Genauigkeiten durch die Imputation stark verändert werden.

Inhaltsverzeichnis

1 Einleitung

Eine Clusteranalyse wird immer dann durchgeführt, wenn es darum geht, in einer heterogenen Gruppe von Objekten kleinere, homogene Teilgesamtheiten (sogenannte Cluster) zu finden. Sie umfasst eine Vielzahl verschiedener Verfahren, die das Auffinden der Cluster ermöglichen. Anwendung findet die Clusteranalyse in vielen Bereichen der Wissenschaft, sei es um in der Chemie Typologien zu erstellen oder in der Marktforschung verschiedene Konsumentengruppen zu bilden. All diese Verfahren basieren auf vollständig beobachteten Datensätzen. Objekte, die fehlende Werte aufweisen, können keiner Gruppe zugeordnet werden. Doch leider sind in der Realität viele Datensätze von fehlenden Werten betroffen. Will man jedoch auch Objekte mit fehlenden Beobachtungen einer Gruppe zuordnen, muss man für diese fehlenden Werte passende Ersatzwerte finden bzw. imputieren. Für die Imputation fehlender Beobachtungen steht ebenfalls ein weites Spektrum an Methoden zur Verfügung. Sie ermöglichen es, unvollständige Datensätze zu vervollständigen und somit auch die Durchführung einer Clusteranalyse.

In dieser Arbeit soll untersucht werden, wie sich verschiedene Clusterverfahren verhalten, wenn fehlende Daten imputiert werden. Dies geschieht mithilfe einer Simulationsstudie, in der man anhand selbst erzeugter Daten bestimmte statistische Verfahren untersucht, welche in diesem Fall unterschiedliche Clusterverfahren sind. Dazu werden in Kapitel 2 Datensätze erzeugt, die jeweils aus verschiedenen Gruppen bestehen. Anschließend werden in Kapitel 3 fehlende Werte in diesen Datensätzen erzeugt und allgemein geschildert, was genau unter fehlenden Daten verstanden wird. In Kapitel 4 werden verschiedene Imputationsverfahren beschrieben und daraufhin werden fehlende Daten mithilfe der multiplen Imputation imputiert. In Kapitel 5 werden die einzelnen Clusterverfahren vorgestellt, die untersucht werden. Schließlich werden die Ergebnisse der einzelnen Clusterverfahren vor und nach der Imputation verglichen und in Kapitel 6 dargestellt. Hier wird die Auswirkung der Imputation auf die einzelnen Verfahren untersucht.

Ziel der Arbeit ist es nun herauszufinden, ob sich manche Clusterverfahren im Falle imputierter Datensätze besser oder schlechter für eine Clusteranalyse eignen als andere. Zudem soll ermittelt werden, inwiefern es überhaupt sinnvoll ist, eine Clusteranalyse auf Basis imputierter Datensätze durchzuführen.

2 Erzeugung der Datensätze

In diesem Kapitel werden die einzelnen Datensätze beschrieben, auf denen die Simulationsstudie basiert. In diesen Datensätzen werden später fehlende Werte erzeugt und wieder imputiert, um anschließend das Verhalten verschiedener Clusterverfahren vor und nach Imputation fehlender Daten analysieren zu können. Die erzeugten Datensätze unterscheiden sich in ihrer Verteilung, in den Korrelationen zwischen den einzelnen Variablen, in den einzelnen Gruppengrößen sowie in der Anzahl an Gruppen(Cluster). Außerdem wird gezeigt, wie diese mithilfe des Programms R erzeugt werden. Es werden insgesamt 5 Datensätze erzeugt.

2.1 Verteilungen

Im folgenden Abschnitt wird kurz auf die Theorie der Verteilungen eingegangen, auf denen die erzeugten Datensätze basieren.

2.1.1 Die multivariate Normalverteilung

Die Datensätze 1 bis 3 sind allesamt multivariat normalverteilt. Die multivariate Normalverteilung gilt als eine der wichtigsten multivariaten Verteilungen und ist eine häufig getroffene Annahme für die Analyse multivariat verteilter Daten. Wie sich schon aus dem Namen ableiten lässt, ist sie die multivariate Form der univariaten Normalverteilung.
Eine univariat normalverteilte Zufallsvariable X besitzt die Dichte

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad x, \mu \in \mathbb{R}, \;\; \sigma^2 > 0$$

wobei μ den Erwartungswert und σ^2 die Varianz darstellt.Wenn X normalverteilt ist, wird dafür kurz $X \sim N(\mu, \sigma^2)$ geschrieben. Es wird von einer multivariaten Normalverteilung gesprochen, wenn alle Linearkombination der Komponenten der Verteilung univariate Normalverteilungen sind. Eine multivariate (bzw. n-variate) normalverteilte Zufallsvariable $\mathbb{X} = (X_1, X_2, ..., X_n)^T$ besitzt die Dichte

$$f(x) = f(x_1, ..., x_n) = \frac{1}{(2\pi)^{\frac{n}{2}}(\det(\mathbf{\Sigma}))^{\frac{1}{2}}} \exp(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu))$$

wobei $x, \mu \in \mathbb{R}^n$ und $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$.$\mu$ stellt wieder den Erwartungswert dar.Die positiv semi-definite Kovarianzmatrix wird mit $\mathbf{\Sigma}$ bezeichnet. Die Kovarianzmatrix enthält in der Diagonalen die Varianzen der Komponenten. Die restlichen Elemente der Matrix beinhalten die paarweisen Kovarianzen,die jeweils den Zusammenhang zwischen zwei Variablen ausdrücken. Sind die Komponenten von $\mathbb{X}$, also $X_1, X_2, ..., X_n$ unabhängig und identisch $X \sim N(\mu, \sigma^2)$

verteilte Zufallsvariablen, dann hat die Kovarianzmatrix die Form

$$\mathbf{\Sigma} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & 1 \end{pmatrix} \quad bzw. \ \mathbf{\Sigma} = \mathbb{1}_n$$

Zwischen den einzelnen Komponenten besteht in diesem Fall kein Zusammenhang, somit ist die Kovarianz in allen Fällen gleich 0. In der Realität kommt es allerdings häufig vor, dass die einzelnen Komponenten von $\mathbb{X}$ nicht komplett unabhängig voneinander sind, sondern ein Zusammenhang zwischen ein oder mehreren Komponenten besteht. Im allgemeinen Fall wird die Kovarianzmatrix wie folgt geschrieben:

$$\mathbf{\Sigma} = \begin{pmatrix} var(X_1) & cov(X_1, X_2) & \cdots & cov(X_1, X_n) \\ cov(X_2, X_1) & var(X_2) & \cdots & cov(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(X_n, X_1) & cov(X_n, X_2) & \cdots & var(X_n) \end{pmatrix}$$

Ist eine n-variate Zufallsvariable $\mathbb{X}$ normalverteilt,so schreibt man kurz $\mathbb{X} \sim N_n(\mu, \mathbf{\Sigma})$. Ein n-variater normalverteilter Zufallsvektor besteht aus n univariat normalverteilten Zufallsvariablen. Gilt also $\mathbb{X} = (X_1, X_2, ..., X_n)^T \sim N_n(\mu, \mathbf{\Sigma})$, ist jede Komponente X_i von $\mathbb{X}$ univariat normalverteilt. Die Informationen für diesen Abschnitt wurden aus [Rahnenfuehrer 2008] und [Fahrmeir et al. 1996] entnommen.

2.1.2 Die Dirichlet Verteilung

Die Dirichlet Verteilung ist eines der grundlegenden Modelle zur Modellierung von proportionalen Daten, wie beispielsweise die Mischung von verschiedenen Vokabeln in einem Text. Sie ist die multivariate Form der Betaverteilung. Die Dichte der Betaverteilung lässt sich darstellen mit

$$f(x) = \frac{1}{B(a_1, a_2)} x^{a_1 - 1}(1 - x)^{a_2 - 1}$$

wobei

$$B(a_1, a_2) = \frac{\Gamma(a_1)\,\Gamma(a_2)}{\Gamma(a_1 + a_2)} \quad wobei \ \ \Gamma(x) = \int_0^\infty s^{x-1} e^{-s} ds$$

die Beta Funktion ist. a_1 und a_2 sind die sogenannten shape Parameter, für die $a_1, a_2 > 0$ gilt. Die Betaverteilung ist der konjugierte Prior zur Binomialverteilung und wird häufig verwendet, um die Unsicherheit über eine

binominale Wahrscheinlichkeit zu beschreiben (mit der gegebenen Anzahl an Versuchen und Erfolgen). Beispielsweise besitzt die Bank eine bestimmte Anzahl an Kreditnehmern eines bestimmten Typs, die ihre Kredite nicht zurückzahlen konnten (s) und die Anzahl aller Kreditnehmer(n) dieses bestimmten Typs. Dann wird die Wahrscheinlichkeit, dass ein Kreditnehmer dieses Typs ausfällt, mit $Beta(s+1, n-s+1)$ geschätzt. Ist die Zufallsvariable X betaverteilt, schreibt man kurz $X \sim Beta(a_1, a_2)$. Die Dirichlet Verteilung hingegen ist der konjugierte Prior zur multivariaten Binominalverteilung, der sogenannten Multinominalverteilung. Somit ist die Dirichlet die multivariate Form der Beta Verteilung. Die Dichte der Drichtlet Verteilung ist gegeben mit

$$f(x_1, x_2, ..., x_n; a_1, a_2, ..., a_n) = \frac{\Gamma(\sum_{i-1}^{n} a_i)}{\prod_{i-1}^{n} \Gamma(a_i)} \prod_{i-1}^{n} x_1^{a_i - 1}$$

Da wie oben beschrieben die Dirichlet Verteilung für die Modellierung proportionaler Daten benutzt wird, gilt $\sum_{i-1}^{n} x_i = 1$ und $0 < x_i < 1$. Einfach ausgedrückt, da es sich um Wahrscheinlichkeiten handelt, liegt der Wertebereich zwischen 0 und 1. Kurz schreibt man $\mathbb{X} \sim Dir(a_1, a_2, ..., a_n)$, wobei die jeweiligen Randverteilungen $X_i \sim Beta(a_i, \sum_{i=1}^{n} a_i \ - a_i)$ verteilt sind. Die Informationen dieses Abschnitts entstammen aus [Hortensius 2012], [Minka 2012] und [Unbekannt 2007].

2.2 Umsetzung in R

Um mithilfe von R multivariat normalverteilte Zufallsvariablen erzeugen zu können, steht die Funktion *mvrnorm* aus dem *MASS* Paket zur Verfügung. Dazu muss ein Vektor mit den Erwartungswerten der Komponenten, die Stichprobengröße n sowie die Kovarianzmatrix $\mathbf{\Sigma}$ übergeben werden. Bei der Erzeugung der Datensätze sollen neben den einzelnen Erwartungswerten und Varianzen auch der Zusammenhang der einzelnen Komponenten bestimmt werden. Dies gestaltet sich allerdings schwierig, wenn die Kovarianzmatrix übergeben werden muss, da die Kovarianz zwar ein Maß für den Zusammenhang darstellt, allerdings maßstabsabhängig ist. Das hat zur Folge, dass wenn statt X der Zehnfache Wert 10 X betrachtet wird, die Kovarianz 10 Cov (X,Y) beträgt. Somit ist aufgrund der Skalenwahl die Kovarianzmatrix als Absolutzahl schwierig interpretierbar. Aus diesem Grund wird die Korrelation bzw. der Korrelationskoeffizient als Alternative verwendet. Die Korrelation ist bestimmt durch

$$Cor(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

Mit Verwendung der Korrelation anstatt der Kovarianz löst sich das Problem der Maßstabsabhängigkeit. Dadurch das der Wertebereich der Korrelation zwischen -1 und 1 liegt, lässt sich der Zusammenhang auch leichter

interpretieren (vgl. [Fairmeir et. al 2011]). Da der *mvrnorm* Funktion allerdings eine Kovarianzmatrix übergeben werden muss, muss eine Funktion erstellt werden, die aus vorgegebenen Korrelationen (und Varianzen) eine Kovarianzmatrix erstellt. Zusätzlich ist es erforderlich, dass die Kovarianzmatrix positiv semi-definit ist. Falls mit den gegebenen Korrelationen und Varianzen keine positv semi-definite Matrix erstellt werden kann, wird mithilfe der Funktion *NearPD* aus den *MASS* Paket diejenige Kovarianzmatrix bestimmt, die den gegebenen Bedingungen am nächsten kommt. Dies ist allerdings nicht immer optimal, da die tatsächlichen Korrelationen im Datensatz dann sehr von den gewünschten Korrelationen abweichen können. Die Funktion, welche es erlaubt, Korrelationen, Erwartungswerte und Varianzen zu übergeben und daraus multivariat normalverteilte Daten zu erzeugen, ist im R-Code zu finden. Die Erzeugung dirichlet verteilter Daten gestaltet sich hingegen deutlich einfacher. Hierzu wird lediglich die *rdirichlet* Funktion aus dem *MCMCpack* Paket benötigt. Übergeben wird der Funktion neben der Stichprobengröße nur noch die shape Parameter. Da Datensätze erstellt werden sollen, die sich gut in homogene Gruppen(Cluster) einteilen lassen, müssen Daten erzeugen werden, die aus verschiedenen Verteilungen stammen. Mit der *MVN*(bwz.*rdirichlet*) Funktion erzeugen wir jeweils Daten aus einer Multinormalverteilung (bzw. Dirichlet Verteilung), die einen Cluster darstellen. Anschließend werden alle erzeugten Cluster zu einem Datensatz zusammengefasst.Dies enthält letztendlich Daten, die aus verschiedenen Gruppen zusammengesetzt sind. Diese Gruppen sollen später mit den Clusterverfahren gefunden werden.

Außerdem sollen sich in den erzeugten Datensätze auch kategoriale Variablen befinden. Dies wird durch die *cut* Funktion erreicht. Diese Funktion teilt ursprünglich metrische Daten in Intervalle ein und ordnet den Intervallen jeweils eine Kategorie zu. Mit dem Zusatz “as.ordered” wird erreicht,dass diese Variable ordinalskaliert ist. Je nach Datensatz wird die kategoriale Variable nominalskaliert oder ordinalskaliert sein.

Die gesamte Umsetzung zur Erzeugung der Datensätze ist im R-Code (im Anhang B.1) ab Seite 94 zu finden.

2.3 Datensätze

Hier werden kurz die Datensätze beschrieben, die erzeugt werden. Die Datensätze 1 bis 4 bestehen aus 3 verschiedenen Gruppen und besitzen jeweils 3 Variablen. Der fünfte Datensatz besteht aus 4 verschiedenen Gruppen. Die einzelnen Gruppen der ersten 3 Datensätze sind multinormal verteilt, die der Datensätze 4 und 5 dirichlet verteilt. Die letzte Variable stellt die kategoriale Variable der einzelnen Datensätze dar. Die Stichprobengröße ist in allen Datensätzen mit insgesamt 5000 Beobachtungen gleich. Die genauen Werte der

Korrelationen, Erwartungswerte, Varianzen, shape Parameter und Gruppengrößen sind im R-Code(ab Seite 94)zu finden. Obwohl der Datensatz dreidimensional ist, wird aufgrund der Anschaulichkeit für jeden Datensatz ein (vereinfachter) zweidimensionaler Plot erstellt, um die verschiedenen Gruppen in den Datensätzen darzustellen. Diese Gruppen werden unterschiedlich farblich markiert.

Datensatz 1

Im Datensatz 1 wurden Daten erzeugt, die aus 3 unterschiedlichen, aber gleich großen Gruppen stammen und 3 Variablen besitzen. Die Korrelationen zwischen den Variablen sind in jeder Gruppe gleich und positiv. Das heißt, falls eine Variable groß ist, sind auch die anderen beiden tendenziell größer. Die Erwartungswerte sind in jeder Gruppe unterschiedlich, die Varianzen dagegen gleich. Die dritte Variable des Datensatzes ist ordinalskaliert mit 3 verschiedenen Ausprägungen. Insgesamt wurde der Datensatz so erstellt, das die Cluster gut voneinander getrennt sind und dadurch auch durch die verschiedenen Verfahren gefunden werden sollten. Der Plot zum Datensatz 1 ist in Abbildung 1 dargestellt.

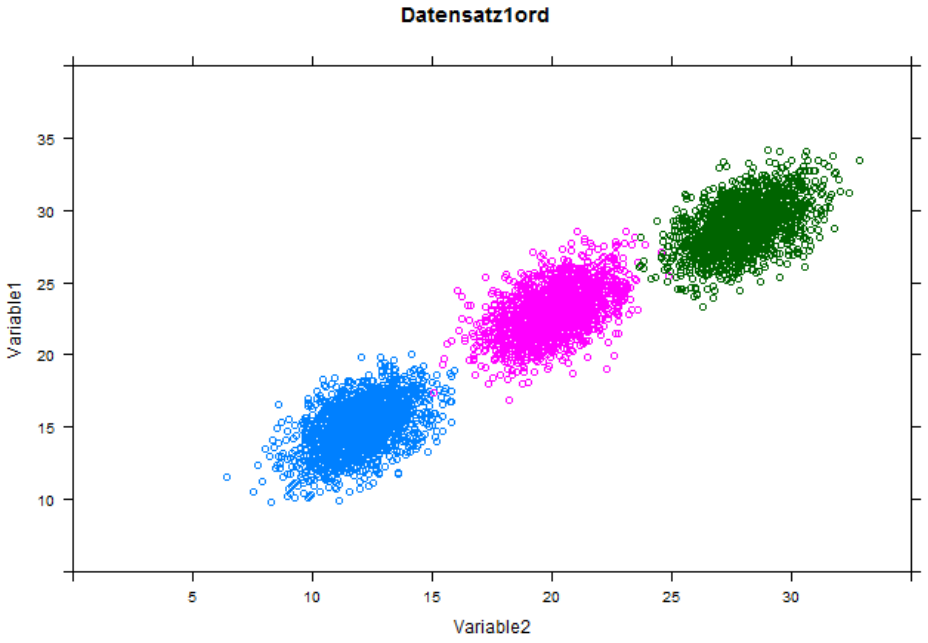

Abbildung 1: Datensatz 1

Datensatz 2

Im Datensatz 2 sind nur noch die Korrelationen zur 3. Variable in jedem Cluster gleich. Diese sind jeweils mit der 1. Variable positiv und mit der 2. Variablen negativ korreliert. Die Gruppen sind nun unterschiedlich groß. Zusätzlich enthält dieser Datensatz im Vergleich zum ersten Datensatz deutlich mehr Überschneidungen bei den einzelnen Gruppen, sodass die Cluster insgesamt schwieriger gefunden werden sollten. Dies wird auch im Plot (siehe Abbildung.2) erkenntlich. Die kategoriale Variable ist wieder ordinalskaliert.

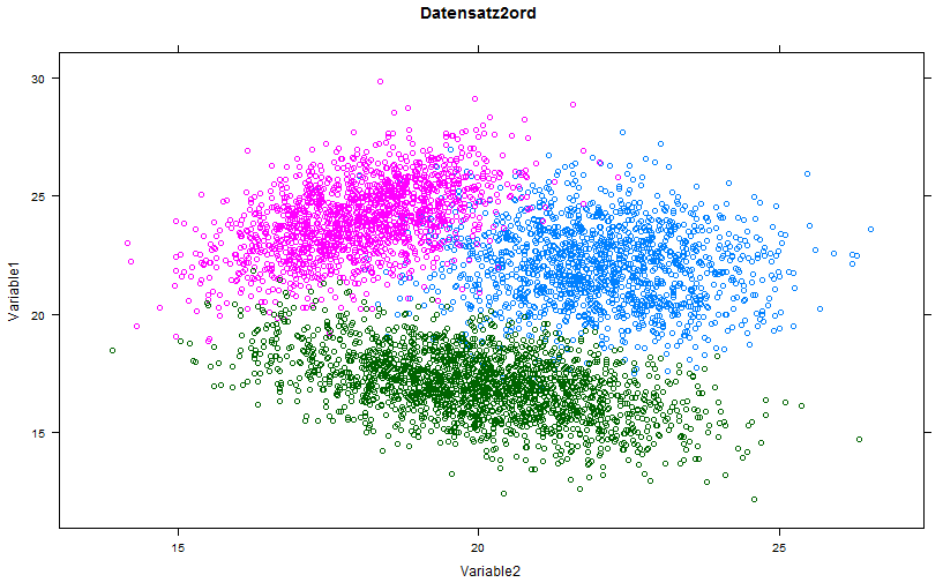

Abbildung 2: Datensatz 2

Datensatz 3

In Datensatz 3 sind nun alle Korrelationen und Varianzen in den einzelnen Gruppen unterschiedlich. Außerdem existiert eine größere Gruppe und zwei Kleinere. Die Erwartungswerte einzelner Variablen in verschieden Gruppen sind teilweise ähnlich. Dies könnte sich sowohl in der Clusteranalyse als auch in der Imputation auswirken. Der Plot ist in Abbildung 3 abgebildet. Die kategoriale Variable ist zudem nominalskaliert. Auch hier lassen sich wieder zahlreiche Überschneidungen erkennen.

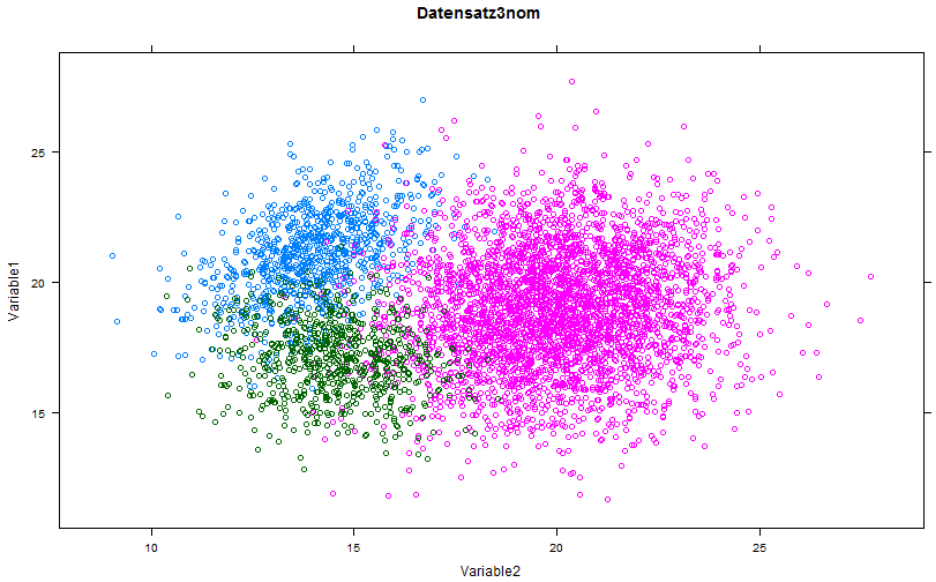

Abbildung 3: Datensatz 3

Datensatz 4

Datensatz 4 ist nun dirichlet verteilt. Er besteht wie die Datensätze 2 und 3 aus drei verschieden großen Gruppen. Auch gibt es wieder zahlreiche Überschneidungen der einzelnen Gruppen (siehe Abbildung 4). Zudem ist die kategoriale Variable ordinalskaliert.

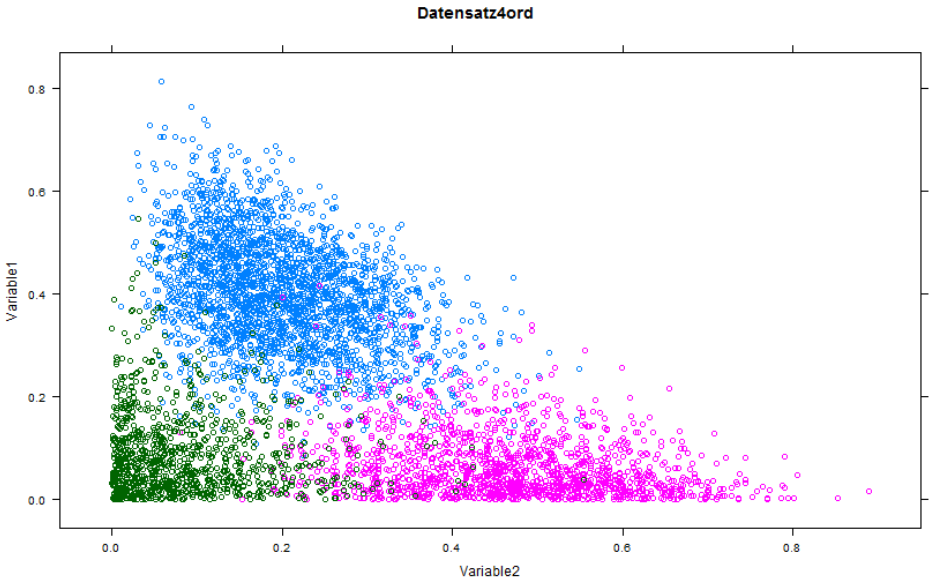

Abbildung 4: Datensatz 4

Datensatz 5

Datensatz 5 ist wie Datensatz 4 Dirichlet verteilt. Er besteht aus 4 gleich großen Gruppen. Die kategoriale Variable ist nominalskaliert. In dem zugehörigen Plot (siehe Abbildung 5) sieht man, dass die vierte Gruppe von den anderen drei umschlossen wird.

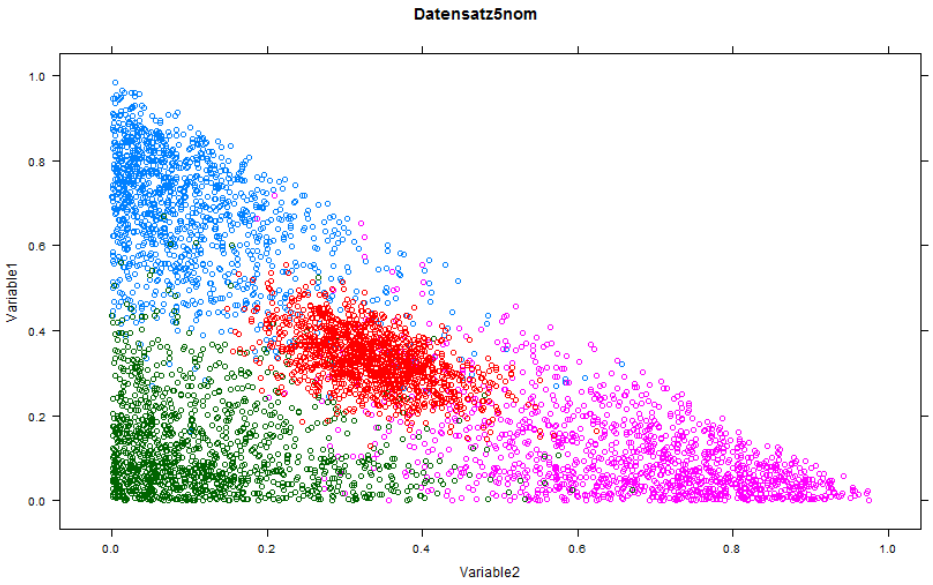

Abbildung 5: Datensatz 5

3 Fehlende Daten

3.1 Definition

Bei empirischen Arbeiten mit Datensätzen tritt häufig das Problem unvollständiger Datensätze auf. Das bedeutet, dass bei bestimmten Merkmalsträger Beobachtungen fehlen. Dies tritt beispielsweise oftmals in der Sozial- und Wirtschaftsforschung, in der Psychologie oder Epidemiologie auf, wo Datensätze oft auf Grundlage freiwilliger Beteiligung entstehen.Fehlende Beobachtungen stellen ein grundsätzliches Problem bei der Analyse von Datensätzen dar. Wie sollen Merkmalsträger behandelt werden, bei denen einzelne Beobachtungen fehlen? Können diese Merkmalsträger überhaupt berücksichtigt werden? Dies sind nur zwei von vielen Fragen, die fehlende Daten in Datensätzen aufwerfen. Bevor jedoch eine Antwort auf die Frage geliefert werden, wie mit dem Problem fehlender Daten umgegangen werden kann, muss genau definiert werden, was unter dem Begriff "Fehlende Daten"verstanden wird. In [Spiess 2008] werden fehlende Daten wie folgt klassifiziert:

> *Werte werden dann als fehlend bezeichnet, wenn als existierend angenommene und als Reaktion auf einen Reiz hervorrufbare Werte, deren Beobachtungen intendiert ist, als Reaktion auf die Reizvorgabe nicht beobachtet werden und auch nicht ohne Unsicherheit aus anderen Informationen ableitbar sind.*

Werte, die sich nicht aus anderen Angaben ableiten lassen, werden demnach nicht als fehlende Werte bezeichnet. Beispielsweise gilt das Alter einer Versuchsperson nicht als fehlend, wenn das Geburtsjahr der entsprechenden Person bekannt ist. Obige Definition schließt allerdings auch Fälle ein, die nicht erhoben wurden, was in der Fachsprache als "missing by design" bezeichnet wird. In dieser Betrachtung wird hingegen davon ausgegangen, dass die Werte unbeabsichtigt fehlend und nicht aufgrund dessen, dass sie schlichtweg nicht erhoben worden sind.

3.2 Klassifikation fehlender Werte

Daten können aus den unterschiedlichsten Gründen fehlen. Beispielsweise können Probanden beim Ausfüllen eines Fragebogens bestimmte Fragen übersehen, die Beobachtungen würden dann rein zufällig fehlen. Verweigern mehrere Probanden allerdings bestimmte Angaben (bsp. die Angabe des Einkommens), so fehlen diese Beobachtungen systematisch. Solche systematisch fehlende Beobachtungen können immensen Einfluss auf das Ergebnis einer Analyse haben. Fehlen beispielsweise vermehrt Angaben von Personen mit höheren Einkommen, so würde das mittlere Einkommen aller befragten Personen systematisch unterschätzt werden. Aus diesem Grund ist es bei der Behandlung fehlender Werte von großer Bedeutung herauszufinden, weshalb

bestimmte Beobachtungen fehlen. Der Mechanismus, der beschreibt wie die fehlenden Werten zustande gekommen sind, wird als Missingmechanismus bezeichnet. Die Missingmechanismen lassen sich in verschiedene Arten gliedern:

- **Missing completely at Random** Ist die Wahrscheinlichkeit für beobachtete Muster an fehlenden und beobachteten Werten unabhängig von den Variablen, deren Werte beobachtet wurden und zugleich aber auch von den Variablen, deren Werte nicht beobachtet worden sind, so spricht man von "completely at random"(MCAR). Die Werte fehlen also komplett zufällig und frei von jeglichen Einflüssen. Das zufällige Löschen von Beobachtungen in einem Datensatz wäre ein Beispiel für Werte, die MCAR sind. Die Antwortrate ist also unabhängig von der jeweiligen Ausprägung selbst und der Ausprägung anderer Merkmale. Mathematisch ausgedrückt sind fehlende Daten MCAR, wenn

  $$f(R|D) = f(R) \ \forall D$$

  Die Verteilung von R ist also komplett frei von jeglichen Einflüssen der Variablen in D. R stellt dabei die Wahrscheinlichkeit für das Fehlen einer Beobachtung dar. Die beobachteten Werte einer Variablen könnten nach dem Entfernen von Werten als Zufallsstichprobe der ursprünglich kompletten Stichprobe angesehen werden. Dies wird bei Betrachtung der beiden Randverteilungen von Variable 1 und Variable 2 vor und nach dem Löschen unter der MCAR-Bedingung (siehe Abb. 6) deutlich. Das ist auch der Grund, das MCAR fehlende Werte den unproblematischsten Missingmechanismus darstellen.

- **Missing at Random** Ist die Wahrscheinlichkeit für das beobachtete Muster an fehlenden und beobachten Werten im Gegensatz zum MCAR-Mechanismus abhängig von beobachteten Variablenwerten, dann wird von "Missing at random"(MAR) gesprochen. Wird beispielsweise nach dem Einkommen einer Person gefragt und die Ausfallwahrscheinlichkeit hängt dabei von dem Alter der befragten Person ab, gelten die fehlenden Daten als MAR. Mathematisch ausgedrückt sind fehlende Daten MAR, wenn

  $$f(R|D) = f(R|D_{obs}) \ \forall D$$

  Die Verteilung von R hängt also nur von den beobachteten Variablen D_{obs} aus D ab. Der Unterschied zum MCAR Mechanismus besteht darin, dass beim MAR Mechanismus die Ausfallwahrscheinlichkeit von den Ausprägungen anderer Merkmale, wie beispielsweise dem Alter der Person, abhängen kann, die Ausfallwahrscheinlichkeit also nicht komplett unabhängig von anderen Variablen ist.Von MAR kann allerdings nicht mehr gesprochen werden, falls Person mit höherem Einkommen

die Antwort eher verweigern als Personen mit niedrigen Einkommen. Die Antwortrate bleibt beim MAR Mechanismus weiterhin unabhängig von der Ausprägung selbst. Die bisher beschriebenen Missingmechanismen gelten als zufällige Ausfallmechanismen.

- **Not Missing at Random** Wenn die fehlenden Daten weder als MAR noch als MCAR klassifiziert werden können, sind die Daten "Not Missing at Random" (NMAR), sie fehlen also nicht zufällig. Die Antwortrate hängt also u.a von der Ausprägung selbst ab. Fehlen beispielsweise häufig Angaben des Einkommens von Personen mit hohen Einkommen, so spricht man von NMAR. In diesem Fall kann die bedingte Verteilung $f(R|D)$ nicht mehr vereinfacht werden.

Während zufällige Ausfallmechanismen das Ergebnis der Analyse normalerweise kaum verzerren, kann eine Nichtberücksichtigung systematischer Missingmechanismen zu fehlerhaften Ergebnissen führen. Die Schätzung des durchschnittlichen Alters in einer Stichprobe würde beispielsweise zu niedrig eingeschätzt werden, wenn überwiegend ältere Personen Angaben verweigern. Anderseits kann der Missingmechanismus nur berücksichtigt werden, wenn er bekannt ist. Ein wichtiger Unterschied des NMAR Missingmechanismuses im Gegensatz zu den beiden vorherigen besteht darin, dass dieser bei Verfahren auf Basis der Likelihoodfunktion nicht ignoriert werden kann und deshalb berücksichtigt werden müsste. Zur Veranschaulichung der Unterschiede der Missingmechanismen werden im Datensatz 3 fehlende Daten mit den verschiedenen Missingmechanismen erzeugt. Anschließend werden die Randverteilungen der Variablen 1 und 2 dargestellt, einmal die des kompletten Datensatzes und einmal die des Datensatzes mit fehlenden Werten. Dabei werden bei dem Datensatz mit den erzeugten fehlenden Werten nur die Merkmalsträger berücksichtigt, die keine fehlenden Werte aufweisen.Die Randverteilungen des kompletten Datensatzes sind mit schwarzen, durchgezogenen Linien dargestellt, die Randverteilungen des Datensatzes mit fehlenden Werten hingegen mit roten gestrichelten Linien.

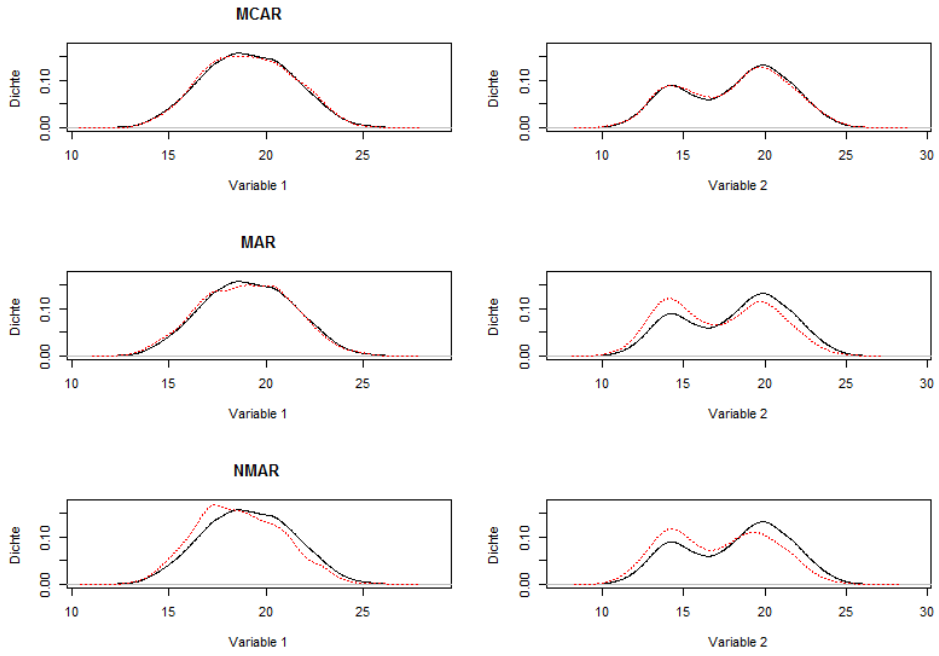

Abbildung 6: Vergleich der Missingmechanismen bei Datensatz 3

Bei den Randverteilungen vor und nach dem Löschen von Daten mit dem MCAR Mechanismus sind keine großen Unterschiede zu erkennen. Betrachtet man hingegen die Randverteilungen des MAR-Mechanismus und die des NMAR-Mechanismuses (Abb.MCAR), so lässt sich feststellen, dass bei beiden eine Analyse nur mit den kompletten Fällen nach dem Löschen zu verzerrten Ergebnissen führen würde. Sowohl die Randverteilungen des MAR- als auch die des NMAR-Mechanismuses unterscheiden sich nach dem Löschen deutlich von den jeweiligen ursprünglichen Randverteilungen. Aber in Anbetracht dessen, dass die fehlenden Werte später imputiert werden sollen, ist zusätzlich die Betrachtung der Regressionsbeziehung, beispielsweise von Variable 2 auf Variable 1 nach und vor dem Löschen, von großer Bedeutung. In Abbildung 7 ist Datensatz 3 (ohne farbliche Darstellung der Cluster) mit eingezeichneten Regressionsgeraden dargestellt. Dort wird auch sichtbar, dass die Regressionsgeraden der MCAR und MAR Mechanismen relativ ähnlich wie die ursprüngliche Regressiongerade der kompletten Daten verlaufen, während die Regressionsgerade der NMAR deutlich von der Ursprungsregressiongerade abweicht. Beim Löschen fehlender Daten mit dem MAR Mechanismus bleibt die Regressionsbeziehung im Gegensatz zum NMAR Mechanismus also relativ unverzerrt erhalten. Dies ist ein entscheidender Unterschied, denn das Wissen über den Erhalt der Regressionsbeziehung nach dem Löschen wird bei der multiplen Imputation ausgenutzt.

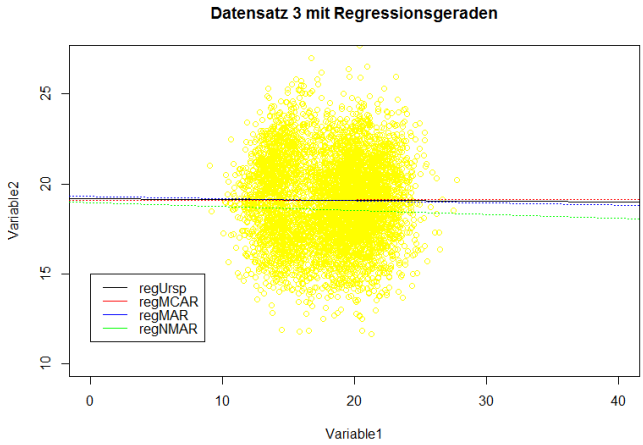

Abbildung 7: Datensatz 3 mit Regressionsgeraden

Beim Löschen mit dem NMAR Mechanismus ist eine Kompensation fehlender Werte auf Basis der Stichprobe nicht mehr möglich, da hier auch die Regressionsbeziehung verzerrt ist.

In der Realität fällt es häufig schwer, den entsprechenden Missingmechanismus zu finden, deshalb ist auch die Entscheidung, ob und wie er berücksichtigt werden soll, ebenfalls schwierig. Zudem kann es vorkommen, dass der Missingmechanismus nicht über alle Beobachtungen hinweg der selbe ist.

3.3 Erzeugung fehlender Daten in R

Nun sollen in den kompletten Datensätzen 1 bis 5 fehlende Werte erzeugt werden. Diese fehlenden Werte wollen wir später imputieren. Da wie im vorherigen Kapitel beschrieben eine Kompensation NMAR fehlender Werte nur anhand der Stichprobe nicht mehr möglich ist, sollten die fehlenden Daten nicht mit dem NMAR Mechanismus erzeugt werden. Zudem sollte das Missingmuster ignorierbar sein, was im Falle NMAR fehlender Daten nicht gegeben wäre.
Aus diesem Grund werden fehlenden Daten mit dem MAR Mechanismus erzeugt. Das Fehlen einer Beobachtung in den erzeugten Datensätzen hängt also von den Ausprägung der anderen Variablen ab, nicht aber von der Ausprägung der Variable selbst. Dies wurde so umgesetzt, dass die Wahrscheinlichkeit für das Fehlen einer Beobachtung in einer Variable umso höher ist, je höher die Ausprägungen der Werte der anderen beiden Variablen sind[1].

[1] Alternativ hätte man bsp. auch festlegen können, dass die Wahrscheinlichkeit für das Fehlen einer Beobachtung umso kleiner ist, je größer die Werte der anderen beiden

Sind beispielsweise die Ausprägungen der Variable 2 und 3 groß, so ist die Wahrscheinlichkeit für ein Fehlen der Beobachtung in Variable 1 ebenfalls groß.
Um mit R eine Erzeugung fehlender Daten nach dem MAR Mechanismus umzusetzen, wird zuerst mithilfe der logistischen Funktion $f(x) = \frac{e^{\eta}}{1+e^{\eta}}$ (bzw. $f(x) = \frac{1}{1+e^{-\eta}}$) eine Wahrscheinlichkeit für das Fehlen einer Beobachtung in einer Variable festgelegt. Das η legt dabei die Beziehung mit den anderen beiden Variablen fest. Beispielsweise ist $\eta = -3.0+0.1*x[2]+0.5*x[3]$, man erhält eine Zahl, die von den anderen beiden Variablen abhängt (In diesem Fall würde die Wahrscheinlichkeit für das Fehlen in Variable 1 berechnet werden, dass von den Variablen 2 und 3 abhängt). Die logistische Funktion sorgt anschließend dafür, dass der Wertebereich zwischen 0 und 1 ist, sodass wir eine Wahrscheinlichkeit erhalten. Je größer also Variable 2 und Variable 3 sind, desto größer ist auch η und somit auch die Wahrscheinlichkeit für ein Fehlen der Beobachtung in Variable 1. Analog werden so Wahrscheinlichkeiten für das Fehlen von Beobachtungen der anderen Variablen festgelegt. Für Datensatz 4 und 5 wird η teilweise direkt verwendet, also keine logistische Funktion zur Hilfe genommen, da hier bei passendem η die Werte bereits zwischen 0 und 1 liegen.

Somit sind die Wahrscheinlichkeiten für das Fehlen einer Beobachtung in jeder Variablen gegeben. Anschließend wird für jede Variable (mithilfe der Binomialverteilung) 5000 mal (entspricht der Größe der Datensätze) eine "Münze" mit der jeweiligen Wahrscheinlichkeit "geworfen". Daraus ergeben sich Einsen (Kopf) und Nullen (Zahl). Die Eins steht dabei für die Möglichkeit, dass dieser Wert fehlen kann, die Null hingegen dafür, dass der Wert nicht fehlen kann. Für jede der drei Variablen erhalten wir also 5000 Daten mit Nullen und Einsen, wobei auch die jeweilige Zeile gespeichert ist. Dann werden jeweils nur die Fälle ausgewählt, welche die Ausprägung 1 haben (für die "Kopf" geworfen wurde), und die jeweiligen Zeilen in einem Vektor gespeichert. So entstehen 3 Vektoren mit Zeilen der Variablen, die potenziell ausfallen könnten. Vektor 1 enthält die Zeilen von Variable 1, Vektor 2 die Zeilen von Variablen 2, und Vektor 3 die Zeilen von Variable 3. Bevor mit dem Löschen von Beobachtungen begonnen wird, wird die Anzahl festgelegt, wie viele Beobachtungen gelöscht werden sollen. Insgesamt werden bei den Datensätzen 30 % der Beobachtungen entfernt. Wie viele Beobachtungen in den jeweiligen Variablen fehlen, wird zufällig festgelegt. Es wird so beispielsweise zufällig ermittelt , dass 1000 Beobachtungen aus Variable 1 entfernt werden sollen. Anschließend werden aus dem Vektor der Variable 1 zufällig 1000 Zeilen ausgewählt, dessen Beobachtungen in Varia-

Variablen sind. Es gäbe mehre Möglichkeiten, MAR fehlende Daten zu erzeugen, es macht allerdings keinen Unterschied, solange die Wahrscheinlichkeit für das Fehlen nicht von der Variable der fehlenden Beobachtung selbst abhängt.

ble 1 gelöscht werden. Genauso läuft der Vorgang des Löschens bei Variable 2 ab. Um die Löschung kompletter Zeilen zu vermeiden, werden bevor Beobachtungen aus Variable 3 entfernt werden, die gelöschten Zeilen von Variable 1 und Variable 2 verglichen. Zeilen, die bei beiden gelöscht wurden, fallen auch aus dem Vektor für Variable 3 heraus, um die Löschung eines kompletten Merkmalsträgers zu vermeiden. Es können allerdings 2 Beobachtungen eines Merkmalträgers fehlen. Schließlich werden auch bei Variable 3 zufällig Zeilen aus Vektor 3 ausgewählt, die gelöscht werden.

Nun soll kurz die Auswirkung der NA (NA steht für fehlenden Wert) Erzeugung gezeigt werden. Bei Datensatz 1 ist die Gruppengröße überall gleich groß. Betrachtet die Gruppengrößen der kompletten Fälle nach dem Löschen mit dem MCAR Mechanismus, so sind die Gruppen wie der ursprüngliche Datensatz etwa gleich groß(grafisch dargestellt in Abb. 8). Werden die Werte auf diese Art und Weise entfernt, sind die Gruppengrößen nicht mehr gleich groß (vgl. Abb. 9), Gruppe 1 ist nach dem Löschen deutlich größer wie die anderen beiden, wobei Gruppe 3 nochmals kleiner als Gruppe 2 ist. Es kann also davon ausgegangen werden, dass die obig beschriebene Vorgehensweise die Daten nicht komplett zufällig (MCAR) entfernt. Nach dem Löschen der Werte erhalten wir einen Datensatz mit den fehlenden Werten nach dem MAR Mechanismus, da das Fehlen nur von den beiden anderen Variablen abhängt.

Aufgrund der Tatsache, dass verschiedene Clusterverfahren nicht mit der gemischt skalierten Datensätzen umgehen können, entstehen für jeden Datensatz zwei unterschiedliche Datensätze mit fehlenden Werten,die jeweils als Datensatz NA1 und Datensatz NA2 bezeichnet werden. Datensatz NA1 enthält nur die zwei metrischen Variablen mit fehlenden Werten (Die zugehörigen Plots bleiben gleich). Die fehlenden Daten werden hier also leicht abgeändert erzeugt (mit 2 statt 3 Variablen). Datensatz NA2 hingegen ist der Datensatz mit der kategorialen Variable, die Erzeugung der fehlenden Daten verläuft wie beschrieben. Die genaue Umsetzung ist im R-Code ab Seite 98 bzw. Seite 115 (für Datensatz 4 und 5) zu finden.

Die Informationen dieses Kapitels stammen aus [Spiess 2008], [Igl 2004], [Lintorf 2011], [Toutenburg et al. 2002] und [Deng 2012].

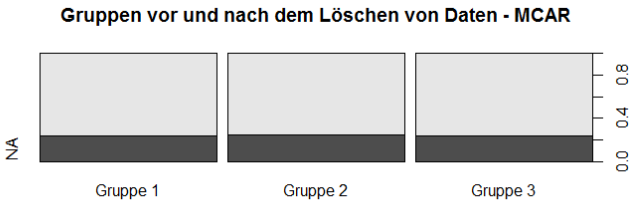

Abbildung 8: Gruppengrößen vor und nach dem Löschen von Daten - MCAR. NA bezeichnet den Datensatz mit fehlenden Werten.

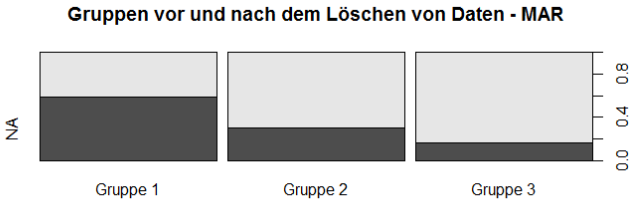

Abbildung 9: Gruppengrößen vor und nach dem Löschen von Daten - MAR

4 Imputation fehlender Werte

Das konkrete Problem, dass fehlende Werte in Anbetracht der Anwendung von Clusterverfahren implizieren, ist dass viele Clusterverfahren nicht mit fehlenden Werten umgehen können. Es könnten lediglich die kompletten Fälle betrachten werden, also nur Merkmalsträger ohne fehlende Werte könnten einer Gruppe zugewiesen werden. Da allerdings jedem Merkmalsträger eine Gruppe zugeordnet werden soll, muss jede fehlende Beobachtung durch einen Alternativwert ersetzt werden. Hierfür verwendet man verschiedene Imputationsverfahren, die fehlende Werte durch plausible Alternativwerte ersetzen. Durch diese Methoden erhält man wieder komplette Datensätze, auf denen die Clusterverfahren problemlos angewendet werden können und somit jedem Merkmalsträger eine Gruppe zugeordnet werden kann. Bei den Imputationsmethoden unterscheidet man nochmal in singuläre Imputationsverfahren, die genau einen Wert für eine fehlende Beobachtung einsetzen, und in die sogenannte multiple Imputation. Hier werden für eine fehlende Beobachtung mehrere Werte eingesetzt, dadurch entstehen auch mehrere imputierte Datensätze. In diesem Kapitel werden verschiedene Imputationsmethoden vorgestellt.

4.1 Singuläre Imputationsverfahren

Wie oben beschrieben, imputieren diese Verfahren exakt einen Wert für jede fehlende Beobachtungen. Im Folgenden werden einige dieser Imputationsverfahren mit kurzer Beschreibung aufgelistet:

Mean Imputation
Eine Möglichkeit wäre es, fehlende Werte mit dem arithmetischen Mittel der beobachteten Werte der jeweiligen Variablen zu ersetzen. Es wäre auch möglich, bedingte Mittelwerte zu imputieren, die je nach Ausprägung einer beobachteten Variable gebildet werden. Allerdings bringt diese Methode zahlreiche Nachteile mit sich. Beispielsweise können damit nur metrische Merkmale imputiert werden. Weit problematischer ist jedoch die Verzerrung der Datenverteilung durch die Mittelwertimputation, da Varianzschätzer inkonsistent geschätzt werden. Alternativ könnte auch der Modus oder Median anstatt des arithmetisches Mittels verwendet werden. Somit könnte man zwar auch nicht metrische Daten imputieren, andere Probleme sind aber weiterhin in ähnlicher Form vorhanden.

Regression Imputation
Diese Imputationsmethode ersetzt fehlende Werte durch vorhergesagte Werte einer Regressionsanalyse, die mit den beobachteten Werten durchgeführt wurde. Beispielsweise wird bei k Variablen, bei denen n-r Werte in der k-ten Variable fehlen, eine lineare Regressionsanalyse mit

den r kompletten Beobachtungen geschätzt. Mithilfe der Koeffizienten der geschätzten Regressionsfunktion

$$\bar{y}_{ik} = \tilde{\beta}_0 + \sum_{j=1}^{k-1} \tilde{\beta}_j y_{ij} \quad \forall i \in [r, n]$$

werden die fehlenden Werte in k-ten Variable imputiert. Für kategoriale Daten wird hier eine logistische Regression durchgeführt. Zusätzlich zur obigen Regressionsfunktion kann noch ein Residuum addiert werden(Stochastische Regression), um so die Unsicherheit des prognostizierten Wertes auszudrücken. Mit dieser Methode wird die Struktur innerhalb einer Variablen ausgenutzt. Dadurch bleibt zwar die Korrelationsstruktur erhalten, allerdings ist die Güte und Validität durch das Missingmuster beeinflusst.

Hot Deck Imputation
Die "Hot Deck" - Imputation ersetzt fehlende Werte durch Werte, die in der selben Stichprobe beobachtet wurden. Für jeden fehlenden Wert wird ein zufälliger Wert aus der beobachten Stichprobe eingesetzt. Das Problem ist, dass nur unter der Voraussetzung MCAR fehlender Daten allgemein unverzerrte Schätzer resultieren. Aus diesem Grund ist diese Methode nicht für die Datensätze 1 bis 5 geeignet.

Cold Deck Imputation
Die "Cold Deck" -Imputation verwendet im Gegensatz zur "Hot Deck" Imputation nicht Werte aus der selben Stichprobe, sondern Werte aus anderen Datensätzen oder Quellen. Es liegt auf der Hand, dass auch diese Methode nicht für die Datensätze verwendet werden kann, weil keine anderen Datensätze oder Quellen vorhanden sind.

Das Problem der singulären Imputationsmethoden ist es, den fehlenden Wert mit einem anderen Wert zu ersetzen und ihn anschließend als "wahren" Wert zu behandeln. Abgesehen von der stochastischen Regression wird dadurch bei singulären Imputationsmethoden die Unsicherheit nicht berücksichtigt. Außerdem wird die Varianz fast immer unterschätzt.

Entnommen wurden die Informationen für diesen Abschnitt aus [Spiess 2008], [Feilke 2009], [Rohrschneider 2007] und [Toutenburg et al. 2002].

4.2 Multiple Imputation

Anstatt wie bei den singulären Imputation den fehlenden Beobachtung nur einen neuen Wert zuzuordnen, ordnet die multiple Imputation den fehlenden Beobachtungen mehrere Werte zu. Dadurch entstehen auch für jeden Datensatz mehrere imputierte Datensätze. Die nicht fehlenden Beobachtungen des

ursprünglichen Datensatzes bleiben dabei in jeden der neuen Datensätze unverändert. Das Verfahren der multiplen Imputation lässt sich in drei Schritte aufteilen. Je nach Anzahl der unterschiedlichen Werte, die für einen fehlenden Wert imputieren sollen, ergibt sich im ersten Schritt eine gewisse Anzahl an Datensätzen. Will man beispielsweise für einen fehlenden Werte 5 verschiedene Werte imputieren, erhält man dementsprechend 5 unterschiedliche Datensätze. Diese werden im zweiten Schritt einzeln analysiert.Die Ergebnisse werden anschließend im dritten und letzten Schritt zu einem Ergebnis zusammengefasst (vgl. Abb. 10). Die verschiedenen imputierten Datensätze werden in dieser Arbeit als "Amelia Datensätze"bezeichnet.

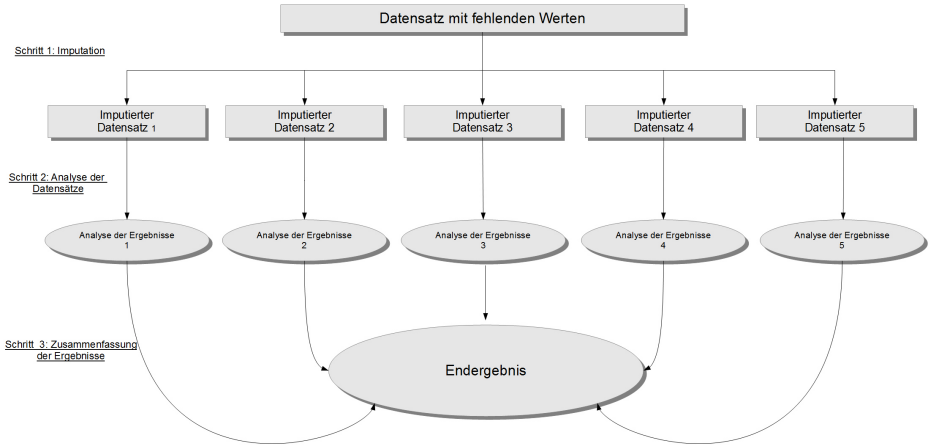

Abbildung 10: Schritte einer multiplen Imputation

Es gibt mehrere Gründe, weshalb die multiple Imputation gut geeignet ist. Aufgrund dessen, dass mehrere Werte imputiert werden, also mehrere Wiederholungen des Schätzprozesses stattfinden, werden die Standardfehler miteinbezogen. Dadurch wird im Gegensatz zu den meisten singulären Imputationmethoden die Unsicherheit berücksichtigt. Dies hat zur Folge, dass die Varianz nicht mehr unterschätzt wird. Zudem stellte sich in zahlreichen Publikationen die Methode der multiplen Imputation als zuverlässig heraus. So hat sich gezeigt, dass die Ergebnisse multipel imputierter Datensätze zuverlässiger als die Ergebnisse mit den "complete cases" sind. Zahlreiche Autoren halten die Multiple Imputation für das am besten geeignete Verfahren zum Umgang mit fehlenden Werten. Dies bestätigt sich auch in der Praxis, wie das Beispiel der "Survey of Consumer Finances" zeigt, dessen Analysten seit mehreren Jahren das Verfahren der multiplen Imputation zur Behandlung fehlender Werte anwenden. Die multiple Imputation ist zwar weit aufwändiger als eine singuläre Imputation, jedoch fällt dies im Zeitalter leistungsfähiger Computer kaum mehr ins Gewicht. Für die Art

und Weise, wie die Werte in Schritt 1 imputiert werden, gibt es zahlreiche Möglichkeiten. Zur praktischen Umsetzung wird das R-package *Amelia 2* verwendet. Aus diesem Grund wird die multiple Imputation so beschrieben, wie sie in Amelia 2 angewendet wird.

Die Informationen dieses Abschnitts stammen aus [Spiess 2008], [Feilke 2009], [Schaefer o.D] und [Masayoshi et al. 2012].

4.2.1 Voraussetzungen

Anzahl Imputationen

Bevor mit den Imputationen begonnen wird, muss festgelegt werden, wie viele Werte für einen fehlenden Wert imputiert werden sollen .Geht man von m Imputationen für einen fehlenden Wert aus, so zeigte Rubin,dass sich die Effizienz approximativ mit $(1 + \frac{\gamma}{m})^{-1}$ darstellen lässt (γ stellt die Rate der fehlenden Beobachtungen dar). Die Effizienzen mit verschiedenen m und γ sind in folgender Tabelle abgebildet:

m/γ	0.1	0.3	0.5	0.7	0.9
3	97	91	86	81	77
5	98	94	91	88	85
10	99	97	95	93	92
20	100	99	98	97	96

Tabelle 1: Effizienzen mit unterschiedlichen γ

In der Tabelle 1 kann man erkennen, dass bei einer niedrigen Fehlrate γ bereits mit wenigen Imputationen m eine hohe Effizienz erreicht werden kann. Aufgrund dessen, dass maximal 30 % der Werte entfernt werden, sind 5 Imputationen ausreichend, um eine hohe Effizienz zu erreichen. Die Tabelle sowie sonstige Informationen stammen aus [Schaefer o.D].

Annahmen

Um potenzielle Werte für die Imputation fehlender Werte zu finden, muss ein statistisches Modell für die kompletten Daten (beobachtete und fehlende Werte) festgelegt werden. Auf Grundlage dieses Modell werden anschließend die m Imputationen für die fehlenden Werte im Datensatz berechnet. Als Modell bewährte sich die Annahme einer multivariaten Normalverteilung, die sich - selbst im Vergleich mit komplizierten Modellen - als

ebenbürtig herausstellte. Es eignet sich also auch, obwohl in "echten Daten" nicht immer von multivariat normalverteilten Daten ausgegangen werden kann.Somit kann sich auch wie im Falle von Datensatz 4 und 5 auf drichtlet verteilte Daten angewendet werden.

Auch das Amelia 2 Paket geht davon aus, dass der (nxk) Datensatz D, bestehend aus den beobachteten Teil D_{obs} und den fehlenden Teil D_{miss}, multivariat normalverteilt ist. Als Annahme wird also

$$D \sim N_k(\mu, \mathbf{\Sigma})$$

getroffen, wobei μ der Mittelwertsvektor und Σ die Kovarianzmatrix darstellt. Ein offensichtliches Problem ist es, dass nur der beobachtete Teil D_{obs} zur Verfügung steht. Dies ist auch der Grund dafür, dass die Annahme (maximal) MAR fehlender Daten getroffen werden muss, falls auf diese Art und Weise Daten imputierten werden sollen. Die Daten dürfen nicht von dem unbeobachteten Teil D_{miss} abhängen, sondern nur von dem beobachteten Teil D_{obs}. Zusammen mit der Fehlmatrix M, die angibt ob ein Wert in den Daten fehlt oder nicht, kann unter der MAR Annahme [2]

$$p(M|D) = p(M|D_{obs})$$

geschrieben bzw. vereinfacht werden. Die Wahrscheinlichkeit, dass ein Wert fehlt, ist unabhängig davon, ob der Wert aus dem beobachteten Teil oder unbeobachteten Teil von D stammt und es reicht daher, nur den beobachteten Teil zu betrachten.
Zudem sollte die angenommene a posteriori Verteilung von D_{obs} die wahre a posterori Verteilung größtenteils approximieren. Aus diesem Grund ist auch der Erhalt der Regressionsbeziehung von Bedeutung.

Die Informationen dieses Kapitel wurden aus [Spiess 2008], [Feilke 2009] und [Honaker et al. 2013] entnommen.

4.2.2 Der EMB-Algorithmus

Die Likelihood unserer beobachteten Daten ist $p(D_{obs}, M|\theta)$, wobei $\theta = (\mu, \mathbf{\Sigma})$. Unter der MAR Annahme kann man also auch

$$p(D_{obs}, M|\theta) = p(M|D_{obs})p(D_{obs}|\theta)$$

schreiben.

[2] Unter der NMAR Annahme wäre eine solche Vereinfachung nicht möglich gewesen. Da bei der multiplen Imputation (bzw. beim EMB Algorithmus) mit der Likelihood gerechnet wird, wäre der Missingmechanismus im Falle NMAR fehlender Daten also nicht mehr ignorierbar und müsste berücksichtigt werden. Somit wäre eine Imputation in der Art und Weise, wie sie hier beschrieben wird, nicht möglich.

Betrachtet man nur die Interferenz der Parameter der kompletten Daten, kann die Likelihood mit

$$L(\theta|D_{obs}) \propto p(D_{obs}|\theta)$$

beschrieben werden. Benutzt man zusätzlich das "Gesetz der iterierten Erwartungen", so kann wiederum

$$p(D_{obs}|\theta) = \int p(D|\theta)dD_{mis}$$

schreiben. Zusammen mit der Flat Prior von θ, ist der Posterior gegeben mit

$$p(\theta|D_{obs}) \propto p(D_{obs}|\theta) = \int p(D|\theta)dD_{mis}$$

Die größte Schwierigkeit besteht nun darin, aus eben dieser Posterior Werte zu "ziehen". Amelia 2 verwendet dafür den EMB Algorithmus. Dieser wird von [Honaker et al. 2013] wie folgt beschrieben:

> *Our EMB algorithm combines the classic EM algorithm with a bootstrap approach to take draws from this posterior. For each draw, we bootstrap the data to simulate estimation uncertainty and then run the EM algorithm to find the mode of the posterior for the bootstrapped data, which gives us fundamental uncertainty too.*

Der EMB-Algorithmus kombiniert also den EM-Algorithmus mit dem Bootstrap Ansatz. Die beiden Ansätze sind nötig, um den EMB-Algorithmus durchzuführen, aus diesem Grund werden sie kurz aufgeführt:

Der EM-Algorithmus
Mit dem EM-Algorithmus soll das unbekannte θ geschätzt werden, um die geschätzte Wahrscheinlichkeitsverteilung von D approximieren zu können. Der EM-Algorithmus ist ein iterativer Prozess, der sich in zwei Schritte einteilen lässt. Im ersten Schritt, dem Expectation-Schritt(E-step), werden auf Grundlage der beobachteten Werte D_{obs} und des Parameterschätzers $\widehat{\theta}$ aus den vorangegangen Iterationsschritt die fehlenden Werte D_{miss} geschätzt. Da am Anfang noch kein vorheriger Iterationsschritt existiert, generiert Amelia 2 zu Beginn einen Startwert für $\widehat{\theta}$. Im zweiten Schritt, dem Maximization-Schritt (M-Step), wird ein neuer Schätzer für den unbekannten Verteilungsparameter θ auf Basis der beobachteten und geschätzten Zielgrößenwerte bestimmt. Diese geschieht mithilfe der Maximum-Likelihood Methode.

Verändert sich der Parameterschätzer $\widehat{\theta}$ im Vergleich zum vorherigen Iterationsschritt nur noch minimal, so bricht der EM-Algorithmus ab. Mit dem Algorithmus ergibt sich ein Schätzer, der dem ML Schätzer $\widehat{\theta}_{ML-Schätzer}$ nahekommt.

Bootstrapping
Bootstrapping ist eine Resampling Methode, eine Möglichkeit für asymptotische Approximationen. Die grundlegende Idee des Bootstrapping ist es, die vorhandene Stichprobe selbst als Verteilungsmodell zu nehmen. Aus der beobachteten Stichprobe wird zufällig n mal mit Zurücklegen gezogen. Daraus erhält man eine Bootstrap-Stichprobe, die denselben Stichprobenumfang wie der ursprüngliche Datensatz besitzt (vgl. [Fairmeir et al. 2010] und [Engel et al. 2008]).

Die Imputation der fehlenden Werten in zwei Schritten durchgeführt. Zuerst entstehen durch Bootstrapping m Stichprobendatensätze , die durch Ziehung mit Zurücklegen aus dem Datensatz D entstehen. Dadurch wird die Unsicherheit bei der Imputation berücksichtigt, da hier die Variabilität der Datensätze entsteht. Anschließend wird der EM-Algorithmus auf die m Stichprobendatensätze angewendet. Damit erhält man den Punktschätzer für den Mittelwertsvektor μ sowie die Kovarianzmatrix $\mathbf{\Sigma}$. Die durch den EM-Algorithmus entstanden Schätzer werden verwendet, um die fehlenden Werte des ursprünglichen Datensatzes zu ersetzen. Im Zuge dessen entstehen dann m verschiedene (imputierte) Datensätze. Abbildung 10 ließe sich erweitern, bzw. der Schritt 1 der Abbildung würde sich in zwei Schritte zerlegen. Somit wäre der ursprüngliche Imputationsschritt in einen Bootstrapping-Schritt und einen EM-Algorithmus Schritt aufgespaltet (siehe Abb. 11).

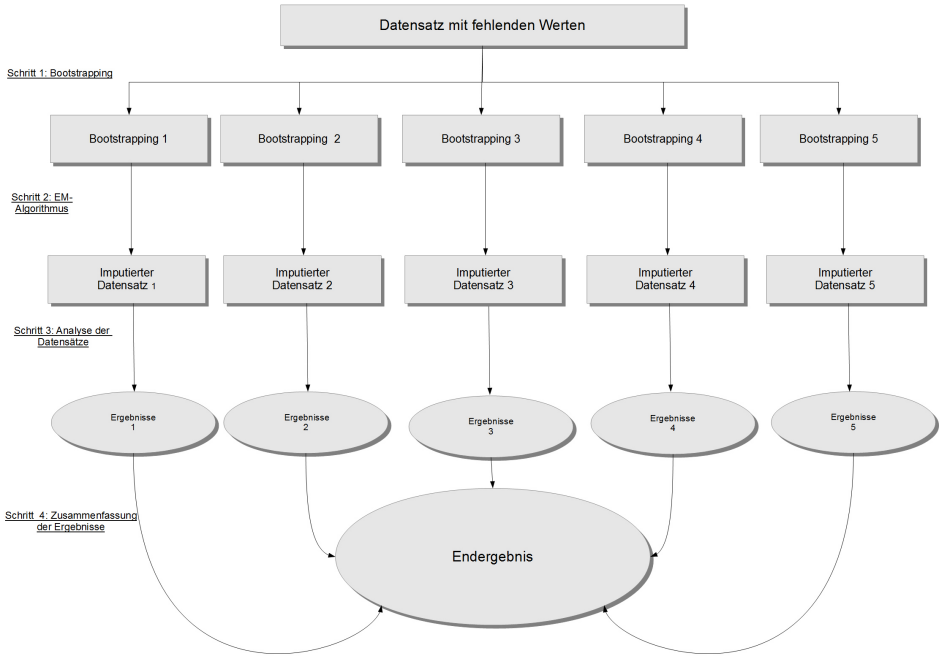

Abbildung 11: Schritte einer multiplen Imputation mit Amelia

Nochmals kurz zusammengefasst, mit der Bootstrapping Methode ziehen wir aus den (unvollständigen) Daten D m-mal mit der jeweiligen Stichprobengröße n. Anschließend wenden wir auf jeden der m Datensätze den EM-Algorithmus an und erhalten so die Punktschätzer θ und $\mathbf{\Sigma}$, die für die Imputation der fehlenden Werte verwendet werden. Schließlich ergeben sich m multipel imputierte Datensätze, die analysiert werden können. Ein Großteil dieses Abschnitts stammt inhaltlich von [Honaker et al. 2013]. Zusätzlich stammen Informationen aus [Feilke 2009] und [Mayer 2010].

4.2.3 Kombination der Ergebnisse

Um die Ergebnisse der verschiedenen Analysen zu kombinieren, schlug Rubin folgende Methode vor: Je nachdem, an welcher statistische Größe Q man interessiert ist (Q könnte beispielsweise der Median oder ein Regressionskoeffizient sein), können die Ergebnisse der Analysen wie folgt zusammengefasst werden:

$$\bar{Q} = \frac{1}{m} \sum_{j=1}^{m} \widehat{Q}_j$$

Beispielsweise könnte man die m erzeugten Datensätze zu einem Einzigen kombinieren, indem man diese Formel auf den Mittelwert der Datensätze anwendet und dadurch den "Mittelwert der Mittelwerte" erhält. Der entstehende Datensatz könnte anschließend mit den Clusterverfahren analysiert werden und die Effizienz ermittelt werden.

Eine bessere Alternative als einen kombinierten Datensatz zu verwenden ist es allerdings, die verschiedenen Clusterverfahren auf jeden der 5 von Amelia imputierten Datensätze anzuwenden und dann die Ergebnisse zu kombinieren. Dazu wird bei jedem Merkmalsträger betrachtet, wie oft er in den einzelnen Amelia Datensätzen in die richtige Gruppe eingeteilt wird. Wurde beispielsweise ein Merkmalsträger in den Amelia Datensätzen 1, 2 und 4 richtig geclustert, so wurde er insgesamt 3 mal richtig eingeteilt. Dies wird für jeden der insgesamt 5000 Merkmalsträger durchgeführt. Dadurch würde folgende Tabelle entstehen:

0xRichtig	1xRichtig	2xRichtig	3xRichtig	4xRichtig	5xRichtig
54	59	67	119	304	4397

Tabelle 2: Beispieltabelle zur Kombination der Ergebnisse

In dieser Form ist das Ergebnis allerdings schwer interpretierbar. Um ein besseres Gesamtergebnis (im Hinblick auf Interpretierbarkeit) zu erhalten, wird die Anzahl der Objekte, die 3 mal oder öfters in den Amelia Datensätzen richtig geclustert wurden, als richtig klassifiziert angesehen. Merkmalsträger, die nur 2 mal oder weniger richtig geclustert wurden, werden dementsprechend als falsch klassifiziert eingestuft. Somit werden also nur Merkmalsträger als richtig eingestuft, die in mehr als der Hälfte der jeweiligen Amelia Datensätze richtig eingestuft worden. Abgeleitet aus Tabelle 2 würde so

Falsch geclustert	Richtig geclustert
180	4820

Tabelle 3: Beispieltabelle 2 zur Kombination der Ergebnisse

resultieren. Daraus könnte man wiederum die Effizienz der Clusteranalyse (nach der Imputation) ableiten durch Division von der Anzahl an richtigen Clustereinteilungen (aus Tabelle 3) durch die Anzahl an Merkmalsträgern (5000). So kann die Genauigkeit der Clusteranalyse nach der Imputation widergespiegelt werden. In dieser Art und Weise können die Ergebnisse der verschiedenen Amelia Datensätze zu einem Endergebnis kombiniert werden.

Die Informationen für diesen Abschnitts wurden aus [Honaker et al. 2013] und [Schaefer o.D] entnommen.

4.2.4 Umsetzung in R

Wie oben beschrieben, wird die multiple Imputation mithilfe des *Amelia 2* packages aus dem Programm R durchgeführt. Dem Paket muss für die multiple Imputation lediglich der unvollständige Datensatz (Datensatz NA1 bzw. Datensatz NA2), die Anzahl an Imputationen m, die durchgeführt werden sollen sowie im Falle kategorialer Variablen das Skalenniveau. Somit werden dem Programm die Datensätze 1 bis 5 mit fehlenden Werten übergeben, einmal mit und einmal ohne kategoriale Variable. Die Anzahl der Datensätze, welche die multiple Imputation erzeugen soll, wird auf 5 festgelegt. Eine größere Anzahl ist, wie in Kapitel 4.2.1 beschrieben, nicht nötig. Es ergeben sich also für jeden der 5 Datensätze mit fehlenden Werten wiederum je 5 Datensätze mit vervollständigten Werten. Diese können einzeln ausgewählt und analysiert werden. Insgesamt erhält man also sowohl für die jeweiligen Datensätze NA1 als auch für jeweiligen Datensätze NA2 je 5 Amelia Datensätze. Für die praktische Umsetzung wurde eine neue Funktion geschrieben. Für die Datensätze 4 und 5 ist zudem eine kleine Modifikation nötig. Da die Werte nur zwischen 0 und 1 liegen dürfen, muss auch die Imputation auf diesen Bereich eingegrenzt werden.

Die praktische Umsetzung zur Kombination der Ergebnisse (wie in Kapitel 4.2.3) ist später in Kapitel 6.1.1 zu lesen, da hierzu die Clusterzuordnungen benötigt werden. Um die Auswirkung der Imputation grafisch darzustellen, sind in den Abbildungen 12 und 13 die einzelnen Datensätze vor und nach der Imputation dargestellt. In jeder Reihe ist nochmal der Original Datensatz abgebildet, dann die Imputation des Datensatzes ohne kategoriale Variable und schließlich die Imputation des Datensatzes mit kategorialer Variable. Exemplarisch wurde jeweils der erste von den 5 imputierten Datensätze zur Veranschaulichung verwendet. Dabei wurden die Merkmalsträger, je nach dem aus welcher Gruppe sie wirklich stammen, farblich markiert. Somit kann bereits ein Urteil darüber abgegeben werden, ob die Imputation als gut eingestuft werden kann, auch wenn es eigentlich für jeden Datensatz mehrere Imputationen gibt.
Im Folgenden sind die Abbildungen der Imputationen der Datensätze mit 30 % entfernten Daten abgebildet:

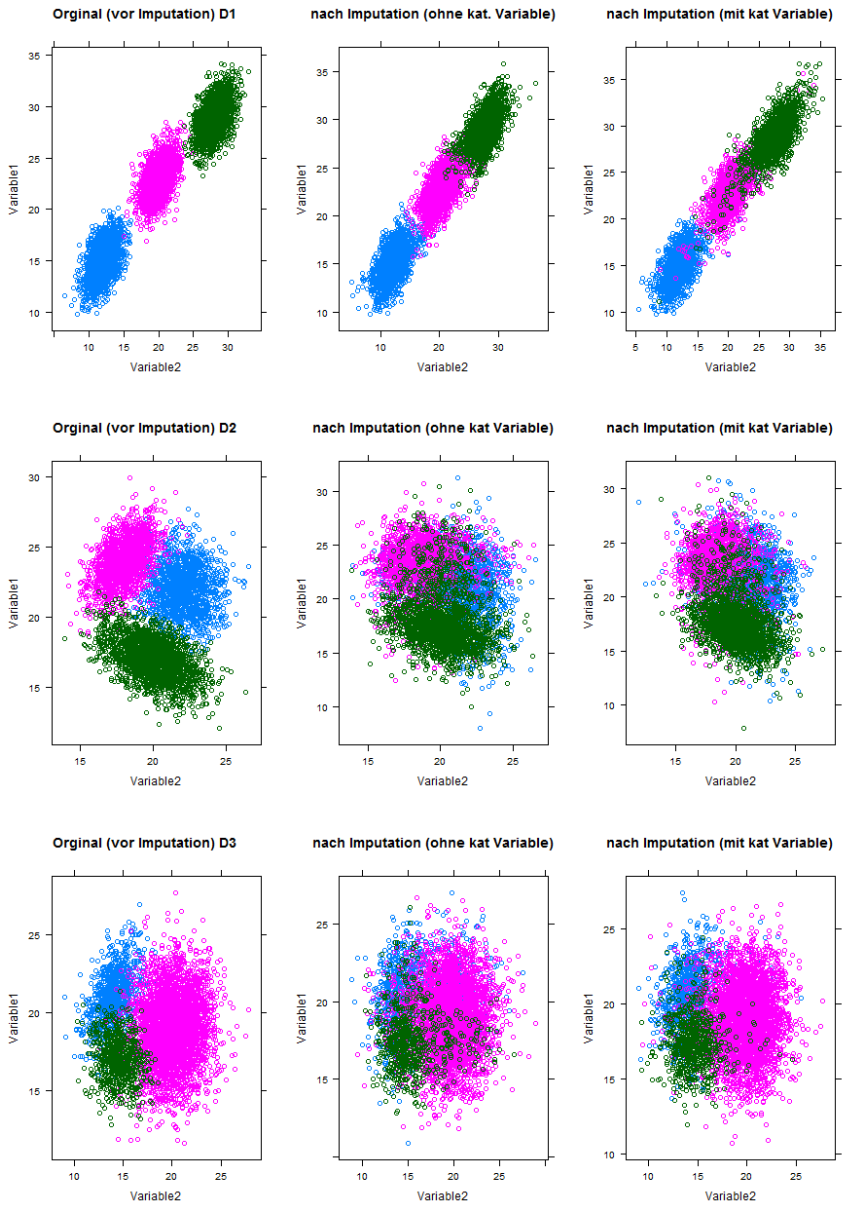

Abbildung 12: Auswirkung der Imputation Datensatz 1 bis 3 (30 % fehlende Daten)

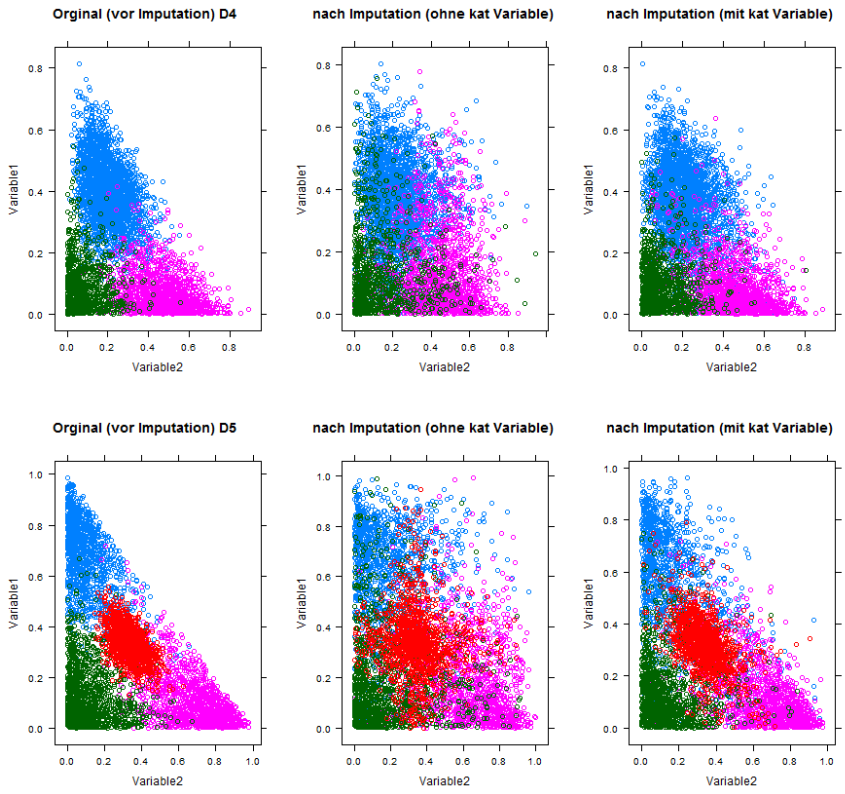

Abbildung 13: Auswirkung der Imputation Datensatz 4 und 5 (30 % fehlende Daten)

Bei Betrachtung der Abbildungen stellt man nach der Imputation Veränderung im Vergleich zum ursprünglichen Datensatz fest. Allerdings halten sich diese im akzeptablen Rahmen. In den Datensätzen 4 und 5 werden zwar nur Werte zwischen 1 und 0 imputiert, allerdings auch viele Werte außerhalb des "Dreiecks", wie es in den ursprünglichen Datensätzen zu beobachten ist. Dies fällt besonders bei den beiden mittleren Plots aus Abbildung 13 auf. Die benötigten Funktionen zur Durchführung sind im R-Code ab Seite 99 bzw. Seite 117 zu finden.

5 Clusteranalyse

In vielen Bereichen der Wissenschaft ist es häufig ein Anliegen, herauszufinden, ob zwischen den betrachteten Untersuchungsobjekten Ähnlichkeiten bestehen. Darüber hinaus möchte man ähnliche Untersuchungsobjekte zu Gruppen (Cluster) zusammenfassen, in der alle Elemente mit ähnlichen Eigenschaften vereint sind. Beispielsweise werden in Biologie Ähnlichkeiten zwischen Kleinlebewesen gesucht, um eventuell vorhandene Verwandtschaftsbeziehungen finden zu können. Ein anderes Beispiel wäre im Bereich des Marketings, wo geografische Gebiete mit ähnlichen Absatzmerkmalen in Gruppen zusammenfasst werden sollen, um beispielsweise neue Produkte zuerst in repräsentativen Märkten testen zu können. Die Clusteranalyse findet in den verschiedensten Bereichen der Wissenschaft Anwendung. Innerhalb der einzelnen Gruppen, in die Merkmalsträger mit ähnlichen Merkmalen eingeteilt werden, sollten dabei möglichst große Ähnlichkeiten bestehen, die Eigenschaften bzw. Merkmale sollten also möglichst homogen sein. Hingegen sollten zwischen den einzelnen homogenen Gruppen möglichst große Unterschiede bestehen. Das Ziel der Clusteranalyse ist die Klassenbildung, aus diesem Grund werden die Clustermethoden auch häufig als Klassifikationsverfahren bezeichnet. Unter dem Begriff Clusteranalyse sind zahlreiche verschiedene Clusterverfahren zusammengefasst. In Abbildung 14 ist eine Übersicht der Clusterverfahren zu finden, die in dieser Arbeit für die Clusteranalyse verwendet und in diesem Kapitel beschrieben werden.

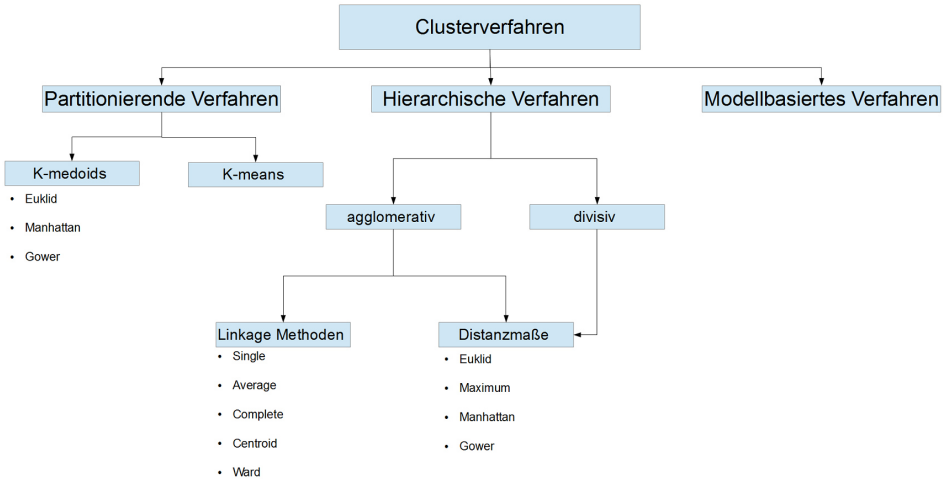

Abbildung 14: Übersicht über verwendete Clusterverfahren

Eine wichtige Vorraussetzung für die Clusterverfahren sind komplette Daten. Merkmalsträger mit fehlenden Beobachtungen können keiner Gruppen zugeordnet werden. Deshalb ist es notwendig, fehlende Daten zu impu-

tieren.

Bei den Clusterverfahren werden alle Merkmale der Merkmalsträger gleichzeitig zur Gruppenbildung berücksichtigt. Da die Gruppen im Vornherein nicht bekannt sind, zählen die Clustermethoden zu den explorativen Verfahren der multivariaten Statistik [Fairmeir et al. 1996].

5.1 Hierarchische Klassifikationsverfahren

Die hierarchischen Klassifikationsverfahren zählen in der Praxis zu den am weitesten verbreiteten Clusterverfahren. Man unterscheidet in agglomerative und divisive hierarchische Klassifikationsverfahren. Bei agglomerativen Verfahren stellt zu Beginn jedes einzelne Objekt einen einzelnen Cluster da, die solange zusammengefasst werden, bis alle Objekte in einem Cluster vereint sind. Im Gegensatz dazu gibt es noch divisive hierarchische Clusterverfahren. Hier wird mit einen großen Cluster begonnen, in dem alle Objekte gesammelt sind und trennt diesen solange, bis jedes einzelne Objekt einen eigenen Cluster repräsentiert. Bei den agglomerativen Verfahren wird die Anforderung an die Homogenität (der einzelnen Cluster) schrittweise verringert, bei divisven schrittweise erhöht. Divisive Verfahren erreicht zwar eine höhere Homogenität, sind aber auch mit weit mehr Rechenaufwand verbunden und aus diesem Grund weniger verbreitet als agglomerative Verfahren. Beide Vorgehensweisen eignen sich gut, wenn in der vorhandenen Objektmenge eine hierarchische Struktur vorhanden ist. Sie werden in Abschnitt 5.1.2 bzw. 5.1.3 näher beschrieben.

5.1.1 Ähnlichkeits- und Distanzmaße

Bevor man Merkmalsträger in Gruppen einteilen kann, müssen Objekte gefunden werden, die sich ähnlich sind. Um die Ähnlichkeit zweier Objekte bzw. Merkmalsträger zu bestimmen, werden Ähnlichkeits- und Distanzmaße verwendet. Sie geben durch einen Zahlenwert die Unterschiede bzw. die Gemeinsamkeiten zweier Objekte wieder.

- Ähnlichkeitsmaße zeigen auf, wie ähnlich zwei Objekte sind. Je größer der Wert des Ähnlichkeitsmaßes, desto ähnlicher sind sich die zwei betrachteten Objekte.
- Distanzmaße spiegeln im Gegensatz zu den Ähnlichkeitsmaßen die Unähnlichkeit wider. Hier gilt, je größer die Distanz, desto unähnlicher sind sich die zwei betrachteten Objekte.

Allgemein werden Ähnlichkeits- und Distanzmaße auch als Proximitätsmaße bezeichnet. Für die in dieser Arbeit angewendeten Clusterverfahren werden

ausschließlich Distanzmaße benötigt. Diese Distanzmaße werden auch Metriken genannt und bilden die Grundlage für die hierarchischen Verfahren. Folgende Distanzmaße werden verwendet:

- **Euklidische Distanz**
  Die Euklidische Distanz ist eines der am weit verbreitetsten Distanzmaße. Sie entspricht auch der anschaulichen Distanzvorstellung, man kann sich die euklidische Distanz auch als Luftlinie zwischen zwei Punktion vorstellen. Für jedes Paar von Objekten wird die Differenz gebildet und quadriert, anschließend werden die Differenzen addiert. Aus dieser Summe wird daraufhin die Quadratwurzel gezogen. So ergibt sich die Euklidische Distanz. Als mathematische Formel wird sie wie folgt ausgedrückt:

  $$d_E(i,j) = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2}$$

  d(i,j) ist dabei die Distanz der Objekte i und j. x_{ik} bzw. x_{jk} ist der Wert der Variablen k bei den Objekten i und j. p entspricht der Anzahl an Variablen. Die euklidische Distanz ist also die Quadratwurzel aus den quadrierten Differenzen.

- **Maximum-Distanz**
  Die Maximum-Distanz gehört ebenfalls zu den verwendeten Standard-Distanzen, sie wird auch die Chebyshev oder Schachspieler Distanz genannt. Betrachtet man ein Schachfeld, entspricht die Maximum-Distanz der Bewegung eines Königs[3], um sich von einem Schachfeld zu einem beliebigen anderen Feld zu bewegen. Mathematisch drückt sich das wie folgt aus:

  $$d_{max}(i,j) = \max_k(|x_i - x_j|)$$

  Die Maximum-Distanz ist also die größte absolute Differenz eines Wertepaares.

- **Manhattan-Distanz**
  Die Manhattan-Distanz ist ebenfalls ein weit verbreitetes Distanzmaß. Stellt man sich zwei Punkte als Ecken eines Häuserblocks vor, so ist die Manhattan-Distanz die Strecke, die entstehen würde, wenn die Häuserblockseiten abgegangen werden[4]. Daher wird sie auch City-

[3] Ein König darf sich im Schachspiel in alle Richtungen bewegen ,also auch diagonal.

[4] Um es erneut in Schachsprache auszudrücken, wäre die Manhattan-Distanz die Bewegung, die ein König auf einen Schachfeld absolviert, wenn er sich nur orthogonal bewegen dürfte.

Block-Metrik oder Taxifahrermetrik bezeichnet. Mathematisch ausgedrückt:

$$d_M(i,j) = \sum_{k=1}^{p} |x_{ik} - x_{jk}|$$

Die Manhattan-Distanz gibt also die Summer der absoluten Differenzen an. Die Distanz wird an der größten absoluten Differenz eines Wertepaares gemessen.

Auf Grundlage jeder einzelnen Distanz d(i,j) zwischen den Datenpunkten wird die sogenannte symmetrische Distanzmatrix gebildet. Diese Matrix benutzen später bestimmte Verfahren, um die Clustereinteilung durchzuführen. Es gibt noch zahlreiche andere Distanz- bzw. Ähnlichkeitsmaße, hier wurden lediglich diejenigen aufgezählt, die später für die Clusteranalyse verwendet werden. Die drei aufgezählten Distanzmaße haben die Gemeinsamkeit, dass diese allesamt Proximitätsmaße für metrische Daten sind. Das bedeutet auch, dass das kategoriale Merkmal aus den erzeugten Datensätzen nicht in der Clusteranalyse berücksichtigt wird, da keine Distanz für gemischt skalierte Daten auf Grundlage dieser Metriken berechnet werden kann. Eine Möglichkeit, die zusätzlich kategoriale Merkmale in die Distanzberechnung miteinzubeziehen, ist die sogenannte Gower Distanz:

- **Gower-Distanz**
  Mathematisch drückt sich die Gower-Distanz folgendermaßen aus:

$$d_G(i,j) = \frac{\sum_{k=1}^{p} \delta_{ij}^{(k)} d_{ij}^{(k)}}{\sum_{k=1}^{p} \delta_{ij}^{(k)}}$$

  Für kategoriale Merkmale gilt dabei:

$$d_{ij}^{(k)} = \begin{cases} 1, & \text{wenn } x_{ik} \neq x_{jk} \\ 0, & \text{wenn } x_{ik} = x_{jk} \end{cases}$$

  Für metrische Merkmale hingegen gilt:

$$d_{ij}^{(k)} = \frac{|x_{ik} - x_{jk}|}{R_k} \quad wobei \;\; R_k = \max_i x_{ik} - \min_i x_{ik}$$

  Mit δ_{ij} wird die Symmetrie binärer Merkmale berücksichtigt.

Mit der Gower Distanz können in der Clusteranalyse also auch kategoriale Variablen berücksichtigt werden.

Die Maße quantifizieren die Ähnlichkeit (bzw. die Distanz). Der Datensatz stellt dabei die Grundlage dar, der anschließend mithilfe der Maße in eine Distanzmatrix überführt wird. Diese Distanzmatrix bildet den Ausgangspunkt für die weiteren Schritte der hierarchischen Clusteranalyse.

Die Informationen dieses Abschnittes stammen größtenteils aus [Fairmeir et al.1996], es wurden aber auch Teile aus [Unbekannt o.D(3)],[Handl 2002] und [Unbekannt 2010] übernommen.

5.1.2 Agglomerative Verfahren

Agglomerative Verfahren lassen sich in drei Schritte gliedern:

Schritt 1: Im ersten Schritt stellt jede Beobachtung ihren eigenen Cluster da. Im Anschluss wird die Distanz zwischen jedem Objekt bzw. Cluster mithilfe der Proximitätsmaße bestimmt.
Schritt 2: Zwei Cluster, die sich am nächsten sind, werden zu einem neuen Cluster zusammengefasst.
Schritt 3: Erneut werden die Distanzen zwischen den Cluster bzw. Objekten berechnet.

Schritt 2 und 3 werden dabei so lange wiederholt, bis alle Objekte in einem einzigen Cluster vereint sind. Die Linkage Verfahren legen bei den agglomerativen Verfahren den Fusionsalgorithmus fest, nachdem die Objekte bzw. Cluster in Schritt 2 zu neuen Cluster zusammengefasst werden. Auch hier gibt es eine Vielzahl an Algorithmen (Linkage Methoden) bzw. Möglichkeiten, die Gruppierung vorzunehmen. Als Ausgangspunkt benutzen diese Algorithmen die mit den Proximitätsmethoden ermittelte Distanzmatrix. Folgende Linkage Verfahren werden verwendet:

- **Single-Linkage**
  Das Single-Linkage-Verfahren (auch nearest neigbour Methode genannt) fusioniert diejenigen Objekte, die in der ermittelten Distanzmatrix die kleinsten Distanzen aufweisen. Mathematisch drückt sich das wie folgt aus:

  $$D_S(C_k, C_l) = \min_{i \in C_k j \in C_l} (d(i, j))$$

  $D(C_k, C_l)$ ist die Distanz der Cluster C_k und C_l. Diese entspricht im Single-Linkage Verfahren der kleinsten Distanz zwischen jeweils einen Objekt aus C_k und C_j. Für die Fusion reicht es, wenn ein ein Objekt aus einer Klasse einem Objekt aus eine anderen Klasse nahe liegt (im Sinne des Distanzmaßes). Die Distanzen der restlichen Objekte

können dabei aber voneinander abweichen, hier existieren keine speziellen Forderungen. Deshalb ist das Single-Linkage-Verfahren im Vergleich zu anderen Linkage Methoden eher in der Lage, Klassen unterschiedlichster Formen zu identifizieren. Dies gilt allerdings nur, wenn zwischen den Klassen keine unbesetzten Räume zu finden sind. Andernfalls werden dann nämlich auch Klassen verbunden, die ansonsten deutlich voneinander getrennt sind. Dies wird als Verkettungseigenschaft bezeichnet. Aus diesem Grund eignet sich das Verfahren nicht, wenn möglichst homogene Gruppen gefunden werden sollen. Jedoch eignet es sich, dadurch das immer die "nächsten Nachbaren" fusioniert werden, zur Auffindung von Ausreißern, da diese erst zu Ende fusioniert werden.

- **Complete-Linkage**
  Im Gegensatz zum Single-Linkage-Verfahren werden beim Complete-Linkage-Verfahren (auch furthest neigbour Methode genannt) die Cluster über den Maximal Abstand zwischen den Objekten zweier Cluster vereint. Dabei wird wie beim Single-Verfahren nur die extremste Distanz betrachtet,was bei dem Complete-Verfahren die maximale Distanz ist. Auch in der Formel ist die Ähnlichkeit zum Single-Verfahren zu sehen, hier wird anstelle des Minimums ein Maximum geschrieben:

  $$D_C(C_k, C_l) = \max_{i \in C_k j \in C_l} (d(i,j))$$

  Durch den Fusionsalgorithmus eignet sich das Verfahren gut, wenn das primäre Ziel die Findung homogener Gruppen ist. Allerdings ergibt sich aufgrund der späte Fusion weit entfernter Klassen die Anfälligkeit gegenüber Ausreißern.

- **Average-Linkage**
  Beim Average-Linkage-Verfahren werden die Klassen zusammengefasst, dessen Objekte im Mittel ähnlich sind. Mathematisch äußert sich das wie folgt:

  $$D_A(C_k, C_l) = \frac{1}{n_k n_l} \sum_{i \in C_k} \sum_{j \in C_l} d_{ij}$$

  n_k bzw. n_l ist die Anzahl der Objekte der Klasse C_k bzw. C_l. Die Distanz zwischen den Gruppen gleicht hier dem Mittel aller Distanzen zwischen den Objekten aus den Klassen. Hier reicht es also, wenn die Objekte im Mittel ähnlich sind, um sie zu fusionieren.

- **Centroid-Verfahren**
  Beim Centroid-Verfahren (Zentroid-Verfahren) wird jede Klasse durch

ihren Klassenschwerpunkt

$$\bar{x}_k = \frac{1}{n_k} \sum_{n \in C_k} x_n$$

charakterisiert. Der Abstand zwischen den Klassen wird anschließend durch die quadrierte euklidische Distanz der Klassenschwerpunkte gemessen:

$$D_Z(C_k, C_l) = ||\bar{x_k} - \bar{x_l}||^2$$

Wie beim Average-Verfahren reicht es zur Klassenfusion aus, wenn die Objekte zweier Klasse im Mittel ähnlich sind. Die beiden Verfahren unterscheiden sich lediglich dadurch, dass beim Average-Verfahren die mittlere Abweichung der Distanzen und beim Centroid-Verfahren die mittlere Abweichung der Variablenausprägungen betrachtet wird. Beide zählen zu den Mittelwertverfahren.

- **Ward-Verfahren**
  Das Ward-Verfahren zählt in der Praxis zu den am weitesten verbreiteten Fusionsalgorithmen. Während die anderen Algorithmen die geringste Distanz zwischen Objekte zu Grunde legen, vereint das Ward Verfahren diejenigen Objekte bzw. Gruppen, die ein vorgegebenes Heterogenitätsmaß am geringsten vergrößert. Als Heterogenitätsmaß wird dabei das Varianzkriterium(Fehlerquadratsumme) verwendet, das sich für eine Gruppe g wie folgt berechnet lässt:

$$V_g = \sum_{f=1}^{K_g} \sum_{j=1}^{J} (x_{fjg} - \bar{x}_{jg})^2$$

  x_{fjg} ist der beobachtete Wert der Variable j (j=1,...,J) bei Objekt f (für alle Objekte f=1,...,K_g in der Gruppe g).$\bar{x}_j g$ ist dabei der Mittelwert über die beobachteten Werte der Variablen j in Gruppe g (entspricht also $\frac{1}{K_g} \sum_{f=1}^{K_g} x_{fjg}$).
  Als Kriterium für die Zusammenfassung werden also die Objekte bzw. Gruppen vereint, welche die Fehlerquadratsumme am wenigsten erhöhen. Mathematisch dargestellt äußert sich das wie folgt:

$$D_W(C_k, C_l) = \frac{n_k n_l}{n_k + n_l} ||\bar{x_k} - \bar{x_l}||^2$$

  Das Ward-Verfahren minimiert zwar sozusagen den Homogenitätsverlust, liefert jedoch nicht zwangsläufig die optimale Partition bzg. des Varianzkriteriums. Mit dem Ward Verfahren gelingt es in der Praxis häufig,

homogene Gruppen zu erzeugen. Deshalb gilt es als eines der leistungsstärksten agglomerative Verfahren. Allerdings neigt das Ward-Verfahren zur Bildung gleich großer Cluster. Aus diesem Grund fällt es dem Verfahren schwer, Gruppen mit stark unterschiedlichen Gruppengrößen zu identifizieren.

Die verschiedenen Linkage Methoden werden nur bei den agglomerativen Clusterverfahren benötigt. Sie legen den Fusionsalgorithmus fest, also das Schema, nachdem verschiedene Objekte bzw. Gruppen zu neuen Gruppen zusammengefasst werden.
Die Ergebnisse der hierarchisch agglomerativen Verfahren lassen sich in übersichtlicher Weise in einem Dendogramm darstellen, das eine grafische Baumstruktur besitzt. Als Beispiel ist in Abbildung 15 das Dendogramm der Clusteranalyse mit euklidischer Distanz und dem Ward-Linkage abgebildet..

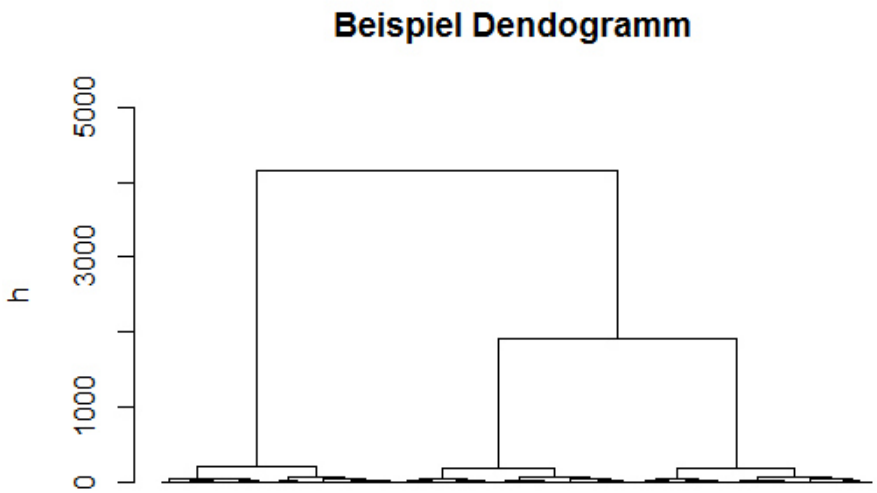

Abbildung 15: Beispiel Dendogramm zu Datensatz 1

Auf der x-Achse befinden sich die n Objekte, die nach und nach zu neuen Klassen fusioniert werden. Der Index h (auf y-Achse) misst die Homogenität der Klassen. Je kleiner dieser Wert h ist, auf dem die Klassen zu einer neuen Klasse fusioniert werden, desto ähnlicher sind sich die Klassen bzw. deren Objekte und desto homogener sind auch die Klassen. Bei agglomerativen Verfahren stellt jedes Objekt zu Beginn einen eigenen Cluster dar und am Ende sind alle Objekte in einem Cluster gesammelt. Dementsprechend konstruiert sich das Dendogramm von unten nach oben. Es liefert den Vorteil, dass der Anwender je nach seiner Homogenitätsanforderung die Klassenbildung ablesen kann. Abhängig von der Wahl des Distanzmaßes und der

Linkage Funktion ergeben sich unterschiedliche Dendogramme.

Die Informationen dieses Abschnitts wurden aus [Fairmeier et al. 1996], [Stein et al. 2011] und [Landscape Ecology Lab o.D] entnommen.

5.1.3 Divisive Verfahren

Zur Erinnerung, divisive Verfahren beginnen mit einem großen gemeinsamen Cluster und werden Schritt für Schritt in kleinere Cluster geteilt, solange jedes Objekt einen eigenen Cluster darstellt. Während bei den agglomerativen Verfahren durch die Linkage-Methoden Objekte bzw. Gruppen zusammengefasst werden, teilt das divisive Verfahren großere Gruppen in kleinere Gruppen bzw. Objekte. Wie bei der agglomerativen Clusterung mit verschiedenen Linkage-Methoden gibt es auch bei den divisiven Verfahren verschiedene Verfahren zur Aufspaltung. Es wird zwischen monoethetischen und polythetischen Verfahren unterschieden. Während monoethetische Verfahren als Grundlage für die Zerlegung nur ein Merkmal betrachten, beziehen polythetischen Verfahren in jedem Schritt alle Merkmale gleichzeitig mit ein. Auch hier lässt sich das Verfahren in Schritte einteilen:

Schritt 1: Im ersten Schritt muss das Objekt gefunden werden, dass die höchste durchschnittliche Abweichung gegenüber allen anderen Objekten besitzt. Dieses Objekt repräsentiert dann einen neuen Cluster R.
Schritt 2: Für jedes Objekt außerhalb dieses neuen Clusters gilt

$$D_h = [average \;\; d(i,j) \;\; j \notin R] - [average \;\; d(i,j) \;\; j \in R]$$

Grundlage der Berechnung sind also wieder die Distanzmaße d(i,j).
Schritt 3: Es muss das Objekt h gefunden werden, für das die Differenz D_h am größten wird. Falls D_h positiv ist, ist h im Durchschnitt nahe an dem Cluster R.
Schritt 4: Der zweite Schritt muss so lange wiederholt werden, bis die Differenzen D_h negativ sind. Dann ist der Datensatz in zwei Cluster geteilt.
Schritt 5: Nun wird der Cluster gesucht, bei dem die Unähnlichkeit zwischen 2 beliebigen Objekten des Clusters am größten ist. Mit diesem Cluster führt man die Schritte 1-4 erneut durch. Dadurch wird dieser Cluster wieder zweigeteilt.

Anschließend wiederholt man Schritt 5 solange, bis jedes Objekt einen eigenen Cluster darstellt. Hier wird deutlich, die Berechnung divisiver Verfahren deutlich aufwändiger ist. Nach diesem beschrieben Ablauf verläuft das divisive Verfahren, wie es von dem R-Paket “DIANA” angewandt wird. Auch hier ist eine grafische Darstellung mit einem Dendogramm möglich. Ein Dendogramm der Clusteranalyse von Datensatz 1 mit dem divisiven Verfahrens und euklidischen Distanzmaß ist in Abb. 16 dargestellt.

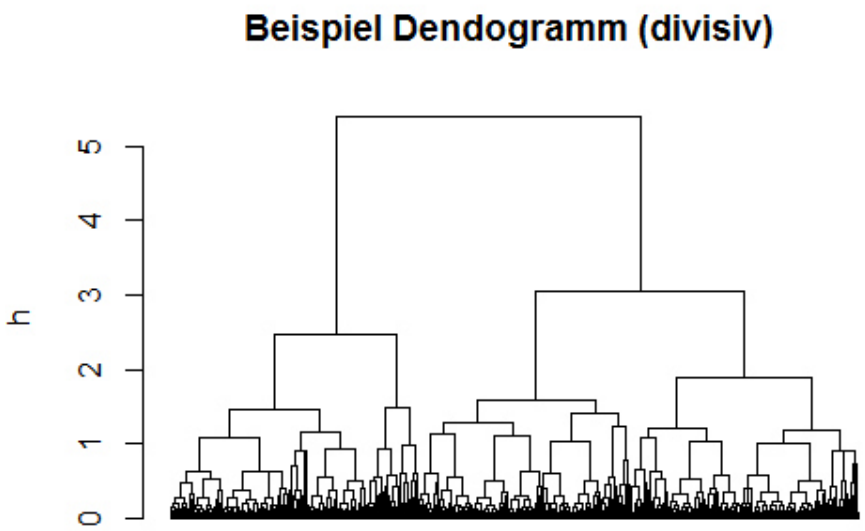

Abbildung 16: Beispiel Dendogramm zu Datensatz 1 (divisiv)

Grafisch ähneln die Dendogramme divisiver Verfahren denen der agglomerativen Verfahren. Bei divisiven Verfahren wird das Dendogram allerdings von oben nach unten konstruiert. Dies entspricht auch dem Vorgehen des divisiven Verfahrens, mit einem großen Cluster zu beginnen und ihn anschließend in die einzelnen Objekte zu zerlegen. Auch hier sind die Klassen homogener, je kleiner der Wert von h ist.

Für diesen Abschnitt wurden Informationen aus [Fairmeir et. al 1996], [Rohwer et al. 2011], [Stein et al. 2011], [Landscape Ecology Lab o.D], [Unbekannt(1) o.D] und [R-Dokumentation(3) o.D] entnommen.

5.1.4 Bestimmung der Clusterzahl

Nachdem die hierarchische Clusteranalyse durchgeführt wurde, ist entweder ein großer Cluster mit allen Objekten oder viele Cluster mit jeweils einem Objekt vorhanden. Mit beiden finalen Lösungen kann man aus praktischer Sicht nichts anfangen. Normalerweise sollte der Anwender festlegen, wie viele Cluster für die jeweilige Anwendung sinnvoll sind. Häufig gibt es beim Anwender allerdings keine sachliche Gründe für eine Gruppierung. Aus diesem Grund sollte mithilfe der Clusteranalyse versucht werden, eine vorhandene Gruppierung in den Daten zu finden. Dies sollte auf Grundlage statistischer Kriterien geschehen. Bei der Entscheidung über die optimale Clusteranzahl muss ein Kompromiss zwischen der Handbarkeit der Clusterlösung und der Homogenitätsanforderung an die Cluster-Lösung gefunden werden. Hier könnten auch sachlogische Überlegungen berücksichtigt

werden, die sich aber nur auf die Anzahl der zu wählenden Cluster beziehen sollte. Eine einfache und gute Möglichkeit zur Identifikation der optimalen Clusteranzahl liefert der sogenannte "Scree-Plot". In diesem Plot wird die Heterogenitätsentwicklung nach der Clusteranzahl betrachtet. Auf der x-Achse ist die Clusteranzahl, auf der y-Achse die Fehlerquadratsumme abgetragen. Zeigt sich in dem Plot ein "Ellenbogen" (in der Entwicklung des Heterogenitätsmaßes), so kann die entsprechende Clusteranzahl als Lösung verwendet werden. Als Beispiel ist der Screeplot zu Datensatz 1 in Abb. 17 dargestellt. Hier zeigt sich, dass sich die Fehlerquadratsumme nach 3 Clustern nicht mehr viel verändert. Dort befindet sich dann auch der "Ellenbogen" (bzw. "Ellbow", siehe ebenfalls Abb. 17).

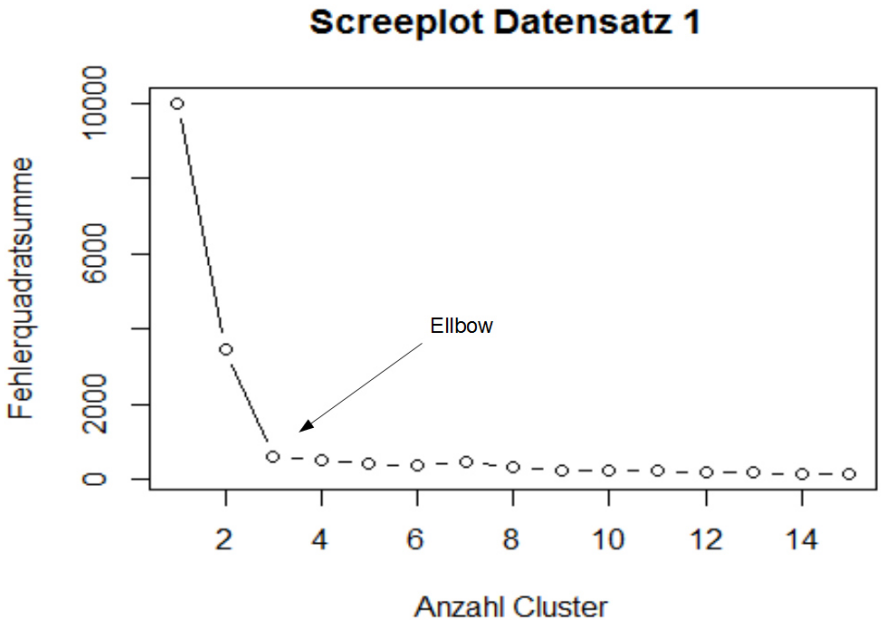

Abbildung 17: Screeplot zu Datensatz 1

Anhand des sogenannten "Ellenbogen" Kriteriums wird die optimale Clusteranzahl bestimmt. Im Datensatz 1 wäre die optimale Anzahl an Clustern also 3, was auch der Anzahl an tatsächlichen Gruppen entspricht. Mit der Kenntnis über die Anzahl der Cluster lässt sich dies auch in das Dendogramm einschließen. Das Dendogramm aus Abbildung 15 sieht dann wie folgt aus:

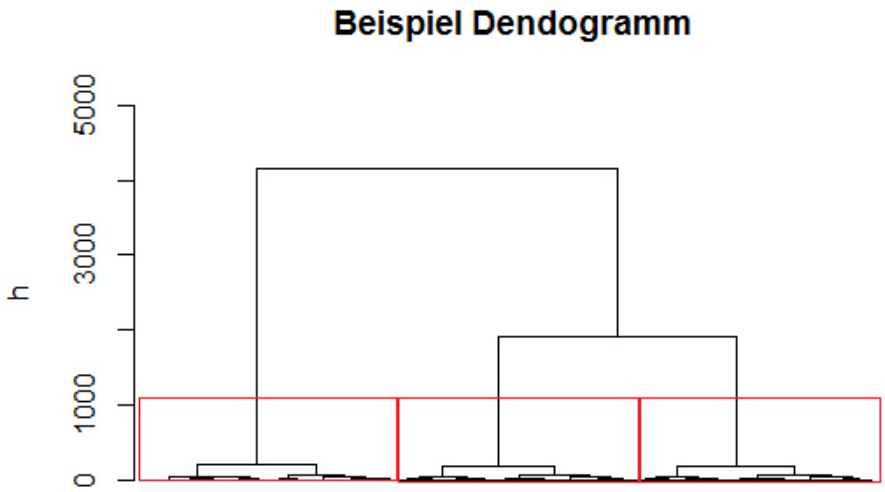

Abbildung 18: Dendogramm mit eingezeichneten Gruppen

Die drei roten Vierecke in Abbildung 18 umfassen jeweils die Objekte eines Clusters (vgl. [Fairmeir et al. 1996]).

5.2 Partitionierende Verfahren

Im Gegensatz zu den hierarchischen Verfahren muss bei den partitionieren Verfahren die Anzahl der Cluster im Voraus bekannt sein. Die Clusteranzahl kann wieder aus dem Screeplot ermittelt werden. Die Klassifikationen der Objekte sind zu Beginn durch zufällige Anordnung in die vorgegebenen Cluster verteilt. Die Zerlegung einer Objektmenge in g Klassen wird auch als Partition bezeichnet. Bei partitionierenden Verfahren wird diejenige Partition gesucht, die hinsichtlich eines bestimmten Gütekriteriums optimal ist. Eine Optimierung des jeweiligen Gütekriteriums wird dadurch erreicht, dass schrittweise Objekte zwischen den Cluster verschoben werden. Dies wird schließlich solange wiederholt, bis sich eine Partition durch weiteres Verschieben von Objekten nicht mehr hinsichtlich des Gütekriteriums optimiert wird(vgl. [Hueftle 2006] und [Fairmeir et al. 1996]).

5.2.1 K-means Verfahren

Einer der bekannsten Vertreter der partionierenden Verfahren ist das sogenannte K-means Verfahren. Bei diesem Verfahren, das auch Austauschverfahren genannt wird, wird als Gütekriterium das auf Seite 46 beschriebene Varianzkriterium verwendet. Die genauen Schritte des Austauschverfahrens gliedern sich wie folgt:

Schritt 1: Im ersten Schritt muss die Clusteranzahl G ($g = 1, ..., G$) vorgegeben werden. Anschließend wählt der Algorithmus zufällig G verschiedene Objekte als Startwerte für die Zentroide aus.
Schritt 2: Jedes Objekt wird denjenigen Cluster zugeordnet, zu dem der euklidische Abstand am geringsten ist.
Schritt 3: Es ergeben sich G Gruppen mit den Objekten. Für jede Gruppe werden die neuen Zentroide berechnet:

$$\bar{x}_j^{(G)} = \frac{1}{n_G} \sum_{i \in C_G} x_{ij}$$

Schritt 4: Die Schritte 2 und 3 werden solange wiederholt, bis die geringste Fehlerquadratsumme gefunden ist.

Ziel ist also die Minimierung der Fehlerquadratsumme (Varianzkriterium). Diese berechnet sich wie folgt:

$$\sum_{g=1}^{G} \sum_{x \in C_g} d_E(c_g, x)^2$$

c_g ist dabei der g-te Cluster mit dem Zentrum c_g. Es wird also die Fehlerquadratsumme in jedem Cluster berechnet und summiert diese anschließend. Aufgrunddessen, dass je nach Wahl der Zentroide in Schritt 1 unterschiedliche Ergebnisse resultieren, wird das K-means mehrmals mit unterschiedlichen Start-Zentroiden durchgeführt. Am Ende entscheidet man sich für den Durchlauf, bei dem die Fehlerquadratsumme am geringsten ist. Die Idee, das Varianzkriterium als Gütekriterium zu verwenden, begründet sich durch die Vermutung, dass eine Klasse mit ähnlichen Objekten auch eine kleine Streuung innerhalb der Klasse besitzt. Aufgrund der Skaleninvarianz des Kriteriums müssen die Daten standardisiert werden. Außer dem Varianzkriterium könnten auch andere Gütekriterium verwendet werden.

Für diesen Abschnitt wurden Informationen aus [Fairmeier et al. 1996], [Wiedenbeck et al. o.D], [R Dokumentation o.D(1)] und [Oellinger 2010] entnommen.

5.2.2 PAM Verfahren

PAM steht für "Partitioning Around Medoids" und stellt eine robustere Verallgemeinerung des K-means Verfahren dar.Auch hier muss die Klassenanzahl vorgegeben werden. Anstelle der Minimierung der Fehlerquadratsumme minimiert dieses Verfahren die Summe der Distanzen. Der PAM Ansatz rechnet zudem mit Medoids anstelle von Zentroiden. Unter dem Begriff "Medoid" versteht man ein repräsentatives Objekt eines Clusters, das zentral in

dem jeweiligen Cluster liegt. Das Verfahren lässt sich in zwei Phasen einteilen:

Build Phase
In der Build Phase sucht der Algorithmus nach geeigneten Startwerten für die Medoids. Dies geschieht durch sukzessive Auswahl repräsentativer Objekte. Das erste dieser Objekte minimiert die Summe aller Distanzen zu den anderen Objekten. In den nächsten Schritten wird jeweils ein weiteres repräsentatives Objekt gesucht. Dabei werden die Objekte gewählt, die am meisten zur Minimierung des oben erwähnten Zielkriteriums beitragen. Dieser Prozess wird solange wiederholt, bis g repräsentative Objekte, also die Anzahl entsprechend der gewünschten Clusteranzahl, gefunden sind. Diese g Objekte stellen dann die Startwerte für die zweite Phase dar.

Swap Phase
In dieser Phase wird versucht, den Satz der repräsentativen Objekte und dadurch auch die Klassifikation zu verbessern. Dazu werden alle Paare von Objekten (j,h) betrachtet, j steht dabei für die repräsentativen Objekte und h für die übrigen Objekte. Jedes Paar wird darauf untersucht, welche Auswirkung der Austausch der beiden Objekte auf das Zielkriterium hat. Im Falle einer Reduktion der Summe der Distanzen - und die damit verbundene Reduktion der Unähnlichkeiten - wird das Objekt h als repräsentatives Objekt gewählt. Dies wird mit allen möglichen Paarkombinationen (j,h) durchgeführt. Der Algorithmus bricht ab, sobald das Zielkriterium durch weitere Wechsel der Objekte nicht mehr verringert werden kann.

Das Ergebnis des Algorithmus ist dabei unbeeinflusst von der Anordnung der Objekte in der Distanzmatrix.
Als Distanzmaß können wieder unterschiedliche Metriken verwendet werden. Auch die Gower Distanz kann zur Berechnung der Distanz herangezogen werden, somit kann auch die kategoriale Variable berücksichtigt werden.

Ein Großteil der für diesen Abschnitt benötigten Informationen wurde aus [Kahn et al. 2001] entnommen. Hilfreich waren zudem [Fairmeir et. al 1996], [R Dokumentation(2) o.D], [Mirkes 2011], [Unesco o.D] und [Greutert 2004].

5.3 Modellbasierte Verfahren

Bei den hierarchischen Verfahren basiert die Einteilung in die Cluster auf die verschiedenen Distanzen zwischen den Objekten bzw. Clustern. Die partitionierenden Verfahren dagegen nahmen die Einteilung auf Grundlage eines Zielkriteriums (wie beispielsweise die Fehlerquadratsumme) vor. Beim modellbasierten Clustern wird angenommen, dass die G unterschiedlichen Gruppen der Daten jeweils einer eigenen Verteilung folgen. Die Idee des

Verfahrens ist es dann, jeder dieser Gruppen einer Likelihood zu unterstellen und diese anschließend zu maximieren. Es wird davon ausgegangen, dass die Beobachtungen $x_1, x_2, ..., x_n$ Realisationen eines Zufallsvektors $\boldsymbol{x}$ sind und $\boldsymbol{x}$ in jeder Klasse eine andere Verteilung besitzt. Die Klassenverteilungen sind dabei zwar bekannt, allerdings nicht die zugehörigen Parameter. Eine häufig getroffene Annahme ist dabei, dass die Beobachtungen in den Gruppen multivariat normalverteilt sind. Die jeweiligen Mittelwerte und Kovarianzmatrizen der Verteilungen in den Clustern unterscheiden sich untereinander. Somit ist $\boldsymbol{x}$ ein Gauss'sche Mischverteilungsmodell mit folgender Dichtefunktion:

$$f(\boldsymbol{x}) = \sum_{g=1}^{G} \psi_g f(\boldsymbol{x}|\mu_g, \Sigma_g)$$

bzw.

$$f(\boldsymbol{x}) = \sum_{g=1}^{G} \psi_g \frac{1}{(2\pi)^{\frac{n}{2}} (\det(\mathbf{\Sigma_g}))^{\frac{1}{2}}} \exp(-\frac{1}{2}(\mathbf{x} - \mu_\mathbf{g})^T \mathbf{\Sigma_g}^{-1}(\mathbf{x} - \mu_\mathbf{g}))$$

Die Klassenzugehörigkeit wird mit g ausgedrückt, dessen Wertebereich ($g = 1, .., G$) ist. Es wird dabei angenommen, dass die Beobachtungen $x_1, x_2, ..., x_n$ unabhängig aus der unbedingten Mischverteilung $f(\boldsymbol{x})$ stammen. $f(\boldsymbol{x}|\mu_g, \Sigma_g)$ ist die Dichtefunktion der multivariaten Normalverteilung $N(\mu_g, \Sigma_g)$.
ψ_g steht für die Wahrscheinlichkeit, dass die Beobachtung zur Gruppe g (von insgesamt G Gruppen) gehört. Hierfür gilt

$$\psi_g \in [0,1] \;\; für \;\; g = 1, ..., G-1, \;\; \sum_{g=1}^{G-1} \in [0,1]$$

Zu Beginn der Analyse liegt eine Stichprobe $y_1, ..., y_n$ vor. Diese wurde von X_{mix} produziert, deren Dichtefunktion obiger $f(\boldsymbol{x})$ entspricht.
ψ_G ergibt sich aus $1 - \sum_{g=1}^{G-1}$. Bevor mit der Klassifikation der Daten begonnen werden kann, müssen die unbekannten Parameter bestimmt werden. Ziel ist die Schätzung der unbekannten μ_g, Σ_g und ψ der Mischungsdichte f. Wir wollen also den Parameter

$$\theta = (\mu_1, ..., \mu_g, \Sigma_1, ..., \Sigma_g, \psi_1, ..., \psi_{g-1})$$

schätzen. Dazu wird der Maximum Likelihood Schätzer verwendet:

$$\hat{\theta} = \arg\max_{\theta} \;\; l(\theta, y_1, ..., y_n)$$

wobei der Log-Likelihood der Stichprobe dargestellt ist mit:

$$l(\theta, y_1, ..., y_n) = In(\prod_{i=1}^{n} \sum_{g=1}^{G} \psi_g f_g(y_i, \theta_g)) \quad \theta_i = (\mu_g, \Sigma_g) \in R^D \times R^{D \times D}$$

Die Komponentenanzahl g ist dabei vorerst festgesetzt und wird später mithilfe des BIC angepasst. Für die Schätzung der Parameter wird wieder der EM-Algorithmus verwendet. Um diesen im Mischmodell anwenden zu können, muss das Modell um versteckte Variablen z ergänzt werden. Der Ausgangspunkt für die Schätzung mit dem EM-Algorithmus ist ein Zufallsvektor T=(X,Z), wobei $X \in \mathbb{R}^{n_x}$ und $Z \in \mathbb{R}^{n_z}$, sowie ein Verteilungsmodell $f_T(t;\theta)$ mit unbekanntem θ. T ist nur in Form einer Teilstichprobe vorhanden, bestehend aus den beobachteten Variablen $(x_1, ..., x_N)$ und den erwähnten versteckten Variablen $(z_1, ..., z_N)$.

Es soll θ mithilfe der ML-Schätzung geschätzt werden. Aufgrund der nicht vorhandenen Werte z ist eine Optimierung der Gesamt-Log-Likelihood-Funktion nach $l(\theta, x, z)$ nach θ nicht möglich. Hier bietet der EM-Algorithmus durch die iterative Vorgehensweise eine Lösung, denn anstelle von $l(\theta, x, z)$ wird hier der bedingte Erwartungswert der Gesamt-Log-Likelihood-Funktion bezüglich f für die Schätzung $\theta^{(g)}$

$$Q(\theta, \theta^{(g)}) = E(l(\theta, X, Z)|X = x, \theta^{(g)})$$

berechnet. Die Schätzung $\theta^{(g)}$ wird anschließend als Funktion von θ maximiert. Zu Beginn wird wieder ein Startwert $\theta^{(0)}$ gesetzt, g wird gleich 0 gesetzt. Im E-Schritt wird

$$Q(\theta, \theta^{(g)}) = E(l(\theta, X, Z)|X = x, \theta^{(g)})$$

berechnet. Im M-Schritt berechnet man

$$\theta^{(g+1)} = \arg\max_{\theta \in \Theta} \; Q(\theta, \theta^{(g)})$$

und setzt anschließend $g = g + 1$. Der zweite und dritte Schritt wird dann bis zu bis zu einem bestimmten Abbruchkriterium wiederholt.

Beobachtet wurde die Stichprobe von X_{mix}, also $(y_1, ..., y_n)$. Die beobachteten Daten werden mit Z $(z_1, ..., z_N)$ erweitert, dabei gilt $z_i = (z_{i1}, ..., z_{iG})$ und

$$z_{ig} = \begin{cases} 1, & \text{wenn } z_{ig} \in g \\ 0, & \text{wenn } z_{ig} \notin g \end{cases}$$

Somit werden die vollständigen Daten durch die Stichprobe von $T = (X_{mix}, Z)$ dargestellt, also $([y_1, z_1], ..., [y_N, Z_N])$. Es wird angenommen, dass $Z_1, ..., Z_N$ i.i.d $M(1, \phi_1, ..., \phi_G)$ multinominal verteilt sind, dann erhält man die bedingte Dichte

$$f_{X_{mix}|Z=z_i}(y_i) = \prod_{g=1}^{G} f_i(y_i, \theta_i)^{z_{ig}}$$

Als Gesamt-Log-Likelihood-Funktion ergibt sich

$$l_{ges}(\theta, \psi, y, z) = \sum_{i=1}^{N} \sum_{g=1}^{G} z_{ig} \log(\psi_g f_g(y_i; \theta))$$

θ ist dabei $(\mu_1, ..., \mu_g, \Sigma_1, ..., \Sigma_G)$ und ψ gleich $\psi_1, ..., \psi_G$. Für den E-Schritt ergibt sich dann die allgemeine Formel

$$\hat{z}_{ig} = \frac{\hat{\psi}_g f_g(y_i, \hat{\theta}_g)}{\sum_{g=1}^{G} \hat{\psi}_g f_g(y_i, \hat{\theta}_g)}$$

Setzt man diese Werte für die fehlenden z_{ig} in die Gesamt-Log-Likelihood ein, erhält man den bedingten Erwartungswert $Q((\theta, \psi), (\hat{\theta}, \hat{\psi}))$. Im M-Schritt ergeben sich dann folgende Formeln:

$$\hat{\psi}_g = \frac{m_g}{N}$$

$$\hat{\mu}_g = \frac{\sum_{i=1}^{N} \hat{z_{ig}} y_i}{m_g}$$

$$m_g = \sum_{i=1}^{N} \hat{z}_{ig}$$

Die Schätzung von Σ_g hängt davon ab, wie die Matrix parametrisiert wurde. Jede Kovarianmatrix wird durch eine Eigenwertzerlegung parametrisiert:

$$\Sigma_g = \lambda_g D_g A_g D_g^T$$

λ_g stellt eine Konstante dar. D_g ist eine orthogonale Matrix der Eigenwerte von Σ_g und A_g ist eine Diagonalmatrix, dessen Elemente proportional zu den Eigenwerten Σ_g sind.

Dadurch erhält man je nachdem wie die Werte D_g,A_g und λ gewählt werden, verschiedene Modelle. Damit können Anforderung bezüglich des Volumens, der Form und der Orientierung gestellt werden. In Tabelle 4 sind mögliche Modelle dargestellt:

ID	Modell(Σ)	Verteilung	ID	Modell(Σ)	Verteilung
EII	λI	sphärisch	VEV	$\lambda_g D_g A D_g^T$	ellipsoid
VII	$\lambda_g I$	sphärisch	EEI	λA	diagonal
EEE	$\lambda D A D^T$	ellipsoid	VEI	$\lambda_g A$	diagonal
VVV	$\lambda_g D_g A_g D_g^T$	ellipsoid	EVI	λA_g	diagonal
EEV	$\lambda D_g A D_g^T$	ellipsoid	VVI	$\lambda_g A_g$	diagonal

Tabelle 4: Mögliche Modelle des modellbasierten Verfahrens

Die ID der Tabelle beschreibt die Modelle dabei genauer. Sie besteht aus 3 Buchstaben, der erste Buchstabe bezeichnet das Volumen, der zweite die Form und der dritte die Orientierung der Modelle. Es gibt zwei verschiedene Möglichkeiten des Volumens, entweder alle Volumen sind gleich ($\lambda_g = \lambda \ \forall g$ entspricht E) oder sie unterscheiden sich (V). Die Formen der Cluster können gleich sein und zusätzlich einer Sphäre entsprechen ($A_g = I \ \forall g$, entspricht I), nur gleich sein (entspricht E) oder variable Formen (von A_g, entspricht V) annehmen. Bei der Orientierung kann sie für alle Cluster gleich sein ($D_g = D \ \forall g$, entspricht E), zusätzlich zur Gleichheit den Koordinatenachsen entsprechen ($D_k = I \ \forall g$, entspricht I) oder andernfalls beliebige Orientierungen besitzen (entspricht V).

Es existieren also mehrere verschiedene Modelle. Diese werden mithilfe des EM-Algorithmus angepasst. Doch hier stellt sich die Frage, welches Modell nun für die Clusteranalyse verwendet werden sollte. Gegeben sind also unterschiedliche Modelle $M_1, ..., M_L (l = 1, ..., L)$ mit den a-priori-Wahrscheinlichkeiten $Pr(M_1), ..., (M_L)$ und die Daten D, dann gilt nach Bayes:

$$Pr(M_l|D) = Pr(D|M_l)\frac{Pr(M_l)}{Pr(D)} \propto Pr(D|M_l)Pr(M_l)$$

$Pr(M_l|D)$ sind die a-posteriori-Wahrscheinlichkeiten. Die integrierte Likelihood des Modells M_k stellt sich durch $Pr(D|M_l)$ dar:

$$Pr(D|M_l) = \int Pr(D|\theta_g, M_l)Pr(\theta_l, M_l)d\theta_l$$

$Pr(\theta_l|M_l)$ ist die a-priori-Verteilung des Parametervektors θ_l des Modells M_l. Daraus lässt sich mit

$$2\log p(D|M_l) \approx 2\log p(D|\hat{\theta}_l, M_l) - v_l \log(n) = BIC_l$$

das BIC (Bayes Information Criterion) berechnen. v_l ist die Anzahl der unabhängigen Parameter, die im Modell M_l geschätzt werden. Man erhält verschiedene Modelle mit verschiedenen BIC Werten. Davon wird das Modell ausgewählt, das den größten BIC besitzt.

Dieses Verfahren eignet sich gut, wenn den einzelnen Gruppen unterschiedliche Verteilungen zugrunde liegen. Zusätzlich hat es den Vorteil, dass durch den modellbasierten Ansatz besser mit Überlappungen umgehen werden kann. Bei starken Überlappungen ist das modellbasierte Verfahren deutlich effizienter. Durch das Modellwahlverfahren wird zudem die Anzahl und Form der Cluster bestimmt, weshalb keine Annahme über die Clusteranzahl nötig ist. Anderseits basiert dieser Ansatz auf die Verteilungsannahmen. Falls diese nicht korrekt sind, sind auch die gelieferten Ergebnisse zumindest fragwürdig.

Ein Großteil dieses Abschnittes wurde inhaltlich von [Alexandrovich 2011] übernommen. Zusätzlich stammen Informationen aus [Oellinger 2010],[Greutert 2004], [Fraley et al. 2007],[Fraley et al. 2012] sowie [Unbekannt 2010].

5.4 Umsetzung in R

In R existieren zahlreiche Pakete, um die verschiedenen Clusterverfahren durchzuführen.Bevor die jeweiligen Datensätze den Clustermethoden in R umgeben werden, wird der Datensatz standardisiert. Damit wird sichergestellt, dass die Größenordnung der Variablen keine Bedeutung hat. Ansonsten würden Variablen mit numerisch großen Werten die Clusteranalyse stärker beeinflussen als Variablen mit kleineren Werten.

Wie in Kapitel 5 schon beschrieben wurde, können nicht alle Verfahren mit Datensätzen, in denen metrische und kategoriale Variablen vermischt sind, umgehen. Aus diesem Grund werden für diese Verfahren die jeweiligen Datensätze (bwz. Amelia Datensätze) ohne die kategoriale Variable betrachtet. Bei der Berechnung der Distanzen für die hierarchische Clusteranalyse wird die *dist* Funktion verwendet. Damit können die Distanzmatrizen auf Grundlage verschiedener Distanzmaße berechnet werden. Mit der Funktion *hclust* wird anschließend mit den verschiedenen Linkage Methoden die Clusteranalyse durchgeführt. Schließlich verwendet man die Funktion *cuttree*, um ein Vektor mit den Clusterzuordnungen zu erhalten. Dieser Vektor besitzt die selbe Länge wie die Anzahl der Objekte im Datensatz, also n. Für die *cuttree* Funktion muss auch die Anzahl der Cluster übergeben werden. Somit erhalten wir mit den verschiedenen Linkage-Methoden (Single, Average, Complete, Centroid und Ward) sowie den verschiedenen Distanzmaßen (Euklid, Maximum, Manhattan) insgesamt 15 verschiedene Verfahren für das agglomerative Clustern. Sowohl die *hclust* als auch die *dist* Funktion befindet sich im *stats* Paket.

Ähnlich verläuft die Berechnung der divisiven Verfahren. Hier wird lediglich die *diana* Funktion aus dem *cluster* Paket benötigt. Da die divisive Clusteranalyse keine unterschiedlichen Linkage Methoden verwendet, erfolgt die Berechnung auf Grundlage der verschiedenen Distanzen. So ergeben sich hier insgesamt 3 verschiedene Verfahren. Auch hier wird wieder die *cuttree* Funktion verwendet, um die Vektoren mit den Gruppenzuordnungen zu erhalten.

Für die Durchführung des K-means Verfahren wird die Funktion *kmeans* aus dem *stats* Paket verwendet. Für die anderen beiden partitionierenden Verfahren wird *pam* aus dem *cluster* Paket verwendet. Im Output (in Form einer Liste), den man mit der *kmeans* bzw. der *pam* Funktion erhält, sind

auch die Gruppenzuordnungen gespeichert. Der Vektor mit den Clusterzuordnungen muss also lediglich extrahiert werden.

Für die Durchführung des modellbasierten Verfahrens wird die *Mclust* Funktion aus dem gleichnamigen Paket verwendet. Auch hier sind die Clusterzuordnungen ähnlich wie beim K-means Verfahren im Output gespeichert und müssen extrahiert werden.

Schließlich gibt es die Verfahren, die mit Datensätzen, welche sowohl metrische als auch kategoriale Variablen enthalten, umgehen können. Bei all diesen Verfahren wird als Distanzmaß die Gower Distanz sowie die jeweilig benötigten Datensätze (bzw. Amelia Datensätze) verwendet. Die Distanzmatrix mit der Gower Metrik lässt sich in R mit der *daisy* Funktion aus dem *cluster* Paket berechnen. Das partitionierende Verfahren auf Basis der Gower Distanz wird mit *pam* berechnet. Somit ergeben sich 7 weitere Verfahren.
Bei allen Verfahren wird die Anzahl der Cluster auf 3 (bzw. 4 für Datensatz 5) festgesetzt.
Ein Problem, dass bei allen Vektoren der Clusterzuordnungen berücksichtigt werden muss, sind die unterschiedlichen levels der Gruppen. Das Problem lässt sich am besten anhand eines kleinen Beispiels erläutern: Ein Datensatz besteht aus insgesamt 10 Objekten, wobei die ersten 5 Objekte aus Gruppe 1 stammen und die restlichen aus Gruppe 2. Die Clusterzuordnungen eines beliebigen Verfahrens bezeichnet allerdings die ersten 5 Objekte als Cluster 2 und die übrigen als Cluster 1. Es wird also die richtige Zuordnung der Cluster erreicht, allerdings werden die Objekte aus Gruppe 1 als Cluster 2 bezeichnet und umgekehrt. Das heißt, die Cluster besitzen andere levels als die Gruppen, obwohl sie dieselben Objekte umfassen. Dies tritt in analoger Form bei den verschiedenen Verfahren in Datensätzen 1 bis 5 auf. Die Lösung des Problems ist die "Zuordnungsfunktion" (siehe R-Code). Es werden sämtliche Permutationen der level Bezeichnungen gebildet und anschließend mit den tatsächlichen Gruppenbezeichnungen verglichen. Man erhält also verschiedene Clusterzuordnungsvektoren mit unterschiedlichen levels und vergleicht diese mit dem echten Gruppenvektor. Die level Bezeichnung, die die größte Übereinstimmung erreicht, wird anschließend verwendet. Das bedeutet also, dass die Zuordnungsfunktion bei jedem der Clusterzuordnungsvektoren angewendet werden sollte, um sie mit der "echten" Zuordnung vergleichen zu können. Zusätzlich zu dem Clustervektor mit den richtigen levels gibt die Zuordnungsfunktion im Output die maximale Anzahl an Übereinstimmungen mit dem "echten" Gruppenvektor wieder. Für die Analyse der Amelia Datensätze sowie für Datensatz 5 muss die Zuordnungsfunktion leicht modifiziert werden.

Insgesamt erhält man 29 verschiedene Verfahren zur Clusterung (vgl. Tabelle

5). Verfahren 1 bis 15 umfassen die verschiedenen agglomerativen Verfahren mit verschiedenen Distanzmaßen und Linkage Methoden. Die Verfahren 16 bis 18 sind die unterschiedlichen divisiven Verfahren mit unterschiedlichen Distanzmaßen. Die verschiedenen partitionierenden Verfahren sind mit Verfahren 19 bis 21 gekennzeichnet. Das modellbasierte Verfahren entspricht Verfahren 22. Die Verfahren mit der Gower Distanz und auf Basis der Datensätze mit kategorialer Variable sind schließlich die Verfahren 23 bis 29. Die Outputs der verschiedenen Verfahren werden anschließend in einer Liste gespeichert. Aus den verschiedenen Vektoren der Clusterzuordnungen (nach Anwendung der Zuordnungsfunktion) wird eine $(n * 29)$ Matrix gebildet, die alle verschiedenen Clusterzuordnungen enthält. Schließlich wird für jedes Verfahren die prozentuale Genauigkeit ermittelt, also wie gut das Verfahren die Objekte in die Gruppen einteilt. Diese Genauigkeit erhält man, indem man die maximale Anzahl an Übereinstimmungen (aus dem Output der Zuordnungsfunktion) durch die Anzahl der Objekte teilt. Die Genauigkeiten werden in einem Vektor gespeichert. Die exakte Durchführung sowie die selbst erstellten Funktionen sind im R-Code ab Seite 105 bzw. Seite 118 zu finden.

Hierarchische Verfahren	
Verfahren 1	agglomerativ (Euklid/Single)
Verfahren 2	agglomerativ (Maximum/Single)
Verfahren 3	agglomerativ (Manhattan/Single)
Verfahren 4	agglomerativ (Euklid/Complete)
Verfahren 5	agglomerativ (Maximum/Complete)
Verfahren 6	agglomerativ (Manhattan/Complete)
Verfahren 7	agglomerativ (Euklid/Average)
Verfahren 8	agglomerativ (Maximum/Average)
Verfahren 9	agglomerativ (Manhattan/Average)
Verfahren 10	agglomerativ (Euklid/Centroid)
Verfahren 11	agglomerativ (Maximum/Centroid)
Verfahren 12	agglomerativ (Manhattan/Centroid)
Verfahren 13	agglomerativ (Euklid/Ward)
Verfahren 14	agglomerativ (Maximum/Ward)
Verfahren 15	agglomerativ (Manhattan/Ward)
Verfahren 16	divisiv (Euklid)
Verfahren 17	divisiv (maximum)
Verfahren 18	divisiv (Manhattan)
Partitionierende Verfahren	
Verfahren 19	K-means
Verfahren 20	K-medoids (Euklid)
Verfahren 21	K-medoids (Manhattan)
Modellbasiert	
Verfahren 22	Modellbasiert
mit Gower Distanz	
Verfahren 23	agglomerativ (Gower/Single)
Verfahren 24	agglomerativ (Gower/Complete)
Verfahren 25	agglomerativ (Gower/Average)
Verfahren 26	agglomerativ (Gower/Centroid)
Verfahren 27	agglomerativ (Gower/Ward)
Verfahren 28	divisiv (Gower)
Verfahren 29	K-medoids (Gower)

Tabelle 5: Übersicht über die verwendeten Clusterverfahren

6 Analyse

6.1 Umsetzung

In diesem Abschnitt soll gezeigt werden, wie das Verhalten verschiedener Clusterverfahren nach Imputation fehlender Daten untersucht wird. Es wird das Vorgehen sowie die praktische Umsetzung in R beschrieben. Am Ende werden die resultierenden Ergebnisse vorgestellt.

6.1.1 Vorgehen

Um analysieren zu können, wie sich verschiedene Clusterverfahren verhalten, wenn fehlende Daten imputiert wurden, muss ebenfalls untersucht werden, wie sich die Clusterverfahren vor der Löschung und anschließenden Imputation der fehlenden Daten verhalten haben. Man untersucht also mit den verschiedenen Clustermethoden nicht nur die 5 durch multiple Imputation erzeugten Amelia Datensätze, sondern auch den ursprünglichen Datensatz[5]. In Kapitel 4.2.3 wurde bereits beschrieben, wie die Ergebnisse der einzelnen Amelia Datensätze zu einem Gesamtergebnis kombiniert werden können. Dieses Gesamtergebnis spiegelt sozusagen die Ergebnisse der Clusterverfahren nach der Imputation wieder. Dies muss nun mit dem Ergebnis der Clusteranalyse des ursprünglichen Datensatzes verglichen werden. Den Ausgangspunkt dazu stellen die jeweiligen Tabellen dar, die zählen, wie oft ein ein Merkmalsträger über die 5 Amelia Datensätze richtig eingeteilt wurde. Zusätzlich werden nur noch diejenigen Merkmalsträger betrachtet, die in der Clusteranalyse mit den ursprünglichen Datensatz richtig bzw. falsch eingeteilt wurden und stellt ebenfalls die Anzahl dar, wie oft sie über die Amelia Datensätze richtig klassifiziert wurden. Schließlich werden nur noch diejenigen Zeilen betrachtet, die imputiert wurden, auch jeweils wieder unterteilt in ursprünglich richtig bzw. ursprünglich falsch geclustert. Als Ergebnis erhält man Vergleichstabelle 1:

Wie oft wurde richtig geclustert?:	Gesamtanzahl	0x	1x	2x	3x	4x	5x
Gesamt	5000	54	59	67	119	304	4397
–davon ursprünglich Richtig geclustert	4990	51	56	67	118	303	4395
–davon ursprünglich Falsch geclustert	10	3	3	0	1	1	2
Imputiert	3252	53	59	66	118	304	2652
–davon ursprünglich Richtig geclustert	3245	51	56	66	118	303	2651
–davon ursprünglich Falsch geclustert	7	2	3	0	0	1	1

Tabelle 6: Beispiel einer Vergleichstabelle 1

Zusammengefasst, ähnlich wie in Kapitel 4.2.3 beschrieben, resultiert Vergleichstabelle 2:

[5] Als ursprünglicher Datensatz werden die Datensätze 1 bis 5 bezeichnet, bevor fehlende Werte erzeugt und wieder imputiert wurden.

Wie oft wurde richtig geclustert?:	Gesamtanzahl.davon	Falsch	Richtig
Gesamt	5000	180	4820
–davon ursprünglich Richtig geclustert	4990	174	4816
–davon ursprünglich Falsch geclustert	10	6	4
Imputiert	3252	178	3074
–davon ursprünglich Richtig geclustert	3245	173	3072
–davon ursprünglich Falsch geclustert	7	5	2

Tabelle 7: Beispiel einer Vergleichstabelle 2

Die als Kombination der Ergebnisse erhaltenen Amelia Datensätze werden also erweitert bzw. genauer betrachtet, indem das Ergebnis mit dem Ergebnis des ursprünglichen Datensatzes verglichen wird. Anhand dieser erweiterten Tabelle kann nun die Auswirkung der Imputation auf die Clusterverfahren untersucht werden. Interessant ist dabei die Betrachtung der Objekte, die ursprünglich genauso eingeteilt wurden wie nach der Imputation. Ist beispielsweise die Anzahl der im ursprünglichen Datensatz richtig (bzw. falsch) geclusterten Objekte gleich der Anzahl der nach der Imputation richtig (bzw. falsch) geclusterten Objekte, so hätte die Imputation keinen Einfluss, da die Objekte genau gleich geclustert werden. Um die Auswirkungen der Imputation auf einen Blick erfassen zu können, wurden Analysetabellen erstellt, die folgende Form besitzen:

	Zeilen, davon	Clusterung wie ursprünglich	Übereinstimmung
imputiert	3252	3077	0.95
nicht imputiert	1748	1745	1.00

Tabelle 8: Beispiel einer Analysetabelle

In dieser Tabelle ist die Übereinstimmung (gerundet auf zwei Stellen nach dem Komma) über die Clustereinteilung zwischen dem ursprünglichen Datensatz sowie dem Gesamtergebnis der imputierten Datensätze dargestellt. Es wird unterschieden, ob fehlende Werte in den Zeilen imputiert wurden oder nicht. Je höher die Übereinstimmung, desto weniger Einfluss hat die Imputation. In dieser Analysetabelle hätte die Imputation nur einen geringen Einfluss, da die nicht imputierten Zeilen nahezu gleich geclustert werden und die imputierten Zeilen eine sehr hohe Übereinstimmung aufweisen. Zusätzlich werden noch die Genauigkeiten (ebenfalls auf zwei Stellen nach dem Komma gerundet) der einzelnen Amelia Datensätze sowie die Durchschnittsgenauigkeit,die Ursprungsgenauigkeit und die Genauigkeit abgeleitet aus Vergleichstabelle 2 (Richtig/Gesamtanzahl) in Form einer weiteren Tabelle(Genauigkeitstabelle) dargestellt:

	Urspr.	AD.1	AD.2	AD.3	AD.4	AD5	Durchschnittlich	aus.Tabelle
Genauigkeit	1.00	0.86	0.92	0.97	0.92	0.64	0.88	0.94

Tabelle 9: Beispiel einer Genauigkeitstabelle

Für jedes Verfahren resultieren also verschiedene Tabellen, die die Aus-

wirkung der Imputation auf die Clusteranalyse repräsentieren sollen. Am aussagekräftigsten für die Analyse eines Verfahrens sind die Übereinstimmungen aus der Analysetabelle sowie die Genauigkeiten vor und nach der Imputation. Die Genauigkeit nach der Imputation wird dabei von der aus der Vergleichstabelle 2 abgeleiteten Genauigkeit widergespiegelt. Die Genauigkeit vor der Imputation wird von der ursprünglichen Genauigkeit wiedergegeben.

6.1.2 Umsetzung in R

Der Ausgangspunkt zur Erstellung der Vergleichstabellen stellen die aus der Clustern-Funktion gewonnenen Gruppenmatrizen dar, die die Gruppenvektoren der verschiedenen Clusterverfahren enthalten. Sowohl für den ursprünglichen Datensatz als auch für die Amelia Datensätze werden die Gruppenmatrizen benötigt und dementsprechend mit der Clustern Funktion ermittelt. Die Gruppenmatrizen enthalten spaltenweise die Gruppenvektoren der einzelnen Verfahren. Die erste Spalte der Gruppenmatrix enthält also den Gruppenvektor des Verfahren 1, die zweite Spalte die des Verfahrens 2 usw. . Zur Untersuchung eines Verfahrens (bzw. zur Erstellung der Tabellen) wählt man also jeweils die Spalten der Gruppenmatrizen aus, die das zu untersuchende Verfahren repräsentieren. Daraus wird eine neue Matrix (Vergleichsmatrix) gebildet. Diese Matrix stellt die Gruppenvektoren der Amelia Datensätze für ein bestimmtes Verfahren dar und enthält dementsprechend 5 Spalten (für jeden Amelia Datensatz) und 5000 Zeilen (Anzahl Merkmalsträger). Somit enthält die Matrix zeilenweise die Clusterzuordnungen der einzelnen Merkmalsträger über die 5 Amelia Datensätze. Anschließend wird mithilfe der RSF-Funktion (vgl. R-Code) für jede Zeile bestimmt, wie oft die Clusterzuordnung der entsprechenden Zeile der tätsachliche Gruppe entspricht. So ergibt sich die erste Zeile für die Vergleichstabellen 1 (vgl.Tabelle 6).Für die restlichen Spalten der Tabellen wird jeweils eine reduzierte Vergleichsmatrix verwendet und die Gruppenmatrix des ursprünglichen, kompletten Datensatzes benötigt. Je nachdem welches Verfahren gerade untersucht, wählt man die entsprechende Spalte der ursprünglichen Gruppenmatrix aus, vergleicht diesen Gruppenvektor mit dem “echten” Gruppenvektor und wählt dann lediglich die Zeilen aus, die richtig (Zeile 2 der Tabelle) bzw. falsch (Zeile 3 der Tabelle) geclustert wurden. Diese Zeilen wählt man anschließend auch bei der Vergleichsmatrix aus und führt die RSF-Funktion durch. Um Spalte 4 zu erhalten, die nur die imputierten Zeilen darstellen, muss mithilfe des Datensatz mit fehlenden Werten ermittelt werden, welche Zeilen imputiert wurden. Ausschließlich diese Zeilen werden anschließend auch bei der Vergleichsmatrix ausgewählt und erneut die Anzahl der richtigen Zuordnungen mit der RSF gezählt. Um die letzten beiden Zeilen der Tabelle zu erhalten, werden nur die Zeilen der Vergleichsmatrix ausgewählt, die ursprünglich richtig bzw. falsch geclustert und

imputiert wurden. Auf diese reduzierten Vergleichsmatrizen wird wieder die RSF Funktion angewendet und es ergibt sich schließlich Zeile 4 und 5 der Tabelle. Die Gesamtanzahl (vgl. Spalte 1 der Tabelle) der Merkmalsträger die in der jeweiligen Zeile untersucht werden, entspricht der Zeilenanzahl der jeweils betrachteten Vergleichsmatrix. Um die Vergleichstabellen 2 zu erhalten, wird die Summe der Spalten 2 bis 4 und die Summe 5 bis 7 aus der zugehörigen Vergleichstabelle 1 gebildet.

Die ursprüngliche Genauigkeit in der Genauigkeitstabelle lässt sich aus dem Clusterobjekt des ursprünglichen Datensatzes entnehmen. Die Genauigkeiten der Elemente 2 bis 6 der Genauigkeiten erhält man aus den Objekten der Amelia Datensätze, die mit der Clusternfunktion erzeugt wurden. Die siebte Genauigkeit ist der Durchschnittswert der Genauigkeiten der Amelia Datensätze. Die letzte Genauigkeit ergibt sich wie beschrieben aus Vergleichstabelle 2. Die genaue Umsetzung ist unter dem Namen “Analysefunktion” (bzw “Analysefunktion 2” für Datensatz 5) im R-Code ab Seite 110 bzw. Seite 127 zu finden.

6.2 Ergebnisse

In diesem Kapitel werden schließlich die Ergebnisse der Analysen für jedes Clusterverfahren gezeigt. Dabei werden die Resultate in 4 Abschnitte eingeteilt. Im ersten Abschnitt werden die Ergebnisse der hierarchischen Verfahren mit dem Datensatz ohne dem kategorialen Merkmal beschrieben. Dies ist notwendig, da die Verfahren 1 bis 18 nicht mit der Mischung aus metrischen und kategorialen Variablen umgehen können. Ebenso verhält es sich mit den Verfahren 19 bis 21 und Verfahren 22, dessen Ergebnisse im Abschnitt “Partitionierende Verfahren”bzw. “Modellbasiertes Verfahren”beschrieben werden. Schließlich wird im letzten Abschnitt der Datensatz mit der kategorialen Variable betrachtet, da die Verfahren 23 bis 29 auch mit gemischt skalierten Merkmalen eine Clusteranalyse durchführen können.

6.2.1 Hierarchische Verfahren

Zu Beginn werden die Ergebnisse der hierarchischen Verfahren (ohne Gower Distanz) dargestellt. Diese unterscheiden sich wiederum in agglomerative und divisive Verfahren.

Agglomerative Verfahren

Zuerst werden die Ergebnisse der agglomerativen Verfahren mit verschiedenen Linkage Funktionen beschrieben.

Single Linkage

Das agglomerative Verfahren mit dem Single Linkage liefert bei jeden der untersuchten Datensätze nur sehr ungenaue Clusterzuordnungen, es werden nahezu alle Merkmalsträger demselben Cluster zugeordnet. Hier zeigt sich deutlich, dass sich das Verfahren für diese Datensätze aufgrund der Verkettungseigenschaft nicht zur Klassifikation eignet. Auch die Verwendung der verschiedenen Metriken ändert nichts an der ungenaugen Klassifikation, aus diesem Grund kann auf eine getrennte Analyse nach Metriken beim Single Linkage verzichtet werden.
Selbst in Datensatz 1, indem nur geringe Überschneidungen bestehen, misslingt schon eine einigermaßen korrekte Klassifikation. Deutlich wird dies auch im zugehörigen Dendrogramm zu Datensatz 1 (in diesem Fall mit euklidischer Metrik):

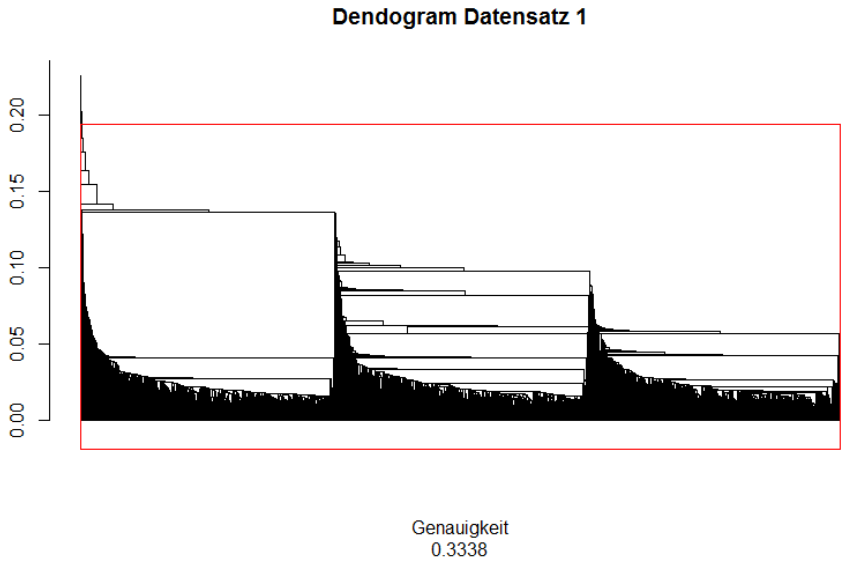

Abbildung 19: Dendrogramm zu Datensatz 1 mit Single Linkage und euklidischer Metrik

In dem Dendrogramm wird deutlich, dass einzelne Objekte erst spät fusioniert werden. Das hat zur Folge, dass bei der Einteilung des Dendrogramms in 3 Gruppen (symbolisiert durch rote Vierecke in Abbildung 19) nur eine große Gruppe erkennbar ist. So müssten in dem Dendrogramm eigentlich drei verschiedene rote Rechtecke zu erkennen sein (ähnlich wie in 18 auf Seite 51), die jeweils einen Cluster umfassen. Aufgrund der sehr kleinen anderen Gruppen ist nur ein rotes Rechteck zu erkennen. Die Dendrogramme zu den anderen Datensätzen sowie die der imputierten Datensätze haben

ähnliche Form wie das Dendrogram in Abbildung 19. Die Imputation scheint auf das Ergebnis der Clusteranalyse keinen großen Einfluss zu haben. So werden die Merkmalsträger, die ursprünglich falsch geclustert wurden, auch nach der Imputation größtenteils falsch geclustert und umgekehrt. Ersichtlich wird dies u.a in den Vergleichstabellen 2 der Datensätze. Zur Demonstration wird die Vergleichstabellen 2 und die Analysetabelle von Datensatz 1 dargestellt (mit Manhattan Metrik):

Wie oft wurde richtig geclustert?:	Gesamtanzahl.davon	Falsch	Richtig
Gesamt	5000	3333	1667
–davon ursprünglich Richtig geclustert	1669	2	1667
–davon ursprünglich Falsch geclustert	3331	3331	0
Imputiert	3000	1636	1364
–davon ursprünglich Richtig geclustert	1365	1	1364
–davon ursprünglich Falsch geclustert	1635	1635	0

Tabelle 10: Vergleichstabelle 2 zu Datensatz 1 mit Single Linkage und Manhattan Metrik

	Zeilen..davon	Clusterung.wie.ursprünglich	X..Übereinstimmung
imputiert	3000	2999	1.00
nicht imputiert	2000	1999	1.00

Tabelle 11: Analysetabelle zu Datensatz 1 mit Single Linkage und Manhattan Metrik

In beiden Tabellen wird deutlich, dass alle Merkmalsträger, unabhängig davon ob fehlende Werte imputiert wurden oder nicht, nahezu exakt so geclustert worden wie zuvor im ursprünglichen Datensatz. Dies bestätigt sich auch in Anbetracht der Übereinstimmungen der anderen Datensätze (vgl. Tabelle 12):

Single/Euklid	D1	D2	D3	D4	D5
imputiert	1.00	1.00	1.00	1.00	1.00
nicht imputiert	1.00	1.00	1.00	1.00	1.00

Single/Maximum	D1	D2	D3	D4	D5
imputiert	1.00	1.00	1.00	1.00	1.00
nicht imputiert	1.00	1.00	1.00	1.00	1.00

Single/Manhattan	D1	D2	D3	D4	D5
imputiert	1.00	1.00	1.00	1.00	1.00
nicht imputiert	1.00	1.00	1.00	1.00	1.00

Tabelle 12: Übersicht über Übereinstimmungen (Single Linkage)

Aus diesem Grund gibt es auch keine Unterschiede bei den Genauigkeiten vor und nach der Imputation. Die Imputation scheint also beim agglomerativen Verfahren mit dem Single Linkage in diesen Datensätzen keinen nennenswerten Einfluss zu haben. Dies liegt aber hauptsächlich daran, dass die Clusteranalyse mit dem Single Linkage nicht effektiv ist und nahezu alle Merkmalsträger in eine einzige Gruppe geclustert werden.

Complete Linkage

Im Vergleich zum Single Linkage Verfahren liefert das Complete Linkage Verfahren eine genauere Klassifikation der Merkmalsträger. Allerdings wird nur bei Datensatz 1 eine hinreichend gute Klassifikation erreicht. Zwischen den einzelnen Metriken gibt es hier kleinere Unterschiede.
Die Imputation fehlender Daten erweist sich hier deutlich einflussreicher als beim Single Linkage. Zwischen den einzelnen Amelia Datensätze kommt es zu Schwankungen der Klassifikationen. Deutlich wird dies beim Betrachten der Dendrogramme (beispielsweise Datensatz 1):

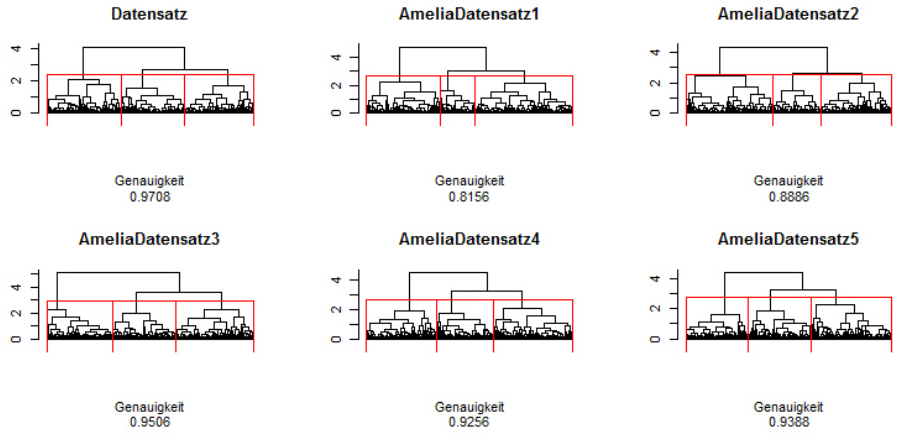

Abbildung 20: Dendrogramme zu Datensatz 1 (Complete/Maximum) und zugehörigen Amelia Datensätzen

Als Beispiel sind die Dendrogramme des Verfahrens mit der Maximum Metrik von Datensatz 1 und den zugehörigen Amelia Datensätzen dargestellt. Man stellt beim Betrachten fest, dass zwischen den einzelnen Dendrogrammen deutliche Unterschiede bestehen, die Genauigkeiten weisen ebenfalls auf eine starke Schwankung des Clusterergebnisses hin. Das Ergebnis der Clusteranalyse scheint also stark von der Imputation abhängig zu sein, da bei jeder der verschiedenen Imputationen (die durch die Amelia Datensätze repräsentiert werden) andere Ergebnisse resultieren. Aufgrund der

Unterschiede zwischen den einzelnen Imputationen wird nochmal deutlich, dass bei singulären Imputationsmethoden Unsicherheit über das Ergebnis der Clusteranalyse vorliegen würde. Durch mehrere Imputationen und die anschließende Kombination der einzelnen Ergebnisse würde diese Unsicherheit zumindest reduziert werden. Aus diesem Grund scheint sich die multiple Imputation in Anbetracht fehlender Daten bei Clusteranalysen also besser zu eignen.

Betrachtet man wieder die Übersicht über die Übereinstimmungen der verschiedenen Datensätze (siehe Tabelle 13), so stellt man fest, dass auch Merkmalsträger, in denen keine Beobachtungen imputiert worden, nach der Imputation anders geclustert werden wie im ursprünglichen Datensatz. Dies hat seine Ursache in der Tatsache, dass durch die Imputation neue Werte und dadurch andere Distanzen berechnet werden und sich dementsprechend auch die Cluster anders zusammengefasst werden. So lassen sich auch die Schwankungen in den Amelia Datensätzen erklären. Die Übereinstimmung ist dabei in den nicht imputierten Zeilen durchweg höher als in den imputierten Zeilen.

Complete/Euklid	D1	D2	D3	D4	D5
imputiert	0.94	0.62	0.59	0.68	0.63
nicht imputiert	0.94	0.91	0.78	0.81	0.66

Complete/Maximum	D1	D2	D3	D4	D5
imputiert	0.91	0.64	0.51	0.77	0.71
nicht imputiert	0.97	0.72	0.61	0.94	0.83

Complete/Manhattan	D1	D2	D3	D4	D5
imputiert	0.84	0.56	0.52	0.66	0.67
nicht imputiert	0.93	0.79	0.53	0.76	0.83

Tabelle 13: Übersicht über Übereinstimmungen (Complete Linkage)

Die Übereinstimmungen unterscheiden sich zwar zwischen den einzelnen Datensätzen und Metriken, allerdings kann keine eindeutige Tendenz dafür gegeben werden, wie genau sich die jeweiligen Ergebnisse der Clusterverfahren bei Imputation verändern. Aufgrund der Veränderungen der Übereinstimmung nach der Imputation kann davon ausgegangen werden, dass sich auch die Genauigkeit der Clusteranalyse nach der Imputation verändert. Aus diesem Grund sind in Tabelle 14 die Genauigkeiten vor und nach der Imputation dargestellt:

Complete/Euklid	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.80	0.58	0.64	0.86
Genauigkeit nach Imputation	0.94	0.61	0.42	0.49	0.55

Complete/Maximum	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.97	0.64	0.71	0.49	0.63
Genauigkeit nach Imputation	0.96	0.40	0.33	0.40	0.54

Complete/Manhattan	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.95	0.70	0.66	0.66	0.65
Genauigkeit nach Imputation	0.92	0.57	0.39	0.41	0.48

Tabelle 14: Veränderung der Genauigkeiten (Complete Linkage)

Average Linkage

Während mit dem Complete Linkage nur Datensatz 1 durchgehend gut geclustert wurde, erreicht man mit dem Average Linkage auch teilweise gute Clusterergebnisse für die anderen Datensätze. Einen ersten Anhaltspunkt zur Auswirkung der Imputation bei diesen Verfahren gibt wieder die Übersicht der Übereinstimmungen sowie die Veränderung der Genauigkeiten nach der Imputation:

Average/Euklid	D1	D2	D3	D4	D5
imputiert	0.97	0.67	0.81	0.78	0.65
nicht imputiert	1.00	0.72	0.80	0.70	0.61

Average/Maximum	D1	D2	D3	D4	D5
imputiert	0.91	0.68	0.80	0.86	0.56
nicht imputiert	0.98	0.71	0.80	0.96	0.63

Average/Manhattan	D1	D2	D3	D4	D5
imputiert	0.96	0.35	0.99	0.71	0.69
nicht imputiert	0.99	0.45	1.00	0.65	0.66

Tabelle 15: Übersicht über Übereinstimmungen (Average Linkage)

Average/Euklid	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.69	0.82	0.92	0.92
Genauigkeit nach Imputation	0.98	0.40	0.68	0.71	0.59

Average/Maximum	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.67	0.82	0.74	0.92
Genauigkeit nach Imputation	0.93	0.40	0.68	0.70	0.53

Average/Manhattan	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.92	0.68	0.92	0.89
Genauigkeit nach Imputation	0.97	0.40	0.68	0.63	0.59

Tabelle 16: Veränderung der Genauigkeiten (Average Linkage)

Beim Betrachten von Tabelle 16 fällt auf, dass bei der Euklid und bei der Manhattan Metrik die Übereinstimmungen in den Datensätzen 4 und 5 der imputierten Zeilen höher sind als die der nicht imputierten Zeilen. Auffällig ist zudem, das die Imputation bei dem Verfahren mit der Manhattan Metrik bei Datensatz 3 keinen Einfluss zu haben scheint. Dies hat allerdings ähnliche Ursache wie beim Clustern mit dem Single Linkage und zwar, dass nahezu alle Merkmalsträger sowohl vor als auch nach der Imputation einem einzigen Cluster zugeordnet werden.

Die Genauigkeiten der Clusteranalyse nehmen wieder bis auf diese Ausnahme ab. Generell kann wieder keine allgemeine Tendenz über die Auswirkung der Imputation gegeben werden, da sie wieder unterschiedlichen Einfluss je nach Datensatz hat. Erwähnenswert ist zudem, dass die Genauigkeiten nach der Imputation teilweise deutlich schlechter sind. Dies fällt besonders bei Datensatz 5 auf. In Abbildung 21 sind Plots von Datensatz 5 und den zugehörigen Amelia Datensätzen abgebildet. Zusätzlich wurden farblich die Gruppen eingezeichnet, wie sie von dem Average Linkage mit euklidischer Metrik gefunden werden.

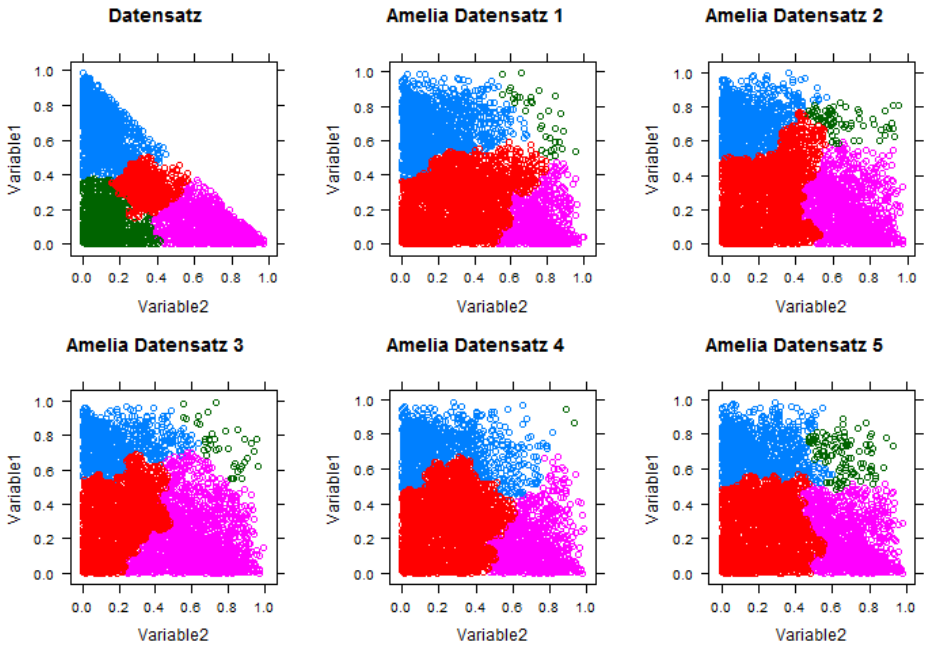

Abbildung 21: Plots zu Datensatz 5 und zugehörigen Amelia Datensätzen mit Average Linkage und Euklid Metrik

Die starke Abnahme der Genauigkeit scheint darin begründet, dass viele Werte außerhalb des "Dreiecks" imputiert werden und sich deshalb das Ergebnis der Clusteranalyse bei diesem Verfahren stark verändert. Besonders deutlich wird dies beim Betrachten der grün markierten Gruppe, die in den Amelia Datensätzen meist sehr klein und an anderer Position zu finden ist.

Centroid Linkage

Bei der Clusteranalyse mit dem Centroid Linkage wird nur in den Datensätzen 1 und 5 eine gute Clusterung erzielt. Mit der Manhattan Metrik erreicht man zusätzlich für Datensatz 4 noch eine gute Clusterung (vgl. Tabelle 18). Um Auswirkungen erkennen zu können, betrachtet man wieder die Übersicht über die Übereinstimmungen sowie die Veränderung der Genauigkeiten:

Centroid/Euklid	D1	D2	D3	D4	D5
imputiert	0.96	0.68	0.99	0.77	0.66
nicht imputiert	0.99	0.68	1.00	0.72	0.62

Centroid/Maximum	D1	D2	D3	D4	D5
imputiert	0.97	1.00	1.00	0.72	0.65
nicht imputiert	1.00	0.99	1.00	0.67	0.61

Centroid/Manhattan	D1	D2	D3	D4	D5
imputiert	0.97	1.00	0.99	0.64	0.60
nicht imputiert	0.99	1.00	0.99	0.43	0.62

Tabelle 17: Übersicht über Übereinstimmungen (Centroid Linkage)

Centroid/Euklid	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.63	0.68	0.73	0.91
Genauigkeit nach Imputation	0.97	0.40	0.68	0.50	0.59

Centroid/Maximum	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.99	0.40	0.68	0.76	0.91
Genauigkeit nach Imputation	0.98	0.40	0.68	0.50	0.58

Centroid/Manhattan	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.40	0.68	0.91	0.90
Genauigkeit nach Imputation	0.98	0.40	0.68	0.50	0.55

Tabelle 18: Veränderung der Genauigkeiten (Centroid Linkage)

Hier fällt auf, dass bei Datensatz 3 mit allen Metriken die Imputation keinerlei Einfluss zu haben scheint. Dies hängt aber wieder mit einer schlechten Clusterung zusammen. Es werden wieder fast alle Merkmalsträger dem selben Cluster zugeordnet. Zur Demonstration sind in Abbildung 22 der ursprüngliche und die Amelia Datensätze mit den von Centroid Linkage und Maximum Metrik gefundenen Gruppen eingezeichnet. Hier wird ersichtlich, dass über alle Amelia Datensätze hinweg nahezu alle Objekt dem selben Cluster zugeordnet worden sind. Aus diesem Grund kommen auch die hohen Übereinstimmung zustande und daraus resultierend keine Veränderung der Genauigkeiten nach der Imputation. So lassen sich auch die hohen Übereinstimmungen von Datensatz 2 mit Maximum und Manhattan Metrik erklären.

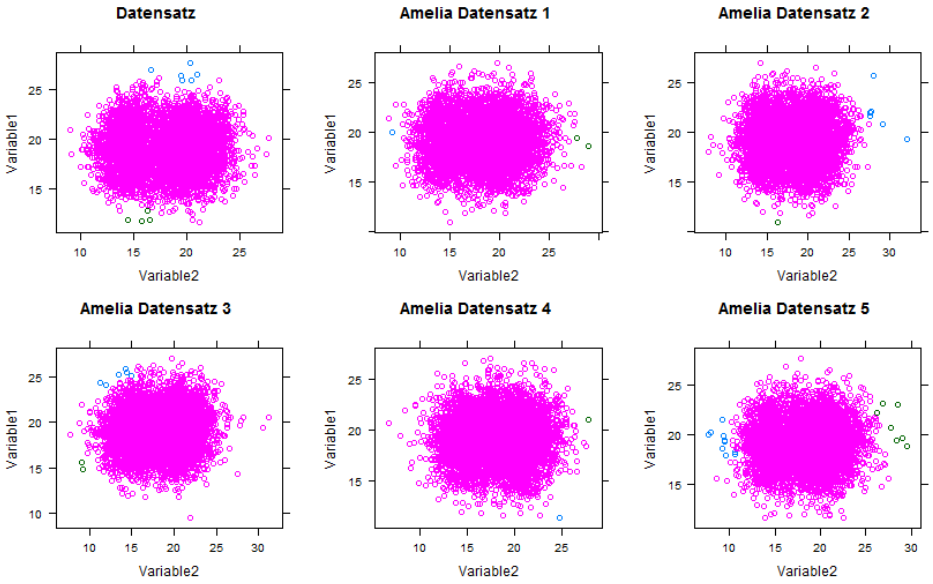

Abbildung 22: Plots zu Datensatz 3 und zughörigen Amelia Datensätzen mit Centroid Linkage und Maximum Metrik

Ward Linkage

Mit dem Ward Linkage werden im Vergleich zu den anderen Linkages die besten Ergebnisse erzielt. So werden abgesehen von Datensatz 3 durchweg gute Ergebnisse (vgl. Tabelle 20) erzielt. Die Auswirkung der Imputation wird wieder anhand dieser Tabellen ersichtlich:

Ward/Euklid	D1	D2	D3	D4	D5
imputiert	0.97	0.61	0.60	0.72	0.66
nicht imputiert	1.00	0.96	0.89	0.90	0.89

Ward/Maximum	D1	D2	D3	D4	D5
imputiert	0.97	0.59	0.58	0.74	0.66
nicht imputiert	1.00	0.94	0.73	0.89	0.94

Ward/Manhattan	D1	D2	D3	D4	D5
imputiert	0.96	0.61	0.57	0.73	0.67
nicht imputiert	0.99	0.95	0.92	0.92	0.93

Tabelle 19: Übersicht über Übereinstimmungen (Ward Linkage)

Ward/Euklid	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.94	0.75	0.94	0.89
Genauigkeit nach Imputation	0.98	0.74	0.61	0.76	0.70

Ward/Maximum	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.95	0.78	0.93	0.88
Genauigkeit nach Imputation	0.98	0.71	0.53	0.78	0.70

Ward/Manhattan	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.93	0.52	0.83	0.87
Genauigkeit nach Imputation	0.97	0.71	0.43	0.77	0.71

Tabelle 20: Veränderung der Genauigkeiten (Ward Linkage)

Die Übereinstimmung in den nicht imputierten Zeilen ist über alle Datensätze gesehen relativ hoch. Der Einfluss der Imputation auf die nicht imputierten Zeilen scheint also nicht so stark zu sein wie bei anderen Linkage Funktionen. Dementsprechend sinken auch die Genauigkeiten nach der Imputation nicht in dem Maße wie es bisher der Fall war. So ist in Abbildung 23 wieder ein Plot wie in Abbildung 22 dargestellt. Beim Vergleich dieser Abbildungen wird deutlich, dass die Amelia Datensätze mit dem Ward Linkage nicht mehr so stark vom ursprünglichen Datensatz abweichen wie noch zuvor beim Average Linkage. So wird hier auch die grün markierte Gruppe besser lokalisiert.

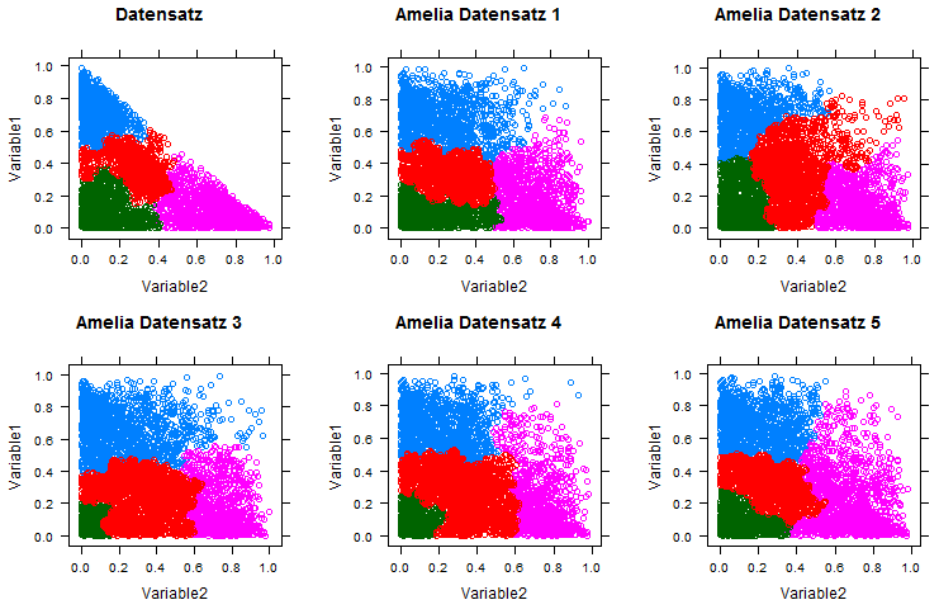

Abbildung 23: Plots zu Datensatz 5 und zugehörigen Amelia Datensätzen mit Ward Linkage und Euklid Metrik

Auch beim Ward Linkage hat die Imputation je nach Datensatz unterschiedlich starken Einfluss. Insgesamt scheint der Einfluss in den meisten Datensätzen allerdings geringer zu sein als bei den anderen Linkage Funktionen. Auch die Genauigkeiten der Clusterungen mit den einzelnen Amelia Datensätze schwanken nicht mehr so stark als es beispielsweise bei dem Complete Linkage (vgl. Abbildung 24 bzw. Abbildung 20) der Fall war.

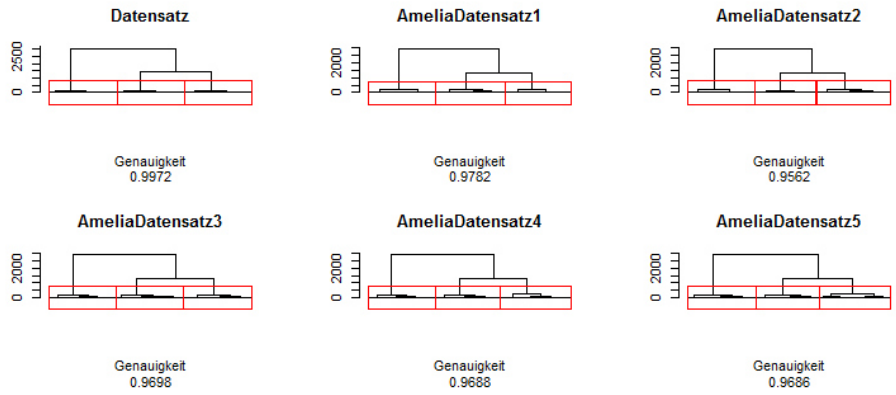

Abbildung 24: Plots zu Datensatz 5 und zugehörigen Amelia Datensätzen mit Ward Linkage und Maximum Metrik

Divisive Verfahren

Mit dem divisiven Verfahren werden die Datensätze größtenteils gut geclustert (vgl.Tabelle 22). Auch hier werden wieder die Übereinstimmung sowie die Veränderung der Genauigkeiten betrachtet:

Divisiv/Euklid	D1	D2	D3	D4	D5
imputiert	0.94	0.67	0.64	0.76	0.69
nicht imputiert	0.98	0.94	0.89	0.86	0.70

Divisiv/Maximum	D1	D2	D3	D4	D5
imputiert	0.94	0.67	0.64	0.76	0.69
nicht imputiert	0.98	0.94	0.89	0.86	0.70

Divisiv/Manhattan	D1	D2	D3	D4	D5
imputiert	0.95	0.58	0.49	0.70	0.61
nicht imputiert	0.99	0.61	0.73	0.94	0.88

Tabelle 21: Übersicht über Übereinstimmungen (Divisiv Linkage)

Divisiv/Euklid	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.97	0.85	0.90	0.92	0.69
Genauigkeit nach Imputation	0.94	0.66	0.67	0.79	0.50

Divisiv/Maximum	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.97	0.85	0.90	0.92	0.69
Genauigkeit nach Imputation	0.94	0.66	0.67	0.79	0.50

Divisiv/Manhattan	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.97	0.66	0.90	0.92	0.81
Genauigkeit nach Imputation	0.94	0.68	0.51	0.76	0.64

Tabelle 22: Veränderung der Genauigkeiten (divisiv)

Zuerst fällt ins Auge, dass die Ergebnisse mit der Euklid Metrik identisch sind zu den Ergebnissen mit der Maximum Metrik. Auffällig ist, dass beim divisiven Clustering mit dem Manhattan Linkage bei Datensatz 2 die Genauigkeit nach der Imputation höher ist als davor. Es werden also mehr Merkmalsträger richtig geclustert, die vor der Imputation falsch waren, als umgekehrt. Erkenntlich wird dies in der zugehörigen Vergleichstabelle 2:

Wie oft wurde richtig geclustert?:	Gesamtanzahl.	davon Falsch	Richtig
Gesamt	5000	1595	3405
–davon ursprünglich Richtig geclustert	3283	955	2328
–davon ursprünglich Falsch geclustert	1717	640	1077
Imputiert	3000	1367	1633
–davon ursprünglich Richtig geclustert	1933	775	1158
–davon ursprünglich Falsch geclustert	1067	592	475

Tabelle 23: Vergleichstabelle 2 zu Datensatz 2 (Divisiv/Manhattan Metrik)

Da die ursprüngliche Genauigkeit schon nicht sehr hoch war und die Veränderung durch die Imputation gering ist, kann dieser Fall als Sonderfall bzw. als Ausnahmefall bezeichnet werden. Zudem tritt dieser Fall nur in Datensatz 2 auf. Ansonsten fallen keine große Besonderheiten auf. Die Imputation hat Einfluss auf das Ergebnis der Clusteranalyse,der je nach Datensatz wieder unterschiedlich stark ist.

6.2.2 Partitionierende Verfahren

K-means Verfahren

Mit den K-means Verfahren werden alle Datensätze bis auf Datensatz 3 sehr gut geclustert (vgl. Tabelle 25). Hier gibt es zwischen den einzelnen Amelia Imputationen keine großen Unterschiede bezüglich der Genauigkeiten. Dies demonstriert Abbildung 25, in der wieder in Datensatz 5 und in den zugehörigen Amelia Datensätze die Gruppen eingezeichnet worden, die mit dem K-means Verfahren gefunden worden. Beim Vergleich dieser Plots mit denen in Abbildung 23 stellt man fest, dass sich die Clusterungen der Amelia Datensätze untereinander weniger stark unterscheiden. Die Genauigkeiten nach den Imputationen nehmen weiterhin ab (vgl. Tabelle 25). Diese Veränderungen sind nochmals leicht geringer als es beim Clustern mit dem Ward Linkage der Fall war. Dies hängt auch damit zusammen, dass in den nicht imputierten Zeilen hohe Übereinstimmungen erreicht werden (vgl. Tabelle 24).

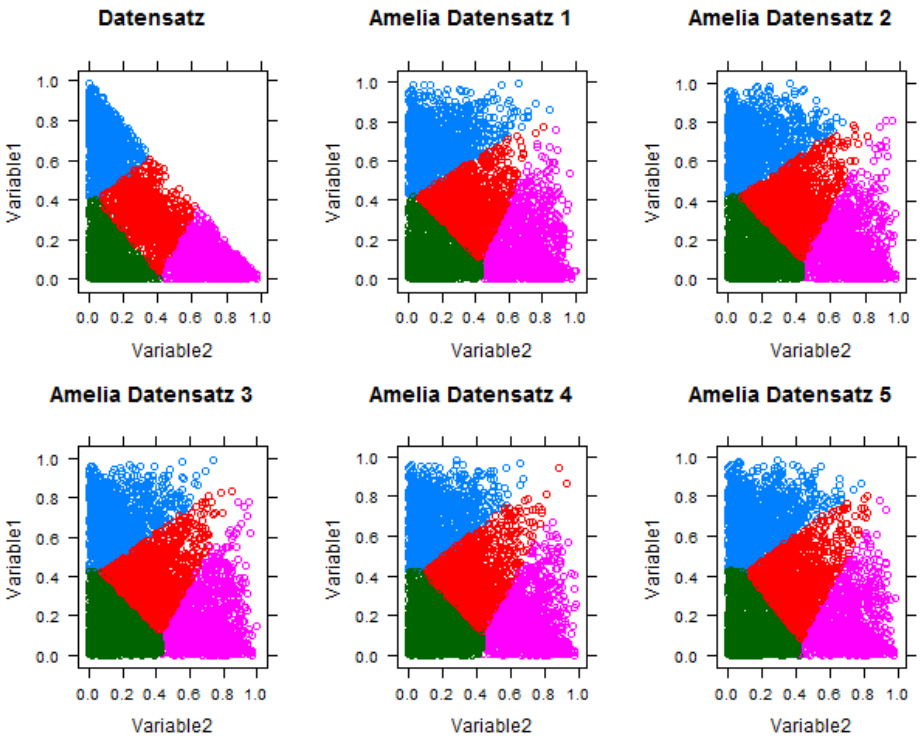

Abbildung 25: Plots zu Datensatz 4 und zugehörigen Amelia Datensätzen mit Kmeans Verfahren

Kmeans	D1	D2	D3	D4	D5
imputiert	0.97	0.63	0.55	0.74	0.68
nicht imputiert	1.00	0.97	0.74	0.96	0.98

Tabelle 24: Übersicht über Übereinstimmungen (Kmeans)

Kmeans	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.93	0.69	0.92	0.90
Genauigkeit nach Imputation	0.98	0.73	0.43	0.78	0.73

Tabelle 25: Veränderung der Genauigkeiten (K-means)

PAM

Mit den “PAM” Verfahren (bzw. K-medoids Verfahren) werden wieder gute Clusterergebnisse (abgesehen von Datensatz 3) vor der Imputation erreicht. (siehe Tabelle 26). Die Ergebnisse dieser Verfahren ähneln denen des

K-means Verfahrens, auch hier sind die Übereinstimmungen in den nicht imputierten Zeilen sehr hoch. Daraus resultierend verändern sich die Genauigkeiten nicht mehr in dem Umfang wie bei den hierarchischen Verfahren. Zwischen den beiden Metriken lassen sich keine großen Unterschiede feststellen.

PAM/Euklid	D1	D2	D3	D4	D5
imputiert	0.97	0.63	0.57	0.76	0.71
nicht imputiert	1.00	0.98	0.82	0.97	0.98

PAM/Manhattan	D1	D2	D3	D4	D5
imputiert	0.97	0.64	0.57	0.77	0.71
nicht imputiert	1.00	0.99	0.86	0.97	0.97

Tabelle 26: Übersicht über Übereinstimmungen (PAM)

PAM/Euklid	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.94	0.62	0.93	0.89
Genauigkeit nach Imputation	0.98	0.74	0.43	0.80	0.74

PAM/Manhattan	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.99	0.93	0.58	0.94	0.89
Genauigkeit nach Imputation	0.98	0.74	0.44	0.81	0.74

Tabelle 27: Veränderung der Genauigkeiten (PAM)

6.2.3 Modellbasiertes Verfahren

Mit dem modellbasierten Verfahren erzielt man für Datensatz 1 und 2 wieder gute Ergebnisse (vgl. Tabelle 29). Bei Datensatz 5 ist die Genauigkeit im Vergleich zu den partitionierenden Verfahren geringer. Für Datensatz 4 wird eine hohe Genauigkeit vor der Imputation erreicht, obwohl diesem Verfahren eine Normalverteilungsannahme zugrunde liegt. Wie schon bei den meisten Verfahren zuvor wird für Datensatz 3 keine gutes Clusterergebnis erzielt. Auch der modellbasierte Ansatz hat hier Probleme, die richtigen Gruppen zu finden.

Wie schon bei dem vorherigen Beispiel zeigt sich bei den Übereinstimmungen, dass diese in den nicht imputierten Zeilen sehr groß und in den imputierten Zeilen relativ klein (vgl. Tabelle 28) sind. Die Genauigkeiten nehmen in den normalverteilten Datensätzen ähnlich stark ab wie bei den partitionierenden Verfahren, bei den dirichlet verteilten allerdings noch ein wenig mehr (vgl. Tabelle 29).

Modellbasiert	D1	D2	D3	D4	D5
imputiert	0.97	0.62	0.64	0.72	0.71
nicht imputiert	1.00	0.97	0.97	0.94	0.88

Tabelle 28: Übersicht über Übereinstimmungen (Modellbasiert)

Modellbasiert/Manhattan	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.96	0.59	0.92	0.79
Genauigkeit nach Imputation	0.98	0.74	0.51	0.74	0.61

Tabelle 29: Veränderung der Genauigkeiten (Modellbasiert)

6.2.4 mit Gower Distanz

Die bisher beschriebenen Verfahren bzw. Ergebnisse basierten alle auf einen Datensatz ohne kategoriale Variable, da diese bei diesen Verfahren nicht berücksichtigt werden konnten. Bei den Ergebnissen der nächsten Verfahren ist dies anders, aus diesem Grund basieren diese auf Datensätze, die zusätzlich eine kategoriale Variable beinhalten.

agglomerativ mit Single Linkage

Zwar werden nicht mehr fast alle Objekte einem Cluster zugeordnet wie es beim Single Linkage zuvor der Fall war, dennoch werden keine guten Clusterergebnisse erzielt. Die Übereinstimmung sind insgesamt bis auf Datensatz 5 hoch, deshalb verändern sich auch die Genauigkeiten nur geringfügig. Bei Datensatz 5 hingegen sind die Übereinstimmungen sehr gering, wodurch es zu einer starken Veränderung der Genauigkeit kommt.

Single/Gower	D1	D2	D3	D4	D5
imputiert	0.85	0.82	0.79	0.83	0.48
nicht imputiert	1.00	1.00	1.00	1.00	0.44

Tabelle 30: Übersicht über Übereinstimmungen (Single/Gower)

Single/Gower	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.78	0.52	0.51	0.63	0.58
Genauigkeit nach Imputation	0.75	0.45	0.46	0.59	0.14

Tabelle 31: Veränderung der Genauigkeiten (Single/Gower)

agglomerativ mit Complete Linkage

Mit dem Complete Linkage und der Gower Distanz werden wieder bessere Ergebnisse im Vergleich zum Single Linkage erzielt. Allerdings kann

nur bei Datensatz 1 und 5 von einer guten Clusterung gesprochen werden (vgl.Tabelle 33). Die Übereinstimmungen sind durchwegs sehr gering (vgl. Tabelle 32), deshalb kommt es bei allen Datensätzen zu einem großen Abfall der Genauigkeiten.

Complete/Gower	D1	D2	D3	D4	D5
imputiert	0.76	0.45	0.57	0.59	0.48
nicht imputiert	0.79	0.39	0.79	0.67	0.56

Tabelle 32: Übersicht über Übereinstimmungen (Complete/Gower)

Complete/Gower	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.89	0.67	0.75	0.73	0.85
Genauigkeit nach Imputation	0.75	0.23	0.46	0.59	0.44

Tabelle 33: Veränderung der Genauigkeiten (Complete/Gower)

agglomerativ mit Average Linkage

Mit diesem Verfahren werden über alle Datensätze hinweg gute Clusterergebnisse erzielt, selbst für Datensatz 3, dessen echte Gruppen in den vorherigen Verfahren größtenteils nicht identifiziert werden konnten. Jedoch kommt es aufgrund der niedrigen Übereinstimmungen zu deutlichen Abfällen der Genauigkeiten, sodass die Clusterergebnisse nach der Imputation unbrauchbar sind.

Average/Gower	D1	D2	D3	D4	D5
imputiert	0.73	0.45	0.58	0.58	0.55
nicht imputiert	0.79	0.53	0.69	0.68	0.78

Tabelle 34: Übersicht über Übereinstimmungen (Average/Gower)

Average/Gower	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.95	0.84	0.94	0.90
Genauigkeit nach Imputation	0.75	0.44	0.47	0.59	0.56

Tabelle 35: Veränderung der Genauigkeiten (Single/Gower)

agglomerativ mit Centroid Linkage

Beim Centroid Linkage mit der Gower Distanz ergibt sich ein ähnliches Bild wie beim Complete Linkage. Auch hier wird nur für Datensatz 1 und 5 eine gute Clusterung erreicht. Die Imputation führt wieder zu einer deutlichen Veränderung der Genauigkeiten.

Centroid/Gower	D1	D2	D3	D4	D5
imputiert	0.73	0.66	0.50	0.54	0.38
nicht imputiert	0.79	0.75	0.44	0.74	0.52

Tabelle 36: Übersicht über Übereinstimmungen (Centroid/Gower)

Centroid/Gower	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	1.00	0.40	0.68	0.67	0.91
Genauigkeit nach Imputation	0.75	0.45	0.47	0.47	0.56

Tabelle 37: Veränderung der Genauigkeiten (Centroid/Gower)

agglomerativ mit Ward Linkage

Auch mit den Ward Linkage ergibt sich ein ähnliches Bild wie schon zuvor, durch die Imputation fallen auch hier die Genauigkeiten, die vor der Imputation größtenteils hoch sind, stark ab.

Ward/Gower	D1	D2	D3	D4	D5
imputiert	0.75	0.42	0.66	0.58	0.61
nicht imputiert	0.83	0.53	0.89	0.69	0.77

Tabelle 38: Übersicht über Übereinstimmungen (Centroid/Gower)

Ward/Gower	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.94	0.98	0.68	0.94	0.91
Genauigkeit nach Imputation	0.75	0.45	0.46	0.59	0.59

Tabelle 39: Veränderung der Genauigkeiten (Ward/Gower)

divisiv

Bei den divisiven Verfahren verhält es sich wie bei den agglomerativen Verfahren mit der Gower Distanz. Die Genauigkeiten vor der Imputation sind hoch, fallen aber durch die Imputation stark ab.

Divisiv/Gower	D1	D2	D3	D4	D5
imputiert	0.69	0.46	0.49	0.56	0.51
nicht imputiert	0.78	0.57	0.51	0.67	0.72

Tabelle 40: Übersicht über Übereinstimmungen (Divisiv/Gower)

Divisiv/Gower	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.90	0.85	0.93	0.92	0.83
Genauigkeit nach Imputation	0.63	0.45	0.44	0.59	0.57

Tabelle 41: Veränderung der Genauigkeiten (Divisiv/Gower)

PAM

Auch bei den partitionierenden Verfahren mit der Gower Distanz ergeben sich dieselben Ergebnisse wie zuvor bei den hierarchischen Verfahren mit der Gower Distanz: Vor der Imputation sind die Genauigkeit hoch, nehmen aber durch die Imputation stark ab.

PAM/Gower	D1	D2	D3	D4	D5
imputiert	0.75	0.44	0.75	0.58	0.63
nicht imputiert	0.81	0.52	0.89	0.69	0.73

Tabelle 42: Übersicht über Übereinstimmungen (PAM/Gower)

PAM/Gower	D1	D2	D3	D4	D5
Genauigkeit vor Imputation	0.97	0.95	0.51	0.94	0.84
Genauigkeit nach Imputation	0.75	0.44	0.47	0.59	0.59

Tabelle 43: Veränderung der Genauigkeiten (PAM/Gower)

6.3 Zusammenfassung der Ergebnisse

Insgesamt wurden die Clusterergebnisse vor der Imputation mit kompletten Daten mit den Clusterergebnissen verglichen, bei denen fehlende Daten erzeugt und anschließend wieder imputiert wurden.

Wie schon des öfteren erwähnt, kann für keines der einzelnen Clusterverfahren eine genaue Tendenz dafür gegeben werden, inwiefern sich das Ergebnis der Clusteranalyse im Falle imputierter fehlender Daten verändert, da dies auch abhängig vom jeweiligen untersuchten Datensatz ist. Jedoch gibt es Clusterverfahren, dessen Ergebnisse durch die Imputation geringer beeinflusst werden als andere. Hierzu zählen die partitionierenden Verfahren sowie das modellbasierte Verfahren, dessen Genauigkeiten im Vergleich mit den meisten hierarchischen Verfahren durch die Imputation weniger stark beeinflusst werden. Auch die Schwankungen der Genauigkeiten zwischen den einzelnen Amelia Datensätzen sind bei diesen Verfahren deutlich geringer. Zusätzlich sind die Übereinstimmungen in den nicht imputierten Zeilen durchwegs höher. Zwischen den einzelnen partitionierenden Verfahren und dem modellbasierten Verfahren lassen sich keine allzu großen Unterschiede feststellen, sodass diese im Hinblick auf die Veränderung der Genauigkeit in etwa auf die selbe Stufe gestellt werden können. Die einzigen Verfahren der hierarchischen Verfahren, bei denen vergleichbare Ergebnisse erzielt werden, sind die agglomerativen Verfahren mit dem Ward Linkage. Allerdings schwanken hier die Genauigkeiten der einzelnen Amelia Datensätze stark, wenn auch im Vergleich zu den anderen hierarchischen Verfahren geringer. Die agglomerativen Verfahren mit den anderen Linkage Funktionen sind nach der Imputation größtenteils unbrauchbar, da die Genauigkeiten nach der Imputation stark absinken. Betrachtet man die Ergebnisse der Verfahren, die auf Grundlage der Gower Distanz und Datensätzen mit einer kategorialen Variable erhalten hat, stellt man selbst bei dem partitionierenden Verfahren einen starken Abfall der Genauigkeit fest. An dieser Stelle sei noch erwähnt, dass die hauptsächliche Ursache für die Abnahme der Genauigkeit die teils ungenauen Imputationen sind und nicht die einzelnen Clusterverfahren. Bei manchen Verfahren wirkt sich die Imputation lediglich weniger stark aus wie bei anderen.

Die Datensätze 1 bis 3 aus dieser Arbeit sind multivariat normal verteilt und entsprechen demnach auch der Modellannahme, welche auch für die Imputation bzw. für das modellbasierte Verfahren getroffen werden. Zudem ist die Variablenanzahl der untersuchten Datensätze relativ gering. Dennoch fallen auch hier die Genauigkeiten nach der Imputation (abgesehen von Datensatz 1) meist stark ab. Übertragt man dies nun auf "echte Datensätze", bei denen eine Normalverteilungsannahme nicht immer zutreffend ist und zumal weit aus mehr Variablen enthalten sind, sind Zweifel darüber ange-

bracht, ob die Ergebnisse der Clusteranalyse im Falle imputierter Daten Sinn machen.

Ist es dennoch das Ziel, in einem Datensatz mit fehlenden Werten den Objekten eine Gruppe zuordnen, so empfiehlt es sich, eines der partitionierenden Verfahren oder das modellbasierte Verfahren zu verwenden. Bei diesen stellte sich der Abfall der Genauigkeiten noch am geringsten dar. Enthält der zu untersuchende Datensatz sowohl metrische als auch kategoriale Variablen, sollte im Falle imputierter Daten gänzlich auf eine Clusteranalyse , zumindest mit den hier angewendeten Verfahren, verzichtet werden.

7 Ausblick

In dieser Arbeit wurde lediglich die Auswirkung der Imputation bei 30 % fehlender Beobachtungen untersucht. Bei einer geringeren Anzahl an fehlender Beobachtungen ist die Auswirkung der Imputation mit großer Sicherheit geringer. Dementsprechend könnte ähnliche Simulationsstudie wie in dieser Arbeit mit beispielsweise 10 % oder 20 % fehlender Beobachtungen durchführt werden. Möglicherweise würden dadurch dann auch in imputierten Datensätzen gute Clusterergebnisse resultieren. Zudem gibt es neben den untersuchten Clusterverfahren zahlreiche weitere Verfahren, bei denen die Auswirkung der Imputation untersucht werden könnte. Ein weiterer Ansatz, um bessere Ergebnisse nach der Imputation zu erhalten, wäre die Verbesserung der Imputation, entweder indem mehr als 5 Imputationen durchführt oder andere Imputationsmethoden anwendet werden.

A Abbildungs- und Tabellenverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

B R-Code

Hier wird der gesamte R-Code dargestellt, der für die Durchführung der Simulationsstudie benötigt wurde.

B.1 Erzeugung der Datensätze

In diesem Abschnitt ist der R-Code zu finden, mit dem die verschiedenen Datensätze erzeugt wurden. Hier können auch die genauen Erwartungswerte, Varianzen und Korrelationen abgelesen werden.

```
# Teil 1: Erzeugung der Datensätze

# Erzeugung der Datensätze

# Pakete
library(MASS)
library(mvtnorm)
library(scatterplot3d)
library(lattice)
library(matrixcalc)
library(Matrix)
library(MCMCpack)

# Datensätze 1 bis 3

#############Normalverteilte Daten#######################################

# MNVfunktion: Erstellt multinormalverteilten Cluster
# n<-Populationsgrösse, mu <-Erwartungswerte, Varianzen<-Sigma,
# cor<- Korrelationen(12,13,23)
MNV <- function(n=100, mu1=5, mu2=8,mu3=4, Varianz1=2, Varianz2=3,Varianz3=5,
                cor12=-0.5,cor13=0.5, cor23=0.3){
  cov12<-cor12*sqrt(Varianz1)*sqrt(Varianz2)
  cov13<-cor13*sqrt(Varianz1)*sqrt(Varianz3)
  cov23<-cor23*sqrt(Varianz2)*sqrt(Varianz3)
  Sigma=matrix(data=c(Varianz1, cov12, cov13,
                      cov12, Varianz2, cov23,
                      cov13,cov23,Varianz3), nrow=3, ncol=3)
  if((is.positive.definite(Sigma))==TRUE){
    Y <- mvrnorm(n=n, mu=c(mu1, mu2,mu3),Sigma)
    return(Y)}
  else
  {
    Sigma=nearPD(Sigma,keepDiag=TRUE)
    Sigma<-as.matrix(Sigma$mat)
    Y <- mvrnorm(n=n, mu=c(mu1, mu2,mu3),Sigma)
    print("Warnung: Korrelationen möglicherweise nicht korrekt")
    return(Y)
  }
}
###############Datensatz 1#################################

# set. seed für Reproduzierbarkeit
set.seed(123)
# n=5000
# A=1666,B=1667,C=1667
# Gleiche Korrelationen
n1=1666
n2=1667
n3=1667

Cluster1<-MNV(n1,15,12,8,3,2,1,0.5,0.5,0.5)
Cluster1=cbind(Cluster1,rep(1,n1)) # für gruppenerstellung (Cluster)
colnames(Cluster1)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster1<-as.data.frame(Cluster1)

# Mithilfe der MVN Funktion wurde einer der drei Cluster erstelt, die der Datensatz
# später enthalten soll.


# Cluster2

Cluster2<-MNV(n2,23,20,10,3,2,1,0.5,0.5,0.5)
Cluster2=cbind(Cluster2,rep(2,n2)) #n=50, für gruppenerstellung
```

```
colnames(Cluster2)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster2<-as.data.frame(Cluster2)



# Cluster3

Cluster3<-MNV(n3,29,28,12,3,2,1,0.5,0.5,0.5)
Cluster3=cbind(Cluster3,rep(3,n3)) #n=50, für gruppenerstellung
colnames(Cluster3)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster3<-as.data.frame(Cluster3)

# Die drei erzeugten Cluster werden nun zu einem Datensatz zusammengefügt:
Datensatz1ord<-as.data.frame(rbind(Cluster1,Cluster2,Cluster3))

# Die dritte Variable wird mit der cut-Funktion in eine kategoriale Variable umgewandelt.
# Die "breaks" legen die Intervalle dafür fest.(Werte von 0 bis 9.2 -> kein,... )
Datensatz1ord$Variable3<-as.ordered(cut(Datensatz1ord$Variable3,breaks=c(0,9.2,11,25),
                                        labels=c("Kein","mittel","viel")))
# Die labels worden "kein", "mittel","viel" benannt, dahinter steckt allerdings keine
# größere Bedeutung, es handelt sich lediglich um frei wählbare Bezeichnungen.
summary(Datensatz1ord)
# Plot: Cluster1 ist blau markiert,Cluster 2 rot und Cluster 3 grün.
xyplot(Datensatz1ord$Variable1~Datensatz1ord$Variable2,
       groups=Datensatz1ord$Gruppe,main="Datensatz1ord",
       xlab="Variable2",ylab="Variable1",xlim=c(0,35),ylim=c(5,40))

rm(Cluster1,Cluster2,Cluster3,n1,n2,n3) # Überflüssige Objekte werden entfernt

###############Datensatz 2################################
# Die Erzeugung der Datensätze 2 und 3 läuft analog dazu, wie Datensatz 1 erzeugt wurde,
# es werden nur andere Korrelationen, Varianzen, Gruppengrößen und Erwartungswerte gewählt.

# set. seed für Reproduzierbarkeit
set.seed(123)
# n=5000
# A=1500,B=1500,C=2000
# Die Korrelationen mit der 3 Variable sind bei allen Clustern gleich.
n1=1500
n2=1500
n3=2000
Cluster1<-MNV(n1,22,22,10,3,2,2.5,-0.1,0.5,-0.5)
Cluster1=cbind(Cluster1,rep(1,n1)) # für gruppenerstellung (Cluster)
colnames(Cluster1)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster1<-as.data.frame(Cluster1)


# Cluster2

Cluster2=MNV(n2,24,18,8,2.5,1.5,1.5,0.5,0.5,-0.5)
Cluster2=cbind(Cluster2,rep(2,n2)) #n=50, für gruppenerstellung
colnames(Cluster2)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster2<-as.data.frame(Cluster2)


# Cluster3

Cluster3=MNV(n3,17,20,7,2,3,2,-0.5,0.5,-0.5)
Cluster3=cbind(Cluster3,rep(3,n3)) #n=50, für gruppenerstellung
colnames(Cluster3)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster3<-as.data.frame(Cluster3)


Datensatz2ord<-as.data.frame(rbind(Cluster1,Cluster2,Cluster3))
Datensatz2ord$Variable3<-as.ordered(cut(Datensatz2ord$Variable3,breaks=c(0,5.5,9.5,18),
                                        labels=c("Kein","mittel","viel")))
# Plot
xyplot(Datensatz2ord$Variable1~Datensatz2ord$Variable2,
       groups=Datensatz2ord$Gruppe,main="Datensatz2ord",
       xlab="Variable2",ylab="Variable1",)

summary(Datensatz2ord)
rm(Cluster1,Cluster2,Cluster3,n1,n2,n3)

###############Datensatz 3################################
# set. seed für Reproduzierbarkeit
set.seed(123)
# n=5000
# A=900,B=3400,C=700
# Unterschiedliche Korrelationen und Varianzen
n1=900
n2=3400
n3=700
Cluster1<-MNV(n1,21,14,7,3,2,1,0.5,0.1,-0.7)
```

```
Cluster1=cbind(Cluster1,rep(1,n1)) # für gruppenerstellung (Cluster)
colnames(Cluster1)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster1<-as.data.frame(Cluster1)



# Cluster2

Cluster2<-MNV(n2,19,20,15,5,4,4,0.1,-0.5,0.3)
Cluster2=cbind(Cluster2,rep(2,n2)) #n=50, für gruppenerstellung
colnames(Cluster2)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster2<-as.data.frame(Cluster2)



# Cluster3

Cluster3<-MNV(n3,17,14.5,7,2,2,2,-0.2,0.5,0.8)
Cluster3=cbind(Cluster3,rep(3,n3)) #n=50, für gruppenerstellung
colnames(Cluster3)<-c("Variable1","Variable2","Variable3","Gruppe")
Cluster3<-as.data.frame(Cluster3)


Datensatz3nom<-as.data.frame(rbind(Cluster1,Cluster2,Cluster3))
Datensatz3nom$Variable3<-(cut(Datensatz3nom$Variable3,breaks=c(0,10.7,15,25),
                              labels=c("Kein","mittel","viel")))
summary(Datensatz3nom)
# Plot: Grün=Cluster3, Blau=Cluster1, Rot=Cluster2
xyplot(Datensatz3nom$Variable1~Datensatz3nom$Variable2,
       groups=Datensatz3nom$Gruppe,main="Datensatz3nom",
       xlab="Variable2",ylab="Variable1")

rm(Cluster1,Cluster2,Cluster3,n1,n2,n3)

############################Drichlet verteilte Daten (mit kategorialer Variable)

# Im Grunde werden die Datensätze wie die vorherigen Datensätze erzeugt, mit dem Unterschied,
# das hier für die Clusterbildung die rdrichtlet Funktion verwendet wird:
# rdrichlet: n=StichprobenVariable1, c(1,2,3) = shape parameter der Drichlet-Verteilung


# set. seed für Reproduzierbarkeit
set.seed(123)


##########Datensatz 4###################

# A=2500,B=1500, C=1000
# Wie groß soll der Cluster sein?
n1=2500
n2=1500
n3=1000

Cluster1<-rdirichlet(n1, c(8,4,8) )
Cluster1=as.data.frame(cbind(Cluster1,rep(1,n1))) #für gruppenerstellung
colnames(Cluster1)<-c("Variable1","Variable2","Variable3","Gruppe")



Cluster2<-rdirichlet(n2, c(1,7,7) )
Cluster2=as.data.frame(cbind(Cluster2,rep(2,n2))) #für gruppenerstellung
colnames(Cluster2)<-c("Variable1","Variable2","Variable3","Gruppe")



Cluster3<-rdirichlet(n3, c(1,1,9) )
Cluster3=as.data.frame(cbind(Cluster3,rep(3,n3))) #für gruppenerstellung
colnames(Cluster3)<-c("Variable1","Variable2","Variable3","Gruppe")



Datensatz4ord<-rbind(Cluster1,Cluster2,Cluster3)


# mit ordinalen Merkmal
Datensatz4ord$Variable3<-as.ordered(cut(Datensatz4ord$Variable3,breaks=c(0,0.4,0.7,1),
                                        labels=c("Kein","mittel","viel")))

# Plot: Grün=Pfirsich, Blau=Apfel, Rot=Birnen
xyplot(Datensatz4ord$Variable1~Datensatz4ord$Variable2, groups=Datensatz4ord$Gruppe
       ,main="Datensatz4ord",xlab="Variable2",ylab="Variable1")

rm(Cluster1,Cluster2,Cluster3,n1,n2,n3)
```

```
##########Datensatz 5###################
# 4 statt 3 Gruppen
# set. seed für Reproduzierbarkeit
set.seed(123)

# Drichlet verteilte Daten (mit kategorialer Variable) und 4 Clustern
# A=B=C=D=1250
# Wie groß sollen die Cluster sein?
n1=1250
n2=1250
n3=1250
n4=1250
library(MCMCpack)
Cluster1<-rdirichlet(n1, c(6,1,2) )
Cluster1=as.data.frame(cbind(Cluster1,rep(1,n1))) #für gruppenerstellung
colnames(Cluster1)<-c("Variable1","Variable2","Variable3","Gruppe")


Cluster2<-rdirichlet(n2, c(1,6,2) )
Cluster2=as.data.frame(cbind(Cluster2,rep(2,n2))) #für gruppenerstellung
colnames(Cluster2)<-c("Variable1","Variable2","Variable3","Gruppe")


Cluster3<-rdirichlet(n3, c(1,1,6) )
Cluster3=as.data.frame(cbind(Cluster3,rep(3,n3))) #für gruppenerstellung
colnames(Cluster3)<-c("Variable1","Variable2","Variable3","Gruppe")


Cluster4<-rdirichlet(n4, c(15,15,15) )
Cluster4=as.data.frame(cbind(Cluster4,rep(4,n4))) #für gruppenerstellung
colnames(Cluster4)<-c("Variable1","Variable2","Variable3","Gruppe")



Datensatz5nom<-as.data.frame(rbind(Cluster1,Cluster2,Cluster3,Cluster4))


# nominal
Datensatz5nom$Variable3<-cut(Datensatz5nom$Variable3,breaks=c(0,0.2,0.6,1),
                             labels=c("Kein","mittel","viel"))


# Plot: 4 Gruppe ist rot markiert
xyplot(Datensatz5nom$Variable1~Datensatz5nom$Variable2,
       groups=Datensatz5nom$Gruppe, xlab="Variable2",ylab="Variable1")

rm(Cluster1,Cluster2,Cluster3,Cluster4,n1,n2,n3,n4)
```

B.2 Funktionen

Hier sind die alle Funktionen beschrieben, die für die Simulationsstudie benötigt worden.

```
#Teil 2 Funktionen

################################## Funktionen für Datensatz 1 bis 3 ##########################

#Pakete
### Pakete
library(Amelia)
library(cluster)
library(mclust)
library(gregmisc)

#############################################################################################
################################## Funktion 1: Zuordnung  ###################################

### Zuordnungsfunktion
# Die Zuordnungsfunktion ist eine sehr wichtige Funktion, die später für die richtige
# Labelbezeichnung sorgt.
# Sie korrigiert die Label Reihenfolge.
# Funktionsanfang

Zuordnung<-function(groups,EchteGruppe){
  perms = permutations(3,3)                  #bildet alle Permuationen die mit 3 Elementen
  EchteGruppe1<-as.factor(EchteGruppe)
  V<-c(1:6)                                  #Dummy Vektor
  for(i in 1:nrow(perms)){                   #For-Schleife: Für jede mögliche Permuation
    groups1<-as.factor(groups)               #wird die Übereinstimmung mit der echten
    levels(groups1)<-perms[i,]               #Gruppe überprüft und die Anzahl die
    VR<-EchteGruppe1==groups1                #der Übereinstimmungen in V gespeichert.
```

```
    V[i]<-length(subset(VR,VR==TRUE))
  }
  S<-which.max(V)                          #Das Maximum von V ist die richtige Zuordnung
  groups<-as.factor(groups)                #S gibt die Stelle des Maximums von V an.
  levels(groups)<-perms[S,]                #Die groups erhalten die Levels, für die sie
                                           #maximal sind, also S.
  list(as.numeric(as.vector(groups)),max(V)) #Die Gruppe wird wieder als numerischer Vektor
}                                          #gespeichert, zusätzlich wird das Maximum von
                                           #V gespeichert (für die Genauigkeitsbestimmung).
#Funktionsende

##############################################################################################
################################## Funktion 2: NAErzeugung  ##################################

### Funktion zur Erzeugung von NA mit MAR Missing Mechanismus
# a1 der prozentuale Anteil der Daten, die aus dem Datensatz entfernt werden sollen. Es
# werden insgesamt 2 Datensätze mit NA mit dieser Funktion erzeugt.
# Der erste ist für den Datensatz ohne drittes Merkmal.
# Der zweite ist für den Datensatz mit drittem Merkmal.

#Funktionsanfang
NAErzeugung <-function (Datensatz,a1){

# Funktion zur Erzeugung von "NA" in Datensätzen ohne 3. Merkmal
  set.seed(123)
  DA<-Datensatz[,1:2]                      #Nur die ersten beiden Zeilen werden ausgewählt.
  a<-nrow(DA)*ncol(DA)*a1/100              #Ermittlung der Anzahl an Beobachtungen, die
                                           #entfernt wird.
  FKT<-function(p){                        #Hilfsfunktion, die einfache Münze mit der
    rbinom(1,1,p)                          #Wahrscheinlichkeit p "wirft". Man erhält
  }                                        #1 oder 0 als Resultat.

#Es werden die Wahrscheinlichkeiten bestimmt, das ein bestimmter Wert fehlt.

  # Wahrscheinlichkeiten
  # Wahrscheinlichkeit p1, das Wert aus X1 fehlt
  Logit<-function(x){
    p1=exp(-4+0.22*x[2])/(1+(exp(-4+0.22*x[2])))      #Funktion zur Bestimmung der Wahrschein-
    p1                                                #keit p in Abhängkeit anderer Merkmal(e).
  }

  L1<-apply(DA,1,Logit)
  #<-#Logit Funktion wird auf jede Zeile angwendet. Für jeden Merkmalsträger ist also die Wahr-
     #keit gegeben, das in dem Merkmalsträger die erste Beobachtung fehlt.

  L1<-cbind(as.vector(sapply(L1,FKT)),c(1:length(L1))) #Obige Hilsfunktion FKT wird angewendet.
  L1<-subset(L1,L1[,1]==1)                           #Die Merkmalsträger, in dem die "Münze"
  L1<-L1[,2]                                         #1 ergeben hat, werden ausgewählt.

  #L1 enthält die Merkmalsträger, die potentiell ausfallen könnten in der ersten Variable
  #Analog enthält man L2, das analog die Merkmalsträger für die zweite Variable enthält.

  # Wahrscheinlichkeit, das Wert aus X2 fehlt
  Logit<-function(x){
    p1=exp(-4+0.22*x[1])/(1+(exp(-4+0.22*x[1])))      #Analog
    p1
  }

  L2<-apply(DA,1,Logit)                              #Analog
  L2<-cbind(as.vector(sapply(L2,FKT)),c(1:length(L2)))
  L2<-subset(L2,L2[,1]==1)
  L2<-L2[,2]

  s<-sample((ncol(DA)),a,replace=TRUE)             #Es wird zufällig ermittelt, wieviele
  s1<-subset(s,s==1)                               #Beobachtungen in den jeweiligen
  s2<-subset(s,s==2)                               #Variablen fehlen.s1 Variable1
                                                   #s2 Variable 2. Insgesamt sollen a
                                                   #Beobachtungen fehlen.

  C1<-cbind((L1[(sample(1:length(L1),length(s1),replace=FALSE))]),s1)
  #<-#Aus L1 werden zufällig s1 viele Zeilen ausgewählt, die gelöscht werden.

  L2<-L2[ !(L2 %in% (as.numeric(C1[,1])))]
  #<-#Damit keine ganzen Zeilen verschwinden, werden in L2 nur Zeilen ausgewählt
  #  #die in der ersten Variable nicht ausgewählt worden.
  C2<-cbind((L2[(sample(1:length(L2),length(s2),replace=FALSE))]),s2)
  #<-#Aus L2 werden zufällig s2 viele Zeilen ausgewählt, die gelöscht werden.

  D<-rbind(C1,C2) #D enthält die Beobachtungen, die gelöscht werden sollen.

  r1<-D[,1]       #Gibt an, das in der ersten Variable Werte gelöscht werden.
  s1<-D[,2]       #Gibt an, das in der zweiten Variable Werte gelöscht werden.
```

```
  for(i in 1:length(r1)){      #Mithilfe dieser Schleife werden die Werte
    r2<-r1[i]                  #gelöscht bzw. NA erzeugt.
    s2<-s1[i]
    is.na(DA[r2,s2])<-TRUE
  }

##############################################################################################

  # Funktion zur Erzeugung von "NA" in Datensätzen mit 3. Merkmal
  # analoges Vorgehen wie oben beschrieben.
  set.seed(123)
  DA1<-Datensatz[,1:3]
  DA1$Variable3<-as.numeric(DA1$Variable3)
  a<-nrow(DA1)*ncol(DA1)*a1/100


  # Wahrscheinlichkeiten
  # Wahrscheinlichkeit, das Wert aus X1 fehlt
  Logit<-function(x){
    p1=exp(-3.0+0.10*x[2]+0.5*x[3])/(1+(exp(-3.0+0.1*x[2]+0.5*x[3])))
    p1
  }

  L1<-apply(DA1,1,Logit)
  L1<-cbind(as.vector(sapply(L1,FKT)),c(1:length(L1)))
  L1<-subset(L1,L1[,1]==1)
  L1<-L1[,2]

  # Wahrscheinlichkeit, das Wert aus X2 fehlt
  Logit<-function(x){
    p1=exp(-3.0+0.10*x[1]+0.5*x[3])/(1+(exp(-3.0+0.10*x[1]+0.5*x[3])))
    p1
  }

  L2<-apply(DA1,1,Logit)
  L2<-cbind(as.vector(sapply(L2,FKT)),c(1:length(L2)))
  L2<-subset(L2,L2[,1]==1)
  L2<-L2[,2]

  # Wahrscheinlichkeit, das Wert aus X3 fehlt
  Logit<-function(x){
    p1=exp(-3+0.10*x[1]+0.10*x[2])/(1+(exp(-3+0.10*x[1]+0.10*x[2])))
    p1
  }

  L3<-apply(DA1,1,Logit)
  L3<-cbind(as.vector(sapply(L3,FKT)),c(1:length(L3)))
  L3<-subset(L3,L3[,1]==1)
  L3<-L3[,2]

  s<-sample((ncol(Datensatz)-1),a,replace=TRUE)

  s1<-subset(s,s==1)
  s2<-subset(s,s==2)
  s3<-subset(s,s==3)

  C1<-cbind((L1[(sample(1:length(L1),length(s1),replace=FALSE))]),s1)
  C2<-cbind((L2[(sample(1:length(L2),length(s2),replace=FALSE))]),s2)

  LOE<-c(C1[,1],C2[,1])
  LOE<-as.numeric(LOE[which(duplicated(LOE))])

  L3<-L3[ !(L3 %in% (LOE))]
  C3<-cbind((L3[(sample(1:length(L3),length(s3),replace=FALSE))]),s3)
  D<-rbind(C1,C2,C3)

  r1<-D[,1]
  s1<-D[,2]


  for(i in 1:length(r1)){
    r2<-r1[i]
    s2<-s1[i]
    is.na(DA1[r2,s2])<-TRUE
  }
DA1$Variable3<-as.factor(DA1$Variable3) # wird wieder kategorial

list(DatensatzNA1=DA, DatensatzNA11=DA1)
#Beide Datensätze mit NA werden in einer Liste gespeichert.
#DatensatzNA1 ist der Datensatz ohne 3.Merkmal, DatensatzNA11 mit 3.Merkmal.
}

#Funktionsende
```

```
#############################################################################################
################################### Funktion 3: AmeliaDatensatzErzeugung  ######################

### Imputation der fehlenden Werte im DatensatzNA1 mit Amelia #######

### Amelia Daten Imputation

# Funktionsanfang
AmeliaDatensatzeErzeugung<-function(DatensatzNA,x="noms"){
  #ohne 3 Variable
  DatensatzNA1<-DatensatzNA[[1]]                #DatensatzNA1 muss ausgewählt werden.

    Amelia<-amelia(DatensatzNA1,m=5)            #führt Imputationen durch


  #AmeliaDatensätze 1 bis 5 werden einzelnen gespeichert:
  AmeliaDatensatz1<-Amelia$imputations$imp1
  AmeliaDatensatz2<-Amelia$imputations$imp2
  AmeliaDatensatz3<-Amelia$imputations$imp3
  AmeliaDatensatz4<-Amelia$imputations$imp4
  AmeliaDatensatz5<-Amelia$imputations$imp5


  Ohnekat<-list(AmeliaDatensatz1,AmeliaDatensatz2, AmeliaDatensatz3,
              AmeliaDatensatz4,AmeliaDatensatz5)

  #AmeliaDatensätze ohne 3 Variable werden als Liste in Ohnekat gespeichert.

  #mit 3 Variable
  ### für x muss noms für Nominal oder ords für ordinal eingegeben werden

  DatensatzNA1<-DatensatzNA[[2]] #DatensatzNA11 muss ausgewählt werden.
  # Je nachdem ob die 3 Variable nominal oder ordinal behandelt werden soll:
  if(x=="noms") { # Wenn nominal
    Amelia<-amelia(DatensatzNA1,m=5,noms= "Variable3")
  # Imputierte Datensätze - Diese Datensätze sind die Datensätze 1 bis 5, die Amelia Imputiert.
  print("Nominal")}
  if(x=="ords") {# Wenn ordinal
    Amelia<-amelia(DatensatzNA1,m=5,ords="Variable3")
  print("Ordinal")}

 #AmeliaDatensätze 1 bis 5
  AmeliaDatensatz1<-Amelia$imputations$imp1
  AmeliaDatensatz2<-Amelia$imputations$imp2
  AmeliaDatensatz3<-Amelia$imputations$imp3
  AmeliaDatensatz4<-Amelia$imputations$imp4
  AmeliaDatensatz5<-Amelia$imputations$imp5


  Mitkat<-list(AmeliaDatensatz1,AmeliaDatensatz2, AmeliaDatensatz3,
       AmeliaDatensatz4,AmeliaDatensatz5)

  #Die AmeliaDatensätze mit 3. Variable werden als Liste in Mitkat gespeichert.
#########

  list("DatensatzNa ohne kategoriale Variable"=Ohnekat,"DatensatzNa mit kategoriale Variable"=Mitkat)
#Mitkat und ohnekat werden gespeichert.
}

#Funktionsende

#############################################################################################
################################### Funktion 4: Clustern1  ##################################

### Clusterfunktion
# Die Clustern Funktion führt 29 verschiedenene Clusterverfahren mit den ursprünglichen Datensatz
# durch.
# Die Verfahren 1-22 werden dabei mit einen um die kategoriale Variable reduzierten Datensatz
# durchgeführt.
# Bei Verfahren 23 bis 29 wird die kategoriale Variable berücksichtigt.
# In der Gruppenmatrix sind die Gruppenvektoren gespeichert.
# Der Genauigkeitsvektor speichert die Genauigkeiten der einzelnen Verfahren.

# Funktionsanfang
Clustern1<-function(Datensatz){

  ######################### Funktion: Clustern############################
  #Quelle http://www.statmethods.net/advstats/cluster.html

  ### Dummys
  # Wahrscheinlichkeitsmatrix: #Dummy:Wird später mit Genauigkeiten gefüllt
  Genauigkeitsvektor<-c(1:29)
  # Gruppenmatrix: # Dummy: wird später mit Gruppenvektoren gefüllt
  Gruppenmatrix<-matrix(nrow=nrow(Datensatz),ncol=29)
```

```
### Skalierung des Datensatze
# skalierter Datensatz # Die Daten müssen standardisiert werden:

# skalierter Datensatz
Variable2<-scale(Datensatz$Variable2)
Variable1<-scale(Datensatz$Variable1)

# Datensatz nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
# unterschiedlichen Skalen umgehen können
Datensatzscale2<-cbind(Variable2,Variable1)


# Datensatzscale2 nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
# unterschiedlichen Skalen umgehen können
# Datensatzscale 2 bildet die Grundlage für die Verfahren 1 bis 22.

################################## Clusterverfahren ########################################

##### Verfahren 1-15 Agglomeratives Clustering ######

# Möglichkeiten Distanz:euclidean,maximum, manhattan, ...
# Möglichkeiten Linkage: Single, complete, average, centroid, ward,..
# Clusterfunktion: a <- Distanzmetrik,b<-Linkage-Verfahren, mit ""!

# Hilfsfunktion
Cluster<-function(a,b){
  d <- dist(Datensatzscale2, method =a) # distance matrix
  Erg <- hclust(d, method=b)
  Erg
}
##### Single Linkage

### Verfahren 1.1 (single) mit  "euclidean"
Verfahren1<-Cluster("euclidean","single")
# Cluster Funktion wird durchgeführt und als Verfahren gespeichert
groups <- cutree(Verfahren1, k=3) # Einteilung in 3 Clustern
groups<-Zuordnung(groups,EchteGruppe) # Gruppenzuordnungen werden angepasst
# Das Problem, das die Zuordnungsfunktion notwendig macht, wird in der Arbeit beschrieben
Gruppenmatrix[,1]<-unlist(groups[1]) # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[1]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))
#Die Genauigkeit wird mithilfe der Zuordnungsfunktion und max(V) bestimmt.
#Dazu dividiert man die maximale Anzahl an Übereinstimmung mit der Gesamtanzahl an Merkmalsträgern.
#Genauigkeit wird ermittelt und im Genauigkeitsvektor gespeichert

### Verfahren 1.2 (single) mit  "maximum"
Verfahren2<-Cluster("maximum","single")
groups <- cutree(Verfahren2, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,2]<-unlist(groups[1])
Genauigkeitsvektor[2]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 1.3 (single) mit  "manhattan"
Verfahren3<-Cluster("manhattan","single")
groups <- cutree(Verfahren3, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,3]<-unlist(groups[1])
Genauigkeitsvektor[3]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

###### Complete Linkage

### Verfahren 2.1 (complete) mit  "euclidean"
Verfahren4<-Cluster("euclidean","complete")
groups <- cutree(Verfahren4, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,4]<-unlist(groups[1])
Genauigkeitsvektor[4]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 2.2 (complete) mit  "maximum"
Verfahren5<-Cluster("maximum","complete")
groups <- cutree(Verfahren5, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,5]<-unlist(groups[1])
Genauigkeitsvektor[5]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 2.3 (complete) mit  "manhattan"
Verfahren6<-Cluster("manhattan","complete")
groups <- cutree(Verfahren6, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,6]<-unlist(groups[1])
Genauigkeitsvektor[6]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))
```

```
###### Average Linkage

### Verfahren 3.1 (average) mit Euklid
Verfahren7<-Cluster("euclidean","average")
groups <- cutree(Verfahren7, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,7]<-unlist(groups[1])
Genauigkeitsvektor[7]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 3.2 (average) mit  "maximum"
Verfahren8<-Cluster("maximum","average")
groups <- cutree(Verfahren8, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,8]<-unlist(groups[1])
Genauigkeitsvektor[8]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 3.3 (average) mit  "manhattan"
Verfahren9<-Cluster("manhattan","average")
groups <- cutree(Verfahren9, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,9]<-unlist(groups[1])
Genauigkeitsvektor[9]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

##### Centroid Linkage

### Verfahren 4.1 (centroid) mit  "euclidean"
Verfahren10<-Cluster("euclidean","centroid")
groups <- cutree(Verfahren10, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,10]<-unlist(groups[1])
Genauigkeitsvektor[10]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 4.2 (centroid) mit  "maximum"
Verfahren11<-Cluster("maximum","centroid")
groups <- cutree(Verfahren11, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,11]<-unlist(groups[1])
Genauigkeitsvektor[11]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 4.3 (centroid) mit  "manhattan"
Verfahren12<-Cluster("manhattan","centroid")
groups <- cutree(Verfahren12, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,12]<-unlist(groups[1])
Genauigkeitsvektor[12]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

###### Ward Linkage

# Verfahren 5.1 , euklid und Ward ,
Verfahren13<-Cluster("euclidean","ward")
groups <- cutree(Verfahren13, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,13]<-unlist(groups[1])
Genauigkeitsvektor[13]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 5.2 mit maximum
Verfahren14<-Cluster("maximum","ward")
groups <- cutree(Verfahren14, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,14]<-unlist(groups[1])
Genauigkeitsvektor[14]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 5.3 mit manhattan
Verfahren15<-Cluster("manhattan","ward")
groups <- cutree(Verfahren15, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,15]<-unlist(groups[1])
Genauigkeitsvektor[15]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

###

#############################################################################
#############################################################################

##### Verfahren 16-19 divisives Clustering ######
# Vorsicht! Große Datensätze dauern lange!!!!!!!!

############### Divisiv

### 6.1 mit euklid
Verfahren16 <- diana(x=Datensatzscale2, metric="euclidean")
```

```
groups <- cutree(Verfahren16, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,16]<-unlist(groups[1])
Genauigkeitsvektor[16]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### 6.2 mit maximum
Verfahren17 <- diana(x=Datensatzscale2, metric="maximum")
groups <- cutree(Verfahren17, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,17]<-unlist(groups[1])
Genauigkeitsvektor[17]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### 6.3 manhattan metrik
Verfahren18 <- diana(x=Datensatzscale2, metric="manhattan")
groups <- cutree(Verfahren18, k=3) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,18]<-unlist(groups[1])
Genauigkeitsvektor[18]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

##############################################
##### Partionierende Verfahren Verfahren 19-21
# Quelle: http://www.statmethods.net/advstats/cluster.html
# http://statmath.wu.ac.at/courses/multverf2/tutorien/kmeans.pdf


### Kmeans Verfahren
Verfahren19 <- kmeans(Datensatzscale2, 3, nstart=25, iter.max=50) #
#nstart bedeutet, das mit 25 verschiedenen Startwerten angefangen wird.
#iter max ist die Anzahl an maximal erlaubten Wiederholungen.
# append cluster assignment
groups<-as.vector(Verfahren19$cluster)
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,19]<-unlist(groups[1])
Genauigkeitsvektor[19]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### PAM

### mit Euklid
Verfahren20<-pam(Datensatzscale2,3, metric="euclidean")

# append cluster assignment
groups<-as.vector(Verfahren20$clustering)
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,20]<-unlist(groups[1])
Genauigkeitsvektor[20]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### mit Manhattan
Verfahren21<-pam(Datensatzscale2,3, metric="manhattan")

# append cluster assignment
groups<-as.vector(Verfahren21$clustering)
groups<-Zuordnung(groups,EchteGruppe)
#Funktion zur Bestimmung der Genauigkeit der Clusteranalyse, hier muss groups eingesetzt werden
Gruppenmatrix[,21]<-unlist(groups[1])
Genauigkeitsvektor[21]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

#############################################################################################
##### Modelbasiert Verfahren 22
# http://cran.r-project.org/web/packages/mclust/mclust.pdf
### Model Based Clustering
Verfahren22<-Mclust(Datensatzscale2, G=3) #! Datensatz2ord AmeliaDatensatz2 -> 4 Cluster
groups<-Verfahren22$classification
groups<-Zuordnung(as.integer(groups),EchteGruppe)
Gruppenmatrix[,22]<-unlist(groups[1])
Genauigkeitsvektor[22]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

#####################################################
#####################################################

##### Nun wird die kategoriale Variable berücksichtigt. Die Berechnung basiert
##### bei allen auf der Gower Distanz.

# Datensatzscale3 enthält zusätzlich zu den skalierten numerischen Variablen
# die dritte kategoriale Variable.
Datensatzscale3<-as.matrix(cbind(Datensatzscale2,Datensatz[,3]))


### mit Daisy - Berücksichtigung der 3.Variable, gower metrik
# Hilfsfunktion
Cluster<-function(b){
  Daisy <- daisy(Datensatzscale3[,1:3]) # distance matrix
  Erg <- hclust(Daisy, method=b)
```

```
  Erg
}

### Verfahren 23 aggl. single
Verfahren23<-Cluster("single")
groups <- as.vector(cutree(Verfahren23, k=3)) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,23]<-unlist(groups[1])
Genauigkeitsvektor[23]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 24 aggl. complete
Verfahren24<-Cluster ("complete")
groups <- as.vector(cutree(Verfahren24, k=3)) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,24]<-unlist(groups[1])
Genauigkeitsvektor[24]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))



### Verfahren 25 aggl. Average
Verfahren25<-Cluster ("average")
groups <- as.vector(cutree(Verfahren25, k=3)) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,25]<-unlist(groups[1])
Genauigkeitsvektor[25]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 26 aggl. Centroid
Verfahren26<-Cluster ("centroid")
groups <- as.vector(cutree(Verfahren26, k=3)) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,26]<-unlist(groups[1])
Genauigkeitsvektor[26]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 27 aggl. ward
Verfahren27<-Cluster ("ward")
groups <- as.vector(cutree(Verfahren27, k=3)) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,27]<-unlist(groups[1])
Genauigkeitsvektor[27]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 28 divisiv

Verfahren28 <- diana(daisy(Datensatzscale3))
groups <- as.vector(cutree(Verfahren28, k=3)) # cut tree into 3 clusters
groups<-Zuordnung(groups,EchteGruppe)
Gruppenmatrix[,28]<-unlist(groups[1])
Genauigkeitsvektor[28]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 29 Partionierend

Verfahren29<-pam(daisy(Datensatzscale3),3)
# append cluster assignment
groups<-as.vector(Verfahren29$clustering)
groups<-Zuordnung(groups,EchteGruppe)
#Funktion zur Bestimmung der Genauigkeit der Clusteranalyse, hier muss groups eingesetzt werden
Gruppenmatrix[,29]<-unlist(groups[1])
Genauigkeitsvektor[29]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

######################################################################

# Die einzelnen Verfahren sind durchgeführt, die Genauigkeiten im Genauigkeitsvektor gespeichert.
# Dieser wird nun benannt, um die einzelnen Genauigkeiten den Verfahren zuordnen zu können.

names(Genauigkeitsvektor)<-c("agglomerativ(Euklid/Single)","agglomerativ(Maximum/Single)",
                             "agglomerativ(Manhattan/Single)",
                             "agglomerativ(Euklid/Complete)","agglomerativ(Maximum/Complete)"
                             ,"agglomerativ(Manhattan/Complete)",
                             "agglomerativ(Euklid/Average)","agglomerativ(Maximum/Average)",
                             "agglomerativ(Manhattan/Average)",
                             "agglomerativ(Euklid/Centroid)","agglomerativ(Maximum/Centroid)",
                             "agglomerativ(Manhattan/Centroid)",
                             "agglomerativ(Euklid/Ward)","agglomerativ(Maximum/Ward)",
                             "agglomerativ(Manhattan/Ward)",
                             "divisiv(Euklid)","divisiv(maximum)","divisiv(Manhattan)",
                             "Kmeans(Hartigan)","Kmediods(Euklid)","Kmediods(Manhattan)",
                             "Modelbasiert",
                             "agglomerativ(Gower/Single)","agglomerativ(Gower/Complete)",
                             "agglomerativ(Gower/Average)",
                             "agglomerativ(Gower/Centroid)","agglomerativ(Gower/Ward)"
                             ,"divisiv(Gower)","Kmediods(Gower)")
# Die Objekte der Verfahren (enthalten teilweise releveante Informationen) werden für eventuelle
# spätere Verwendung gespeichert.
```

```
  Verfahren<-list(Verfahren1,Verfahren2,Verfahren3,Verfahren4, Verfahren5,
                  Verfahren7,Verfahren8,Verfahren3,Verfahren9, Verfahren10,
                  Verfahren11,Verfahren12,Verfahren13,Verfahren14, Verfahren15,
                  Verfahren16,Verfahren17,Verfahren18,Verfahren19, Verfahren20,
                  Verfahren21,Verfahren22,Verfahren23,Verfahren24, Verfahren25,
                  Verfahren26,Verfahren27,Verfahren28, Verfahren29)

  list("Genauigkeitsvektor des ursprünglichen Datensatzes"=Genauigkeitsvektor,
       "Gruppenmatrix des ursprünglichen Datensatzes"=Gruppenmatrix,
       "Clusterobjekte des ursprünglichen Datensatzes"=Verfahren)
  # Am Ende wird der Genauigkeitsvektor, die Gruppenmatrix und die einzelnen Verfahren in Form
  # einer Liste gespeichert.
}
#Funktionsende

##############################################################################################
################################## Funktion 5: Clustern2  ####################################

### Clusterfunktion
# Für die Amelia Datensätze ist eine angepasste Clustern Funktion von Nöten.
# Der Grund hierfür ist eine angepasste Zuordnungsfunktion (siehe unten).
# Die Clustern Funktion führt 29 verschiedenene Clusterverfahren mit den Amelia Datensätzen durch.
# Die Verfahren 1-22 werden dabei mit einen um die kategoriale Variable reduzierten Datensatz
# durchgeführt.Zahl gibt an, welcher Amelia Datensatz geclustert werden soll.
# Bei Verfahren 23 bis 29 wird die kategoriale Variable berücksichtigt.
# Clustern2 ist größtenteils wie Clustern aufgebaut.

Clustern2<-function(AmeliaDatensatze, Zahl){

  ######################### Funktion: Clustern############################
  #Quelle http://www.statmethods.net/advstats/cluster.html

  Datensatz1<-AmeliaDatensatze[[1]][[Zahl]]
  #Für die Verfahren 1 bis 22 kann die kategoriale Variable nicht berücksichtigt
  #werden.
  ### Dummys
  # Wahrscheinlichkeitsmatrix: #Dummy:Wird später mit Genauigkeiten gefüllt
  Genauigkeitsvektor<-c(1:29)
  # Gruppenmatrix: # Dummy: wird später mit Gruppenmatrixen gefüllt
  Gruppenmatrix<-matrix(nrow=nrow(Datensatz),ncol=29)

  ### Skalierung des Datensatze
  # skalierter Datensatz # Die Daten müssen standardisiert werden:

  # skalierter Datensatz
  Variable2<-scale(Datensatz1$Variable2)
  Variable1<-scale(Datensatz1$Variable1)

  # Datensatz nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
  # unterschiedlichen Skalen umgehen können
  Datensatzscale2<-cbind(Variable2,Variable1)


  # Datensatzscale2 nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
  # unterschiedlichen Skalen umgehen können
  # Datensatzscale 2 bildet die Grundlage für die Verfahren 1 bis 22.

  ################################## Clusterverfahren ##########################################

  ##### Verfahren 1-15 Agglomeratives Clustering ######

  # Möglichkeiten Distanz:euclidean,maximum, manhattan, ...
  # Möglichkeiten Linkage: Single, complete, average, centroid, ward,..
  # Clusterfunktion: a <- Distanzmetrik,b<-Linkage-Verfahren, mit ""!

  # Hilfsfunktion
  Cluster<-function(a,b){
    d <- dist(Datensatzscale2, method =a) # distance matrix
    Erg <- hclust(d, method=b)
    Erg
  }

# Zuordnung 3 ist größtenteils wie Zuordnung aufgebaut.
# Jedoch werden die levels auf die levels der ursprünglichen
# Clusteranalyse angepasst. Dies ist notwendig, das die
# Auswirkungen der Clusteranalyse erkennbar ist. In den
# meisten Fällen ändern sich die levels allerdings nicht.
# Zudem wird die Genaugikeit direkt mit ausgegeben.
  Zuordnung3<-function(groups,EchteGruppe,Label){
    perms = permutations(3,3)
    EchteGruppe1<-as.factor(as.vector(GruppenmatrixDatensatz[,Label]))
    V<-c(1:6)
    for(i in 1:nrow(perms)){
```

```
    groups1<-as.factor(groups)
    levels(groups1)<-perms[i,]
    VR<-EchteGruppe1==groups1
    V[i]<-length(subset(VR,VR==TRUE))
  }
  S<-which.max(V)
  groups2<-as.factor(groups)
  levels(groups2)<-perms[S,]
  VR<-EchteGruppe==groups2
  Genauigkeit<-(length(subset(VR,VR==TRUE))/((nrow(Datensatz))))
  list(as.numeric(as.vector(groups2)),Genauigkeit)
}
##### Single Linkage

### Verfahren 1.1 (single) mit  "euclidean"
Verfahren1<-Cluster("euclidean","single")
# Cluster Funktion wird durchgeführt und als Verfahren gespeichert
groups <- (cutree(Verfahren1, k=3)) # Einteilung in 3 Clustern
VR<-Zuordnung3(groups,EchteGruppe,1)
#
Gruppenmatrix[,1]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[1]<-VR[[2]] #Genauigkeit aus Zuordnungsfunktion 3

# Genauigkeit wird im Genauigkeitsvektor gespeichert

### Verfahren 1.2 (single) mit  "maximum"
Verfahren2<-Cluster("maximum","single")
groups <- cutree(Verfahren2, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,2)
#
Gruppenmatrix[,2]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[2]<-VR[[2]]

### Verfahren 1.3 (single) mit  "manhattan"
Verfahren3<-Cluster("manhattan","single")
groups <- cutree(Verfahren3, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,3)
#
Gruppenmatrix[,3]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[3]<-VR[[2]]

###### Complete Linkage

### Verfahren 2.1 (complete) mit  "euclidean"
Verfahren4<-Cluster("euclidean","complete")
groups <- cutree(Verfahren4, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,4)
#
Gruppenmatrix[,4]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[4]<-VR[[2]]

### Verfahren 2.2 (complete) mit  "maximum"
Verfahren5<-Cluster("maximum","complete")
groups <- cutree(Verfahren5, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,5)
#
Gruppenmatrix[,5]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[5]<-VR[[2]]

### Verfahren 2.3 (complete) mit  "manhattan"
Verfahren6<-Cluster("manhattan","complete")
groups <- cutree(Verfahren6, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,6)
#
Gruppenmatrix[,6]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[6]<-VR[[2]]
###### Average Linkage

### Verfahren 3.1 (average) mit Euklid
Verfahren7<-Cluster("euclidean","average")
groups <- cutree(Verfahren7, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,7)
#
Gruppenmatrix[,7]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[7]<-VR[[2]]

### Verfahren 3.2 (average) mit  "maximum"
Verfahren8<-Cluster("maximum","average")
groups <- cutree(Verfahren8, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,8)
#
Gruppenmatrix[,8]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[8]<-VR[[2]]
```

```
### Verfahren 3.3 (average) mit  "manhattan"
Verfahren9<-Cluster("manhattan","average")
groups <- cutree(Verfahren9, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,9)
#
Gruppenmatrix[,9]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[9]<-VR[[2]]
##### Centroid Linkage

### Verfahren 4.1 (centroid) mit  "euclidean"
Verfahren10<-Cluster("euclidean","centroid")
groups <- cutree(Verfahren10, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,10)
#
Gruppenmatrix[,10]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[10]<-VR[[2]]
### Verfahren 4.2 (centroid) mit  "maximum"
Verfahren11<-Cluster("maximum","centroid")
groups <- cutree(Verfahren11, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,11)
#
Gruppenmatrix[,11]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[11]<-VR[[2]]

### Verfahren 4.3 (centroid) mit  "manhattan"
Verfahren12<-Cluster("manhattan","centroid")
groups <- cutree(Verfahren12, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,12)
#
Gruppenmatrix[,12]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[12]<-VR[[2]]
###### Ward Linkage

# Verfahren 5.1 , euklid und Ward ,
Verfahren13<-Cluster("euclidean","ward")
groups <- cutree(Verfahren13, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,13)
#
Gruppenmatrix[,13]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[13]<-VR[[2]]
### Verfahren 5.2 mit maximum
Verfahren14<-Cluster("maximum","ward")
groups <- cutree(Verfahren14, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,14)
#
Gruppenmatrix[,14]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[14]<-VR[[2]]
### Verfahren 5.3 mit manhattan
Verfahren15<-Cluster("manhattan","ward")
groups <- cutree(Verfahren15, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,15)
#
Gruppenmatrix[,15]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[15]<-VR[[2]]
###

#############################################################################
#############################################################################

##### Verfahren 16-19 divisives Clustering ######
# Vorsicht! Große Datensätze dauern lange!!!!!!!!
#
############### Divisiv

### 6.1 mit euklid
Verfahren16 <- diana(x=Datensatzscale2, metric="euclidean")
groups <- cutree(Verfahren16, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,16)
#
Gruppenmatrix[,16]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[16]<-VR[[2]]
### 6.2 mit maximum
Verfahren17 <- diana(x=Datensatzscale2, metric="maximum")
groups <- cutree(Verfahren17, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,17)
#
Gruppenmatrix[,17]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[17]<-VR[[2]]
### 6.3 manhattan metrik
Verfahren18 <- diana(x=Datensatzscale2, metric="manhattan")
groups <- cutree(Verfahren18, k=3) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,18)
#
Gruppenmatrix[,18]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
```

```
Genauigkeitsvektor[18]<-VR[[2]]
##############################################
##### Partionierende Verfahren Verfahren 19-21
# Quelle: http://www.statmethods.net/advstats/cluster.html
# http://statmath.wu.ac.at/courses/multverf2/tutorien/kmeans.pdf


### Kmeans Verfahren
Verfahren19 <- kmeans(Datensatzscale2, 3, nstart=25, iter.max=50) #

# append cluster assignment
groups<-as.vector(Verfahren19$cluster)
VR<-Zuordnung3(groups,EchteGruppe,19)
#
Gruppenmatrix[,19]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[19]<-VR[[2]]

### PAM

### mit Euklid
Verfahren20<-pam(Datensatzscale2,3, metric="euclidean")

# append cluster assignment
groups<-as.vector(Verfahren20$clustering)
VR<-Zuordnung3(groups,EchteGruppe,20)
#
Gruppenmatrix[,20]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[20]<-VR[[2]]

### mit Manhattan
Verfahren21<-pam(Datensatzscale2,3, metric="manhattan")

# append cluster assignment
groups<-as.vector(Verfahren21$clustering)
VR<-Zuordnung3(groups,EchteGruppe,21)
#
Gruppenmatrix[,21]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[21]<-VR[[2]]
#############################################################################################
##### Modelbasiert Verfahren 22
# http://cran.r-project.org/web/packages/mclust/mclust.pdf
### Model Based Clustering
Verfahren22<-Mclust(Datensatzscale2, G=3) #! Datensatz2ord AmeliaDatensatz2 -> 4 Cluster
groups<-Verfahren22$classification
VR<-Zuordnung3(groups,EchteGruppe,22)
#
Gruppenmatrix[,22]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[22]<-VR[[2]]
#####################################################
#####################################################

##### Nun wird die kategoriale Variable berücksichtigt. Die Berechnung basiert
##### bei allen auf der Gower Distanz.

#Es müssen die AmeliaDatensätze mit 3 Merkmal ausgewählt werden:
Datensatz1<-AmeliaDatensatze[[2]][[Zahl]]

### Skalierung des Datensatze
# skalierter Datensatz # Die Daten müssen standardisiert werden:

# skalierter Datensatz
Variable2<-scale(Datensatz1$Variable2)
Variable1<-scale(Datensatz1$Variable1)

# Datensatz nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
# unterschiedlichen Skalen umgehen können
Datensatzscale2<-cbind(Variable2,Variable1)
# Datensatzscale3 enthält zusätzlich zu den skalierten numerischen Variablen
# die dritte kategoriale Variable.
Datensatzscale3<-as.data.frame(cbind(Datensatzscale2,(as.factor(Datensatz1[,3]))))
Datensatzscale3[,3]<-Datensatz1[,3]

### mit Daisy - Berücksichtigung der 3.Variable, gower metrik
# Hilfsfunktion
Cluster<-function(b){
  Daisy <- daisy(Datensatzscale3[,1:3]) # distance matrix
  Erg <- hclust(Daisy, method=b)
  Erg
}

### Verfahren 23 aggl. single
Verfahren23<-Cluster("single")
groups <- as.vector(cutree(Verfahren23, k=3)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,23)
```

```
#
Gruppenmatrix[,23]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[23]<-VR[[2]]

### Verfahren 24 aggl. complete
Verfahren24<-Cluster ("complete")
groups <- as.vector(cutree(Verfahren24, k=3)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,24)
#
Gruppenmatrix[,24]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[24]<-VR[[2]]


### Verfahren 25 aggl. Average
Verfahren25<-Cluster ("average")
groups <- as.vector(cutree(Verfahren25, k=3)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,25)
#
Gruppenmatrix[,25]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[25]<-VR[[2]]

### Verfahren 26 aggl. Centroid
Verfahren26<-Cluster ("centroid")
groups <- as.vector(cutree(Verfahren26, k=3)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,26)
#
Gruppenmatrix[,26]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[26]<-VR[[2]]
### Verfahren 27 aggl. ward
Verfahren27<-Cluster ("ward")
groups <- as.vector(cutree(Verfahren27, k=3)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,27)
#
Gruppenmatrix[,27]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[27]<-VR[[2]]
### Verfahren 28 divisiv

Verfahren28 <- diana(daisy(Datensatzscale3))
groups <- as.vector(cutree(Verfahren28, k=3)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,28)
#
Gruppenmatrix[,28]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[28]<-VR[[2]]
### Verfahren 29 Partionierend

Verfahren29<-pam(daisy(Datensatzscale3),3)
# append cluster assignment
groups<-as.vector(Verfahren29$clustering)
VR<-Zuordnung3(groups,EchteGruppe,29)
#
Gruppenmatrix[,29]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[29]<-VR[[2]]
#########################################################################

# Die einzelnen Verfahren sind durchgeführt, die Genauigkeiten im Genauigkeitsvektor gespeichert.
# Dieser wird nun benannt, um die einzelnen Genauigkeiten den Verfahren zuordnen zu können.

names(Genauigkeitsvektor)<-c("agglomerativ(Euklid/Single)","agglomerativ(Maximum/Single)",
                             "agglomerativ(Manhattan/Single)",
                             "agglomerativ(Euklid/Complete)","agglomerativ(Maximum/Complete)"
                             ,"agglomerativ(Manhattan/Complete)",
                             "agglomerativ(Euklid/Average)","agglomerativ(Maximum/Average)",
                             "agglomerativ(Manhattan/Average)",
                             "agglomerativ(Euklid/Centroid)","agglomerativ(Maximum/Centroid)",
                             "agglomerativ(Manhattan/Centroid)",
                             "agglomerativ(Euklid/Ward)","agglomerativ(Maximum/Ward)",
                             "agglomerativ(Manhattan/Ward)",
                             "divisiv(Euklid)","divisiv(maximum)","divisiv(Manhattan)",
                             "Kmeans(Hartigan)","Kmediods(Euklid)","Kmediods(Manhattan)",
                             "Modelbasiert",
                             "agglomerativ(Gower/Single)","agglomerativ(Gower/Complete)",
                             "agglomerativ(Gower/Average)",
                             "agglomerativ(Gower/Centroid)","agglomerativ(Gower/Ward)"
                             ,"divisiv(Gower)","Kmediods(Gower)")
# Die Objekte der Verfahren (enthalten teilweise releveante Informationen) werden für eventuelle
# spätere Verwendung gespeichert.

Verfahren<-list(Verfahren1,Verfahren2,Verfahren3,Verfahren4, Verfahren5,
                Verfahren7,Verfahren8,Verfahren3,Verfahren9, Verfahren10,
                Verfahren11,Verfahren12,Verfahren13,Verfahren14, Verfahren15,
                Verfahren16,Verfahren17,Verfahren18,Verfahren19, Verfahren20,
                Verfahren21,Verfahren22,Verfahren23,Verfahren24, Verfahren25,
                Verfahren26,Verfahren27,Verfahren28, Verfahren29)
```

```
  list("Genauigkeitsvektor des AmeliaDatensatzes"=Genauigkeitsvektor,
       "Gruppenmatrix des AmeliaDatensatzes"=Gruppenmatrix,
        "Clusterobjekte des AmeliaDatensatzes"=Verfahren)
  # Am Ende wird der Genauigkeitsvektor, die Gruppenmatrix und die einzelnen Verfahren in Form
  # einer Liste gespeichert.
}
#Funktionsende


##############################################################################################
################################### Funktion 6: Analysefunktion  ##############################

# Analysefunktion
# Eine  Möglichkeit zur Analyse stellt diese Möglichkeit dar:
# Analyse der imputierten AmeliaDatensätze 1 bis 5
# Hier wird jeder der 5 imputierten AmeliaDatensätze geclustert und die Ergebnisse
# anschließend verglichen bzw. zusammengefasst.


Analysefunktion<-function(DatensatzNA){

  # Die Clustern Funktion für den ursprünglichen Datensatz muss aus einem bestimmten Grund vorab
  # durchgeführt werden. Deshalb ist davon auszugehen, dass das Objekt "ClusterUDatensatz" und
  # die "GruppenmatrixDatensatz" bereits bekannt sind.

  # Die "Clustern2" Funktion wird auf jede der 5 (bzw. 10) Amelia Datensätze angewendet.
  # Man führt die verschiedenen Clusterverfahren also bei jedem Datensatz durch.
  # Die AmeliaDatensatze unterteilen sich jeweils in 2 Amelia Datensätze (mit und ohne kat.
  # Variable) und werden automatisch, je nachdem welcher der beiden benötigt wird, von der
  # Clustern2 Funktion ausgewählt (vgl. Clustern2 Funktion)
  ClusterADatensatz1<-(Clustern2(AmeliaDatensatze,1))
  ClusterADatensatz2<-(Clustern2(AmeliaDatensatze,2))
  ClusterADatensatz3<-(Clustern2(AmeliaDatensatze,3))
  ClusterADatensatz4<-(Clustern2(AmeliaDatensatze,4))
  ClusterADatensatz5<-(Clustern2(AmeliaDatensatze,5))

  # Die Clusterobjekte werden für eventuelle spätere Verwendung gespeichert:
  ClusterObjekte<-list("ClusterObjektA1"=ClusterADatensatz1,
                       "ClusterObjektA2"=ClusterADatensatz2,
                       "ClusterObjektA3"=ClusterADatensatz3,
                       "ClusterObjektA4"=ClusterADatensatz4,
                       "ClusterObjektA5"=ClusterADatensatz5,
                       "ClusterObjektU" =ClusterUDatensatz)

  # Die ClusterADatensätze enthalten die Gruppenmatrizen der AmeliaDatensätze:
  # Die Gruppenmatrizen bilden die Grundlage dieser Funktion.
  GruppenmatrixDatensatz1<-as.data.frame(ClusterADatensatz1[2])
  GruppenmatrixDatensatz2<-as.data.frame(ClusterADatensatz2[2])
  GruppenmatrixDatensatz3<-as.data.frame(ClusterADatensatz3[2])
  GruppenmatrixDatensatz4<-as.data.frame(ClusterADatensatz4[2])
  GruppenmatrixDatensatz5<-as.data.frame(ClusterADatensatz5[2])

  # Die Gruppenmatrizen werden für eventuelle spätere Verwendung gespeichert:
  Gruppenmatrizen<-list("Gruppenmatrix ADatensatz1"=GruppenmatrixDatensatz1,
                        "Gruppenmatrix ADatensatz2"=GruppenmatrixDatensatz2,
                        "Gruppenmatrix ADatensatz3"=GruppenmatrixDatensatz3,
                        "Gruppenmatrix ADatensatz4"=GruppenmatrixDatensatz4,
                        "Gruppenmatrix ADatensatz5"=GruppenmatrixDatensatz5,
                        "Gruppenmatrix UDatensatz"= GruppenmatrixDatensatz)


  # Hilfsfunktion
  # Diese Funktion betrachtet die verschiedenen Verfahren über die 5 Datensätze. Sie gibt dann aus,
  # wie oft die Merkmalsträger richtig eingeteilt wurden in den 5 Datensätzen.

  RSF<-function(MA1){
    L<-(MA1[1:5]==MA1[6])
    length(subset(L,L==TRUE))
    if(length(subset(L,L==TRUE))==0)
    {MA1[7]<-"0"}
    if(length(subset(L,L==TRUE))==1)
    {MA1[7]<-"1"}
    if(length(subset(L,L==TRUE))==2)
    {MA1[7]<-"2"}
    if(length(subset(L,L==TRUE))==3)
    {MA1[7]<-"3"}
    if(length(subset(L,L==TRUE))==4)
    {MA1[7]<-"4"}
    if(length(subset(L,L==TRUE))==5)
    {MA1[7]<-"R"}

    MA1[7]
  }
```

```
########################################
### Komplett
### Die Gruppenmatrizen enthalten die einzelnen Clustervektoren der verschiedenen Verfahren.
### Es werden die Vektoren der einzelnen Verfahren über alle 5 Datensätze hinweg
### verglichen. Für das erste Verfahren wählt man also die erste Spalte der Gruppenmatrixen aus.
### Die 5 Vektoren, die die Gruppenvektoren für die 5 AmeliaDatensatze mit dem jeweiligen
### Verfahren enthalten und der Vektor der echten Gruppe, werden zu einer Matrix zusammengefügt.
### Man erhält also eine Matrix mit 6 Vektoren. In den Zeilen sind die Zuordnungen in den
### verschiedenen AmeliaDatensätzen und die Echte Zuordnung.Bsp
### A1:3 A2:1 A3:3 A4:3 A5:3 EchteGruppe:3. Die RSF Funktion zählt dann in jeder Zeile,
### wie oft die ersten 5 Elemente übereinstimmen, in diesem Beispiel wäre es
### 4mal richtig geclustert worden.
### Für jeden Merkmalsträger erhält man also eine Anzahl, wie oft er richtig
### eingeteilt worden. Die Matrix hat die selbe Dimension wie eine
### der Gruppenmatrixen, nur das anstatt der zugeordneten Gruppe die
### erwähnte Anzahl gegeben ist.

SZ<-GruppenmatrixDatensatz1 # Dummy

for(i in 1:29){
  MA1<-cbind(GruppenmatrixDatensatz1[,i],
             GruppenmatrixDatensatz2[,i],
             GruppenmatrixDatensatz3[,i],
             GruppenmatrixDatensatz4[,i],
             GruppenmatrixDatensatz5[,i],EchteGruppe)

  SZ[,i]<-apply(MA1,1,RSF)
}

# SZ betrachtet die Kompletten Datensätze und zählt, wie oft die Merkmalsträger richtig bzw.
# falsch in den verschiedenen Datensätzen eingeteilt wurden.



##################################################

### Betrachtung der Zeilen mit imputierten Daten
### Es werden nur die jenigen Teile der Datensätze betrachtet, bei den Zeilen imputiert wurden.


#Gruppenmatrizen Verfahren 1 bis 23
#C ist ist ein Vektor mit den Zeilen, in denen Daten imputiert wurden.
C1<-c(as.numeric(rownames(na.omit(DatensatzNA[[1]]))))
C2<-c(1:nrow(Datensatz))
C<-C2[-C1]


# Gruppenmatrizen nur mit imputierten Zeilen
GruppenmatrixImpDatensatz1<-GruppenmatrixDatensatz1[C,]
GruppenmatrixImpDatensatz2<-GruppenmatrixDatensatz2[C,]
GruppenmatrixImpDatensatz3<-GruppenmatrixDatensatz3[C,]
GruppenmatrixImpDatensatz4<-GruppenmatrixDatensatz4[C,]
GruppenmatrixImpDatensatz5<-GruppenmatrixDatensatz5[C,]

# EchteGruppe nur mit imputierten Zeilen
EchteGruppeImp<-EchteGruppe[C]


#Gruppenmatrizen für Verfahren 23 bis 29
#C4 ist ist ein Vektor mit den Zeilen, in denen Daten imputiert wurden.
C1<-c(as.numeric(rownames(na.omit(DatensatzNA[[2]]))))
C2<-c(1:nrow(Datensatz))
C4<-C2[-C1]


# Gruppenmatrizen nur mit imputierten Zeilen
GruppenmatrixImpDatensatz11<-GruppenmatrixDatensatz1[C4,]
GruppenmatrixImpDatensatz21<-GruppenmatrixDatensatz2[C4,]
GruppenmatrixImpDatensatz31<-GruppenmatrixDatensatz3[C4,]
GruppenmatrixImpDatensatz41<-GruppenmatrixDatensatz4[C4,]
GruppenmatrixImpDatensatz51<-GruppenmatrixDatensatz5[C4,]

# EchteGruppe nur mit imputierten Zeilen
EchteGruppeImp1<-EchteGruppe[C4]

####

SZ2<-GruppenmatrixDatensatz1  #Dummy

for(i in 1:29){
  if(i<=22){                                    #Für die Verfahren 1 bis 22
    MA1<-cbind(GruppenmatrixImpDatensatz1[,i],  #werden diese Gruppenmatrizen
               GruppenmatrixImpDatensatz2[,i],  #benötigt
               GruppenmatrixImpDatensatz3[,i],
```

```
              GruppenmatrixImpDatensatz4[,i],
              GruppenmatrixImpDatensatz5[,i],EchteGruppeImp)}

  if(i>22){
    MA1<-cbind(GruppenmatrixImpDatensatz11[,i],   #Für die Verfahren 23 bis 29
              GruppenmatrixImpDatensatz21[,i],   #werden diese Gruppenmatrizen
              GruppenmatrixImpDatensatz31[,i],   #benötigt
              GruppenmatrixImpDatensatz41[,i],
              GruppenmatrixImpDatensatz51[,i],EchteGruppeImp1)}


  M<-apply(MA1,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))

  SZ2[,i]<-MO
}
# SZ2 betrachtet die imputierten Zeilen der Datensätze und zählt, wie oft die Merkmalsträger
# richtig bzw. falsch in den verschiedenen Datensätzen eingeteilt wurden.


################################################

################################################

# Gruppenmatrix des kompletten, ursprünglichen Datensatzes

# Hilfsfunktion
# SC: Einfache Funktion, die einen Vektor ausgibt. Der vektor enthält die Zeilen,
# die richtig geclustert wurden.
SC<-function(x){
  which(x==EchteGruppe)
}
Richtig<-apply(GruppenmatrixDatensatz,2,SC) # SC Funktion wird bei jeder Spalte angewendet.


SZ3<-GruppenmatrixDatensatz1 #Dummy

for(i in 1:29){

  Test<-Richtig[i]
  names(Test)<-NULL
  x<-unlist(Test)
  MA2<-cbind(GruppenmatrixDatensatz1[x,i],
             GruppenmatrixDatensatz2[x,i],
             GruppenmatrixDatensatz3[x,i],
             GruppenmatrixDatensatz4[x,i],
             GruppenmatrixDatensatz5[x,i],EchteGruppe[x])

  M<-apply(MA2,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))
  # Damit wird jeder Vektor automatisch auf die Länge 5000 gebracht
  SZ3[,i]<-MO
}
# SZ3: Es werden nur die Merkmalsträger betrachtet, die beim ursprünglichen Datensatz
#richtig geclustert wurden.
#############################################################################
################################################
# Nun werden nun die Zeilen betrachtet, die ursprünglich falsch geclustert wurden:


# SC2: Einfache Funktion, die einen Vektor ausgibt. Der vektor enthält die Zeilen,
# die falsch geclustert wurden.
SC2<-function(x){
  which((x==EchteGruppe)==FALSE)
}
Falsch<-apply(GruppenmatrixDatensatz,2,SC2) # SC2 Funktion wird bei jeder Spalte angewendet.


SZ4<-GruppenmatrixDatensatz1 #Dummy

for(i in 1:29){

  Test<-Falsch[i]
  names(Test)<-NULL
  x<-unlist(Test)
  MA2<-cbind(GruppenmatrixDatensatz1[x,i],
             GruppenmatrixDatensatz2[x,i],
             GruppenmatrixDatensatz3[x,i],
             GruppenmatrixDatensatz4[x,i],
             GruppenmatrixDatensatz5[x,i],EchteGruppe[x])
  M<-apply(MA2,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))
  # Damit wird jeder Vektor automatisch auf die Länge 5000 gebracht
  SZ4[,i]<-MO
```

```
}
# SZ4: Es werden nur die Merkmalsträger betrachtet, die beim ursprünglichen Datensatz
# falsch geclustert wurden.

################################################################################

# Nun werden die Zeilen betrachtet, die imputiert wurden und gleichzeitig ursprünglich richtig
# geclustert wurden.
SZ5<-GruppenmatrixDatensatz1 #Dummy

for(i in 1:29){
  if(i <=22){
    Test<-Richtig[i]
    names(Test)<-NULL
    x<-unlist(Test)
    x<-c(x,C)
    x<-x[which(duplicated(x))]
    MA2<-cbind(GruppenmatrixDatensatz1[x,i],
               GruppenmatrixDatensatz2[x,i],
               GruppenmatrixDatensatz3[x,i],
               GruppenmatrixDatensatz4[x,i],
               GruppenmatrixDatensatz5[x,i],EchteGruppe[x])}

  if(i >22){
    Test<-Richtig[i]
    names(Test)<-NULL
    x<-unlist(Test)
    x<-c(x,C4)
    x<-x[which(duplicated(x))]
    MA2<-cbind(GruppenmatrixDatensatz1[x,i],
               GruppenmatrixDatensatz2[x,i],
               GruppenmatrixDatensatz3[x,i],
               GruppenmatrixDatensatz4[x,i],
               GruppenmatrixDatensatz5[x,i],EchteGruppe[x])}

  M<-apply(MA2,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))
  # Damit wird jeder Vektor automatisch auf die Länge 5000 gebracht
  SZ5[,i]<-MO
}
# SZ5 enthält Merkmalsträger, die ursprünglich richtig geclustert wurden
# und gleichzeitig imputiert wurden.
################################################################################
################################################################################
# Nun werden die Zeilen betrachtet, die imputiert wurden und gleichzeitig
# ursprünglich falsch geclustert wurden.

SZ6<-GruppenmatrixDatensatz1 #Dummy

for(i in 1:29){
  if(i<=22){
    Test<-Falsch[i]
    names(Test)<-NULL
    x<-unlist(Test)
    x<-c(x,C)
    x<-x[which(duplicated(x))]
    MA2<-cbind(GruppenmatrixDatensatz1[x,i],
               GruppenmatrixDatensatz2[x,i],
               GruppenmatrixDatensatz3[x,i],
               GruppenmatrixDatensatz4[x,i],
               GruppenmatrixDatensatz5[x,i],EchteGruppe[x])}

  if(i>22){
    Test<-Falsch[i]
    names(Test)<-NULL
    x<-unlist(Test)
    x<-c(x,C4)
    x<-x[which(duplicated(x))]
    MA2<-cbind(GruppenmatrixDatensatz1[x,i],
               GruppenmatrixDatensatz2[x,i],
               GruppenmatrixDatensatz3[x,i],
               GruppenmatrixDatensatz4[x,i],
               GruppenmatrixDatensatz5[x,i],EchteGruppe[x])}

  M<-apply(MA2,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))
  # Damit wird jeder Vektor automatisch auf die Länge 500 gebracht
  SZ6[,i]<-MO
}
# SZ6 enthält Merkmalsträger, die ursprünglich richtig geclustert wurden
# und gleichzeitig imputiert wurden.

############################################################
# Funktion zur Erstellung der tables (apply funktioniert nicht)
```

```
Tabelle<-function(SZ){
  Tab<-matrix(NA,nrow=29,ncol=6) #Dummy
  for(i in 1:29){
    Vektor<-factor(SZ[,i], levels=c(0,1,2,3,4,"R"))
    Tab[i,]<-(as.vector(table(Vektor)))
  }
  Tab
}
### Tabelle 1 Betrachtung über alle Merkmalsträger
Table1<-Tabelle(SZ)

# Tabelle 3 Betrachtung über Merkmalsträger, die ursprünglich richtig geclustert wurden:
Table3<-Tabelle(SZ3)
# Tabelle 4 Betrachtung über Merkmalsträger, die ursprünglich falsch geclustert wurden:
Table4<-Tabelle(SZ4)

### Tabelle 2 Betrachtung der imputierten Zeilen
Table2<-Tabelle(SZ2)
# Tabelle 5 Betrachtung der imputierten Zeilen, die ursprünlich richtig geclustert wurden
Table5<-Tabelle(SZ5)
# Tabelle 6 Betrachtung der imputierten Zeilen, die ursprünlich richtig geclustert wurden
Table6<-Tabelle(SZ6)

##########################
### Nun werden die einzelnen Tables gespeichert.
### Für jedes Verfahren wird ausgegeben , wie oft es über die 5 AmeliaDatensätze betrachtet
### richtig geclustert wurde(->Element 1). Dieses Ergebnis wird nochmal zusammengefasst
### (->Element 2).
### Schließlich wird noch mal ein Vektor mit den Genauigkeiten dargestellt(->Element 3).
### Für jedes Verfahren erhält man also eine Liste mit 3 Elementen.

Dummy<-rep( list(list(matrix(nrow=6, ncol=7),matrix(nrow=6, ncol=3)
                      ,matrix(nrow=2,ncol=3),matrix(nrow=1,ncol=(8)))
), 29 )  # Dummy Liste

for(i in 1:29){
  # Element 1 Vergleichstabelle 1
  GR<-t(cbind(Table1[i,],Table3[i,],Table4[i,],
              Table2[i,],Table5[i,],Table6[i,]))
  GR<-cbind(as.vector(c(sum(GR[1,]),sum(GR[2,]),sum(GR[3,]),sum(GR[4,]),sum(GR[5,]),
                        sum(GR[6,]))),GR)
  colnames(GR)<-c("Gesamtanzahl,davon","0xRichtig","1xRichtig","2xRichtig",
                  "3xRichtig","4xRichtig","Immer Richtig")
  rownames(GR)<-c("Gesamt","--davon ursprünglich Richtig geclustert",
                  "--davon ursprünglich Falsch geclustert", "Imputiert",
                  "--davon ursprünglich Richtig geclustert",
                  "--davon ursprünglich Falsch geclustert")
  ##########################################################################################
  # Zusammenfassung <3xRichtig -> Falsch ### >=3xRichtig -> Richtig (Element 2)
  a<-c(sum(GR[1,1]),sum(GR[1,2:4]),sum(GR[1,5:7]))
  b<-c(sum(GR[2,1]),sum(GR[2,2:4]),sum(GR[2,5:7]))
  c<-c(sum(GR[3,1]),sum(GR[3,2:4]),sum(GR[3,5:7]))
  d<-c(sum(GR[4,1]),sum(GR[4,2:4]),sum(GR[4,5:7]))
  e<-c(sum(GR[5,1]),sum(GR[5,2:4]),sum(GR[5,5:7]))
  f<-c(sum(GR[6,1]),sum(GR[6,2:4]),sum(GR[6,5:7]))
  GR2<-t(cbind(a,b,c,d,e,f))
  colnames(GR2)<-c("Gesamtanzahl,davon","-Falsch","-Richtig")
  rownames(GR2)<-c("Gesamt","--davon ursprünglich Richtig geclustert",
                   "--davon ursprünglich Falsch geclustert", "Imputiert",
                   "--davon ursprünglich Richtig geclustert",
                   "--davon ursprünglich Falsch geclustert")



  ######################################################################
  ### Element 3 Vergleichstabelle3
  #Genauigkeiten der AmeliaDatensätze
  Genauigkeiten<-t(as.matrix(c(as.vector(unlist(ClusterUDatensatz[1]))[i],
                               as.vector(unlist(ClusterADatensatz1[1]))[i],
                               as.vector(unlist(ClusterADatensatz2[1]))[i],
                               as.vector(unlist(ClusterADatensatz3[1]))[i],
                               as.vector(unlist(ClusterADatensatz4[1]))[i],
                               as.vector(unlist(ClusterADatensatz5[1]))[i])))

  ##Aus GR2 resultierende Genauigkeit
  Genauigkeit<-GR2[1,3]/GR2[1,1]


  Genauigkeiten<-cbind(Genauigkeiten, mean(Genauigkeiten), Genauigkeit)

  colnames(Genauigkeiten)<-c("UrsprungsGenauigkeit", "AD 1","AD 2",
                             "AD 3",
                             "AD 4","AD5", "Durchschnittlich",
                             "aus Tabelle")
```

```
    rownames(Genauigkeiten)<-"Genauigkeit"
    ######################################################
    # Element 4, Analysetabelle
    Tab<-GR2
    C1<-c(Tab[4,1],Tab[5,3]+Tab[6,2],(Tab[5,3]+Tab[6,2])/Tab[4,1])
    #nicht imputiert
    C2<-c((Tab[1,1]-Tab[4,1]),((Tab[3,2]-Tab[6,2])+(Tab[2,3]-Tab[5,3]))
          ,((Tab[3,2]-Tab[6,2])+(Tab[2,3]-Tab[5,3]))/(Tab[1,1]-Tab[4,1]))

    C<-t(cbind(C1,C2))

    rownames(C)<-c("imputiert","nicht imputiert")
    colnames(C)<-c("Zeilen, davon", "Clusterung wie ursprünglich", "% Übereinstimmung")

    #Schließlich werden diese Elemente in Dummy gespeichert:
    Dummy[[i]]<-list(
      "Vergleichstabelle 1:Anzahl, wie oft die Merkmalsträger über die
      5 Datensätze eingeteilt wurden:"=GR,
      "Vergleichstabelle 2:Zusammenfassung von Tabelle 1:Anzahl<3xRichtig-> falsch,
      Anzahl>=3xRichtig-> richtig"=GR2,
      "Analysetabelle"=C,
      "Genauigkeiten der einzelnen Datensätze sowie abgeleitet aus Tabelle 2"=Genauigkeiten)
  }
  # Benennung von Dummy
  names(Dummy)<-c("agglomerativ(Euklid/Single)","agglomerativ(Maximum/Single)",
                  "agglomerativ(Manhattan/Single)",
                  "agglomerativ(Euklid/Complete)","agglomerativ(Maximum/Complete)",
                  "agglomerativ(Manhattan/Complete)",
                  "agglomerativ(Euklid/Average)","agglomerativ(Maximum/Average)",
                  "agglomerativ(Manhattan/Average)",
                  "agglomerativ(Euklid/Centroid)","agglomerativ(Maximum/Centroid)",
                  "agglomerativ(Manhattan/Centroid)",
                  "agglomerativ(Euklid/Ward)","agglomerativ(Maximum/Ward)",
                  "agglomerativ(Manhattan/Ward)",
                  "divisiv(Euklid)","divisiv(maximum)","divisiv(Manhattan)",
                  "Kmeans(Hartigan)","Kmediods(Euklid)","Kmediods(Manhattan)",
                  "Modelbasiert",
                  "agglomerativ(Gower/Single)","agglomerativ(Gower/Complete)",
                  "agglomerativ(Gower/Average)",
                  "agglomerativ(Gower/Centroid)","agglomerativ(Gower/Ward)"
                  ,"divisiv(Gower)","Kmediods(Gower)")

  list("Vergleichstabellen"=Dummy,"Clusterobjekte"=ClusterObjekte,"Gruppenmatrizen"=Gruppenmatrizen)
  # Schließlich wird Dummy (Liste mit den Elementen 1,2,3 und 4) ausgegeben, sowie für
  # eventuelle spätere Verwendung die Clusterobjekte sowie die Gruppenmatrizen.
}

#Funktionsende

#############################################################################################

#Funktionen für Datensatz 4 und 5

###
# Für die Datensätze 4 bis 5 sind leichte Modifikationen der Funktionen von Nöten, die
# für die Datensätze 1 bis 3 verwendet wurden.
# NAErzeugung2, AmeliaDatensatzErzeugung2                    -> Für Datensatz 4
# Zuordnung2, Clustern3,Clustern4 NAErzeugung2 Analysefunktion2
# AmeliaDatensatzErzeugung2 -> Für Datensatz 5


#############################################################################################################
################################### Funktion 7: Zuordnung2  ##################################################

### Zuordnungsfunktion 2
# für 4 statt 3 Cluster (Datensatz5)
# korrigiert Label Reihenfolge
# Funktionsanfang

Zuordnung2<-function(groups,EchteGruppe){
  perms = permutations(4,4) # 4 statt 3
  EchteGruppe1<-as.factor(EchteGruppe)
  V<-c(1:24) #24 statt 6 Möglichkeiten
  for(i in 1:nrow(perms)){
    groups1<-as.factor(groups)
    levels(groups1)<-perms[i,]
    VR<-EchteGruppe1==groups1
    V[i]<-length(subset(VR,VR==TRUE))
  }
  S<-which.max(V)
  groups<-as.factor(groups)
  levels(groups)<-perms[S,]
  list(as.numeric(as.vector(groups)),max(V))
}
```

```
#Funktionsende

#########################################################################################################
################################## Funktion 8: NAErzeugung2  ############################################

### Funktion zur Erzeugung von "NA" in Datensätzen 4 und 5
# a ist die Anzahl der Daten, die aus dem Datensatz entfernt werden sollen

#Funktionsanfang
NAErzeugung2 <-function (Datensatz,a1){

  # Funktion zur Erzeugung von "NA" in Datensätzen ohne 3. Merkmal
  set.seed(123)
  DA<-Datensatz[,1:2]                       #Nur die ersten beiden Zeilen werden ausgewählt.
  a<-nrow(DA)*ncol(DA)*a1/100               #Ermittlung der Anzahl an Beobachtungen, die
  #entfernt wird.
  FKT<-function(p){                         #Hilfsfunktion, die einfache Münze mit der
    rbinom(1,1,p)                           #Wahrscheinlichkeit p "wirft". Man erhält
  }                                         #1 oder 0 als Resultat.

  #Es werden die Wahrscheinlichkeiten bestimmt, das ein bestimmter Wert fehlt.

  # Wahrscheinlichkeiten
  # Wahrscheinlichkeit p1, das Wert aus X1 fehlt
  Logit<-function(x){
    p1=exp(-1+7*x[2])/(1+exp(-1+7*x[2]))
    p1}

  L1<-apply(DA,1,Logit)
  #<-#Logit Funktion wird auf jede Zeile angwendet. Für jeden Merkmalsträger ist also die Wahr-
  #keit gegeben, das in dem Merkmalsträger die erste Beobachtung fehlt.

  L1<-cbind(as.vector(sapply(L1,FKT)),c(1:length(L1))) #Obige Hilsfunktion FKT wird angewendet.
  L1<-subset(L1,L1[,1]==1)                             #Die Merkmalsträger, in dem die "Münze"
  L1<-L1[,2]                                           #1 ergeben hat, werden ausgewählt.

  #L1 enthält die Merkmalsträger, die potentiell ausfallen könnten in der ersten Variable
  #Analog enthält man L2, das analog die Merkmalsträger für die zweite Variable enthält.

  # Wahrscheinlichkeit, das Wert aus X2 fehlt
  Logit<-function(x){
    p1=exp(-1+7.5*x[1])/(1+exp(-1+7.5*x[1]))
    p1}

  L2<-apply(DA,1,Logit)                              #Analog
  L2<-cbind(as.vector(sapply(L2,FKT)),c(1:length(L2)))
  L2<-subset(L2,L2[,1]==1)
  L2<-L2[,2]

  s<-sample((ncol(DA)),a,replace=TRUE)              #Es wird zufällig ermittelt, wieviele
  s1<-subset(s,s==1)                                #Beobachtungen in den jeweiligen
  s2<-subset(s,s==2)                                #Merkmalsträger fehlen.s1 Variable1
  #s2 Variable 2. Insgesamt sollen a
  #Beobachtungen fehlen.

  C1<-cbind((L1[(sample(1:length(L1),length(s1),replace=FALSE))]),s1)
  #<-#Aus L1 werden zufällig s1 viele Zeilen ausgewählt, die gelöscht werden.

  L2<-L2[ !(L2 %in% (as.numeric(C1[,1])))]
  #<-#Damit keine ganzen Zeilen verschwinden, werden in L2 nur Zeilen ausgewählt
  #  #die in der ersten Variable nicht ausgewählt worden.
  C2<-cbind((L2[(sample(1:length(L2),length(s2),replace=FALSE))]),s2)
  #<-#Aus L2 werden zufällig s2 viele Zeilen ausgewählt, die gelöscht werden.

  D<-rbind(C1,C2) #D enthält die Beobachtungen, die gelöscht werden sollen.

  r1<-D[,1]       #Gibt an, das in der ersten Variable Werte gelöscht werden.
  s1<-D[,2]       #Gibt an, das in der zweiten Variable Werte gelöscht werden.


  for(i in 1:length(r1)){      #Mithilfe dieser Schleife werden die Werte
    r2<-r1[i]                  #gelöscht bzw. NA erzeugt.
    s2<-s1[i]
    is.na(DA[r2,s2])<-TRUE
  }


  # Funktion zur Erzeugung von "NA" in Datensätzen mit 3. Merkmal
  # analoges Vorgehen wie oben beschrieben.
  set.seed(123)
  DA1<-Datensatz[,1:3]
  DA1$Variable3<-as.numeric(DA1$Variable3)
  a<-nrow(DA1)*ncol(DA1)*a1/100
```

```
  # Wahrscheinlichkeiten
  # Wahrscheinlichkeit, das Wert aus X1 fehlt
    Logit<-function(x){
      p1=0.65*x[1]+0.1*x[3]
      p1
    }

  L1<-apply(DA1,1,Logit)
  L1<-cbind(as.vector(sapply(L1,FKT)),c(1:length(L1)))
  L1<-subset(L1,L1[,1]==1)
  L1<-L1[,2]

  # Wahrscheinlichkeit, das Wert aus X2 fehlt
    Logit<-function(x){
      p1=0.70*x[1]+0.1*x[3]
      p1
    }

  L2<-apply(DA1,1,Logit)
  L2<-cbind(as.vector(sapply(L2,FKT)),c(1:length(L2)))
  L2<-subset(L2,L2[,1]==1)
  L2<-L2[,2]

  # Wahrscheinlichkeit, das Wert aus X3 fehlt
    Logit<-function(x){
      p1=0.8*x[1]+0.8*x[2]
      p1
    }

  L3<-apply(DA1,1,Logit)
  L3<-cbind(as.vector(sapply(L3,FKT)),c(1:length(L3)))
  L3<-subset(L3,L3[,1]==1)
  L3<-L3[,2]

  s<-sample((ncol(Datensatz)-1),a,replace=TRUE)

  s1<-subset(s,s==1)
  s2<-subset(s,s==2)
  s3<-subset(s,s==3)

  C1<-cbind((L1[(sample(1:length(L1),length(s1),replace=FALSE))]),s1)
  C2<-cbind((L2[(sample(1:length(L2),length(s2),replace=FALSE))]),s2)

  LOE<-c(C1[,1],C2[,1])
  LOE<-as.numeric(LOE[which(duplicated(LOE))])

  L3<-L3[ !(L3 %in% (LOE))]
  C3<-cbind((L3[(sample(1:length(L3),length(s3),replace=FALSE))]),s3)
  D<-rbind(C1,C2,C3)

  r1<-D[,1]
  s1<-D[,2]


  for(i in 1:length(r1)){
    r2<-r1[i]
    s2<-s1[i]
    is.na(DA1[r2,s2])<-TRUE
  }
  DA1$Variable3<-as.factor(DA1$Variable3) # wird wieder kategorial
  #(Es ist egal, ob nominal oder ordinal, da dies bei der Imputation berücksichtigt wird)
  list(DatensatzNA1=DA, DatensatzNA11=DA1)
  #Beide Datensätze mit NA werden in einer Liste gespeichert.
  #DatensatzNA1 ist der Datensatz ohne 3.Merkmal, DatensatzNA11 mit 3.Merkmal.
}

#Funktionsende

#########################################################################################################
################################### Funktion 9: AmeliaDatensatzErzeugung2  ##############################


### Imputation der fehlenden Werte im DatensatzNA1 mit Amelia #######

### Amelia Daten Imputation für Datensatz 4 und 5
# Die ursprüngliche Funktion muss leicht angepasst werden, da bei Datensatz 4 und 5 auf das
# Intervall 0 bis 1 beschränkt werden sollte.

# Funktionsanfang
AmeliaDatensatzeErzeugung2<-function(DatensatzNA,x="noms"){
  #ohne 3 Variable
  DatensatzNA1<-DatensatzNA[[1]]                #DatensatzNA1 muss ausgewählt werden.
```

```
  Amelia<-amelia(DatensatzNA1,m=5,bound=as.matrix(rbind(c(1,0,1),c(2,0,1))))
  #führt Imputationen durch


  #AmeliaDatensätze 1 bis 5 werden einzelnen gespeichert:
  AmeliaDatensatz1<-Amelia$imputations$imp1
  AmeliaDatensatz2<-Amelia$imputations$imp2
  AmeliaDatensatz3<-Amelia$imputations$imp3
  AmeliaDatensatz4<-Amelia$imputations$imp4
  AmeliaDatensatz5<-Amelia$imputations$imp5


  Ohnekat<-list(AmeliaDatensatz1,AmeliaDatensatz2, AmeliaDatensatz3,
                AmeliaDatensatz4,AmeliaDatensatz5)

  #AmeliaDatensätze ohne 3 Variable werden als Liste in Ohnekat gespeichert.

  #mit 3 Variable
  ### für x muss noms für Nominal oder ords für ordinal eingegeben werden

  DatensatzNA1<-DatensatzNA[[2]] #DatensatzNA11 muss ausgewählt werden.
  # Je nachdem ob die 3 Variable nominal oder ordinal behandelt werden soll:
  if(x=="noms") { # Wenn nominal
    Amelia<-amelia(DatensatzNA1,m=5,noms="Variable3",bound=as.matrix(rbind(c(1,0,1),c(2,0,1))))
    # Imputierte Datensätze - Diese Datensätze sind die Datensätze 1 bis 5, die Amelia Imputiert.
    print("Nominal")}
  if(x=="ords") {# Wenn ordinal
    Amelia<-amelia(DatensatzNA1,m=5,ords="Variable3",bound=as.matrix(rbind(c(1,0,1),c(2,0,1))))
    print("Ordinal")}

  #AmeliaDatensätze 1 bis 5
  AmeliaDatensatz1<-Amelia$imputations$imp1
  AmeliaDatensatz2<-Amelia$imputations$imp2
  AmeliaDatensatz3<-Amelia$imputations$imp3
  AmeliaDatensatz4<-Amelia$imputations$imp4
  AmeliaDatensatz5<-Amelia$imputations$imp5


  Mitkat<-list(AmeliaDatensatz1,AmeliaDatensatz2, AmeliaDatensatz3,
               AmeliaDatensatz4,AmeliaDatensatz5)

  #Die AmeliaDatensätze mit 3. Variable werden als Liste in Mitkat gespeichert.
  #########

  list("DatensatzNa ohne kategoriale Variable"=Ohnekat,"DatensatzNa mit kategoriale Variable"=Mitkat)
  #Mitkat und ohnekat werden gespeichert.
}

#Funktionsende



##########################################################################################################
################################### Funktion 10: Clustern3  ##############################

# Angepasste Clustern3 Funktion für (den ursprünlichen) Datensatz 5
# Hier müssen immer jeweils 4 statt 3 Gruppen betrachtet werden. Dadurch ändern sich die
# Angaben in den einzelnen Verfahren sowie die Zuordnungsfunktion.
# Funktionsanfang
Clustern3<-function(Datensatz){

  ######################### Funktion: Clustern############################
  #Quelle http://www.statmethods.net/advstats/cluster.html

  ### Dummys
  # Wahrscheinlichkeitsmatrix: #Dummy:Wird später mit Genauigkeiten gefüllt
  Genauigkeitsvektor<-c(1:29)
  # Gruppenmatrix: # Dummy: wird später mit Gruppenmatrixen gefüllt
  Gruppenmatrix<-matrix(nrow=nrow(Datensatz),ncol=29)

  ### Skalierung des Datensatze
  # skalierter Datensatz # Die Daten müssen standardisiert werden:

  # skalierter Datensatz
  Variable2<-scale(Datensatz$Variable2)
  Variable1<-scale(Datensatz$Variable1)

  # Datensatz nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
  # unterschiedlichen Skalen umgehen können
  Datensatzscale2<-cbind(Variable2,Variable1)


  # Datensatzscale2 nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
  # unterschiedlichen Skalen umgehen können
```

```
# Datensatzscale 2 bildet die Grundlage für die Verfahren 1 bis 22.

################################## Clusterverfahren ########################################

##### Verfahren 1-15 Agglomeratives Clustering ######

# Möglichkeiten Distanz:euclidean,maximum, manhattan, ...
# Möglichkeiten Linkage: Single, complete, average, centroid, ward,..
# Clusterfunktion: a <- Distanzmetrik,b<-Linkage-Verfahren, mit ""!

# Hilfsfunktion
Cluster<-function(a,b){
  d <- dist(Datensatzscale2, method =a) # distance matrix
  Erg <- hclust(d, method=b)
  Erg
}
##### Single Linkage

### Verfahren 1.1 (single) mit  "euclidean"
Verfahren1<-Cluster("euclidean","single")
# Cluster Funktion wird durchgeführt und als Verfahren gespeichert
groups <- cutree(Verfahren1, k=4) # Einteilung in 3 Clustern
groups<-Zuordnung2(groups,EchteGruppe) # GruppenZuordnungen werden angepasst
Gruppenmatrix[,1]<-unlist(groups[1]) # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[1]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

# Genauigkeit wird ermittelt und im Genauigkeitsvektor gespeichert

### Verfahren 1.2 (single) mit  "maximum"
Verfahren2<-Cluster("maximum","single")
groups <- cutree(Verfahren2, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,2]<-unlist(groups[1])
Genauigkeitsvektor[2]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 1.3 (single) mit  "manhattan"
Verfahren3<-Cluster("manhattan","single")
groups <- cutree(Verfahren3, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,3]<-unlist(groups[1])
Genauigkeitsvektor[3]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

###### Complete Linkage

### Verfahren 2.1 (complete) mit  "euclidean"
Verfahren4<-Cluster("euclidean","complete")
groups <- cutree(Verfahren4, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,4]<-unlist(groups[1])
Genauigkeitsvektor[4]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 2.2 (complete) mit  "maximum"
Verfahren5<-Cluster("maximum","complete")
groups <- cutree(Verfahren5, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,5]<-unlist(groups[1])
Genauigkeitsvektor[5]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 2.3 (complete) mit  "manhattan"
Verfahren6<-Cluster("manhattan","complete")
groups <- cutree(Verfahren6, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,6]<-unlist(groups[1])
Genauigkeitsvektor[6]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

###### Average Linkage

### Verfahren 3.1 (average) mit Euklid
Verfahren7<-Cluster("euclidean","average")
groups <- cutree(Verfahren7, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,7]<-unlist(groups[1])
Genauigkeitsvektor[7]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 3.2 (average) mit  "maximum"
Verfahren8<-Cluster("maximum","average")
groups <- cutree(Verfahren8, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,8]<-unlist(groups[1])
Genauigkeitsvektor[8]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))
```

```
### Verfahren 3.3 (average) mit  "manhattan"
Verfahren9<-Cluster("manhattan","average")
groups <- cutree(Verfahren9, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,9]<-unlist(groups[1])
Genauigkeitsvektor[9]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

##### Centroid Linkage

### Verfahren 4.1 (centroid) mit  "euclidean"
Verfahren10<-Cluster("euclidean","centroid")
groups <- cutree(Verfahren10, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,10]<-unlist(groups[1])
Genauigkeitsvektor[10]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 4.2 (centroid) mit  "maximum"
Verfahren11<-Cluster("maximum","centroid")
groups <- cutree(Verfahren11, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,11]<-unlist(groups[1])
Genauigkeitsvektor[11]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 4.3 (centroid) mit  "manhattan"
Verfahren12<-Cluster("manhattan","centroid")
groups <- cutree(Verfahren12, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,12]<-unlist(groups[1])
Genauigkeitsvektor[12]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

###### Ward Linkage

# Verfahren 5.1 , euklid und Ward ,
Verfahren13<-Cluster("euclidean","ward")
groups <- cutree(Verfahren13, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,13]<-unlist(groups[1])
Genauigkeitsvektor[13]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 5.2 mit maximum
Verfahren14<-Cluster("maximum","ward")
groups <- cutree(Verfahren14, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,14]<-unlist(groups[1])
Genauigkeitsvektor[14]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### Verfahren 5.3 mit manhattan
Verfahren15<-Cluster("manhattan","ward")
groups <- cutree(Verfahren15, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,15]<-unlist(groups[1])
Genauigkeitsvektor[15]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

###

#############################################################################
#############################################################################

##### Verfahren 16-19 divisives Clustering ######
# Vorsicht! Große Datensätze dauern lange!!!!!!!!
#
############### Divisiv

### 6.1 mit euklid
Verfahren16 <- diana(x=Datensatzscale2, metric="euclidean")
groups <- cutree(Verfahren16, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,16]<-unlist(groups[1])
Genauigkeitsvektor[16]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### 6.2 mit maximum
Verfahren17 <- diana(x=Datensatzscale2, metric="maximum")
groups <- cutree(Verfahren17, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,17]<-unlist(groups[1])
Genauigkeitsvektor[17]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

### 6.3 manhattan metrik
Verfahren18 <- diana(x=Datensatzscale2, metric="manhattan")
groups <- cutree(Verfahren18, k=4) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,18]<-unlist(groups[1])
```

```
Genauigkeitsvektor[18]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

##############################################
##### Partionierende Verfahren Verfahren 19-21
# Quelle: http://www.statmethods.net/advstats/cluster.html
# http://statmath.wu.ac.at/courses/multverf2/tutorien/kmeans.pdf


### Kmeans Verfahren
Verfahren19 <- kmeans(Datensatzscale2, 4, nstart=25, iter.max=50) #

# append cluster assignment
groups<-as.vector(Verfahren19$cluster)
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,19]<-unlist(groups[1])
Genauigkeitsvektor[19]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### PAM

### mit Euklid
Verfahren20<-pam(Datensatzscale2,4, metric="euclidean")

# append cluster assignment
groups<-as.vector(Verfahren20$clustering)
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,20]<-unlist(groups[1])
Genauigkeitsvektor[20]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### mit Manhattan
Verfahren21<-pam(Datensatzscale2,4, metric="manhattan")

# append cluster assignment
groups<-as.vector(Verfahren21$clustering)
groups<-Zuordnung2(groups,EchteGruppe)
#Funktion zur Bestimmung der Genauigkeit der Clusteranalyse, hier muss groups eingesetzt werden
Gruppenmatrix[,21]<-unlist(groups[1])
Genauigkeitsvektor[21]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

#############################################################################################
##### Modelbasiert Verfahren 22
# http://cran.r-project.org/web/packages/mclust/mclust.pdf
### Model Based Clustering
Verfahren22<-Mclust(Datensatzscale2, G=4) #! Datensatz2ord AmeliaDatensatz2 -> 4 Cluster
groups<-Verfahren22$classification
groups<-Zuordnung2(as.integer(groups),EchteGruppe)
Gruppenmatrix[,22]<-unlist(groups[1])
Genauigkeitsvektor[22]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

#####################################################
#####################################################

##### Nun wird die kategoriale Variable berücksichtigt. Die Berechnung basiert
##### bei allen auf der Gower Distanz.

# Datensatzscale3 enthält zusätzlich zu den skalierten numerischen Variablen
# die dritte kategoriale Variable.
Datensatzscale3<-as.matrix(cbind(Datensatzscale2,Datensatz[,3]))


### mit Daisy - Berücksichtigung der 3.Variable, gower metrik
# Hilfsfunktion
Cluster<-function(b){
  Daisy <- daisy(Datensatzscale3[,1:3]) # distance matrix
  Erg <- hclust(Daisy, method=b)
  Erg
}

### Verfahren 23 aggl. single
Verfahren23<-Cluster("single")
groups <- as.vector(cutree(Verfahren23, k=4)) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,23]<-unlist(groups[1])
Genauigkeitsvektor[23]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


### Verfahren 24 aggl. complete
Verfahren24<-Cluster ("complete")
groups <- as.vector(cutree(Verfahren24, k=4)) # cut tree into 3 clusters
groups<-Zuordnung2(groups,EchteGruppe)
Gruppenmatrix[,24]<-unlist(groups[1])
Genauigkeitsvektor[24]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))
```

```
  ### Verfahren 25 aggl. Average
  Verfahren25<-Cluster ("average")
  groups <- as.vector(cutree(Verfahren25, k=4)) # cut tree into 3 clusters
  groups<-Zuordnung2(groups,EchteGruppe)
  Gruppenmatrix[,25]<-unlist(groups[1])
  Genauigkeitsvektor[25]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))


  ### Verfahren 26 aggl. Centroid
  Verfahren26<-Cluster ("centroid")
  groups <- as.vector(cutree(Verfahren26, k=4)) # cut tree into 3 clusters
  groups<-Zuordnung2(groups,EchteGruppe)
  Gruppenmatrix[,26]<-unlist(groups[1])
  Genauigkeitsvektor[26]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

  ### Verfahren 27 aggl. ward
  Verfahren27<-Cluster ("ward")
  groups <- as.vector(cutree(Verfahren27, k=4)) # cut tree into 3 clusters
  groups<-Zuordnung2(groups,EchteGruppe)
  Gruppenmatrix[,27]<-unlist(groups[1])
  Genauigkeitsvektor[27]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

  ### Verfahren 28 divisiv

  Verfahren28 <- diana(daisy(Datensatzscale3))
  groups <- as.vector(cutree(Verfahren28, k=4)) # cut tree into 3 clusters
  groups<-Zuordnung2(groups,EchteGruppe)
  Gruppenmatrix[,28]<-unlist(groups[1])
  Genauigkeitsvektor[28]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

  ### Verfahren 29 Partionierend

  Verfahren29<-pam(daisy(Datensatzscale3),4)
  # append cluster assignment
  groups<-as.vector(Verfahren29$clustering)
  groups<-Zuordnung2(groups,EchteGruppe)
  #Funktion zur Bestimmung der Genauigkeit der Clusteranalyse, hier muss groups eingesetzt werden
  Gruppenmatrix[,29]<-unlist(groups[1])
  Genauigkeitsvektor[29]<-(as.numeric(unlist(groups[2]))/nrow(Datensatz))

  ######################################################################

  # Die einzelnen Verfahren sind durchgeführt, die Genauigkeiten im Genauigkeitsvektor gespeichert.
  # Dieser wird nun benannt, um die einzelnen Genauigkeiten den Verfahren zuordnen zu können.

  names(Genauigkeitsvektor)<-c("agglomerativ(Euklid/Single)","agglomerativ(Maximum/Single)",
                               "agglomerativ(Manhattan/Single)",
                               "agglomerativ(Euklid/Complete)","agglomerativ(Maximum/Complete)"
                               ,"agglomerativ(Manhattan/Complete)",
                               "agglomerativ(Euklid/Average)","agglomerativ(Maximum/Average)",
                               "agglomerativ(Manhattan/Average)",
                               "agglomerativ(Euklid/Centroid)","agglomerativ(Maximum/Centroid)",
                               "agglomerativ(Manhattan/Centroid)",
                               "agglomerativ(Euklid/Ward)","agglomerativ(Maximum/Ward)",
                               "agglomerativ(Manhattan/Ward)",
                               "divisiv(Euklid)","divisiv(maximum)","divisiv(Manhattan)",
                               "Kmeans(Hartigan)","Kmediods(Euklid)","Kmediods(Manhattan)",
                               "Modelbasiert",
                               "agglomerativ(Gower/Single)","agglomerativ(Gower/Complete)",
                               "agglomerativ(Gower/Average)",
                               "agglomerativ(Gower/Centroid)","agglomerativ(Gower/Ward)"
                               ,"divisiv(Gower)","Kmediods(Gower)")
  # Die Objekte der Verfahren (enthalten teilweise releveante Informationen) werden für eventuelle
  # spätere Verwendung gespeichert.

  Verfahren<-list(Verfahren1,Verfahren2,Verfahren3,Verfahren4, Verfahren5,
                  Verfahren7,Verfahren8,Verfahren3,Verfahren9, Verfahren10,
                  Verfahren11,Verfahren12,Verfahren13,Verfahren14, Verfahren15,
                  Verfahren16,Verfahren17,Verfahren18,Verfahren19, Verfahren20,
                  Verfahren21,Verfahren22,Verfahren23,Verfahren24, Verfahren25,
                  Verfahren26,Verfahren27,Verfahren28, Verfahren29)

  list("Genauigkeitsvektor des ursprünglichen Datensatzes"=Genauigkeitsvektor,
       "Gruppenmatrix des ursprünglichen Datensatzes"=Gruppenmatrix,
       "Clusterobjekte des ursprünglichen Datensatzes"=Verfahren)
  # Am Ende wird der Genauigkeitsvektor, die Gruppenmatrix und die einzelnen Verfahren in Form
  # einer Liste gespeichert.
}
#Funktionsende

##############################################################################################
################################### Funktion 11: Clustern4  ##################################
```

```
### Clusterfunktion
# Clustern4 für AmeliaDatensätze von Datensatz 5
# Auch hier sind wieder Modifikationen für die Amelia Datensätze nötig.
# Die Clustern Funktion führt 29 verschiedenene Clusterverfahren mit den Amelia Datensätzen durch.
# Die Verfahren 1-22 werden dabei mit einen um die kategoriale Variable reduzierten Datensatz
# durchgeführt.
# Bei Verfahren 23 bis 29 wird die kategoriale Variable berücksichtigt.
# In der Gruppenmatrix sind die Gruppenvektoren gespeichert.
# Der Genauigkeitsvektor speichert die Genauigkeiten der einzelnen Verfahren.

Clustern4<-function(AmeliaDatensatze, Zahl){

  ######################### Funktion: Clustern############################
  #Quelle http://www.statmethods.net/advstats/cluster.html

  Datensatz1<-AmeliaDatensatze[[1]][[Zahl]]
  ### Dummys
  # Wahrscheinlichkeitsmatrix: #Dummy:Wird später mit Genauigkeiten gefüllt
  Genauigkeitsvektor<-c(1:29)
  # Gruppenmatrix: # Dummy: wird später mit Gruppenmatrixen gefüllt
  Gruppenmatrix<-matrix(nrow=nrow(Datensatz),ncol=29)

  ### Skalierung des Datensatze
  # skalierter Datensatz # Die Daten müssen standardisiert werden:

  # skalierter Datensatz
  Variable2<-scale(Datensatz1$Variable2)
  Variable1<-scale(Datensatz1$Variable1)

  # Datensatz nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
  # unterschiedlichen Skalen umgehen können
  Datensatzscale2<-cbind(Variable2,Variable1)


  # Datensatzscale2 nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
  # unterschiedlichen Skalen umgehen können
  # Datensatzscale 2 bildet die Grundlage für die Verfahren 1 bis 22.

  ################################## Clusterverfahren ########################################

  ##### Verfahren 1-15 Agglomeratives Clustering ######

  # Möglichkeiten Distanz:euclidean,maximum, manhattan, ...
  # Möglichkeiten Linkage: Single, complete, average, centroid, ward,..
  # Clusterfunktion: a <- Distanzmetrik,b<-Linkage-Verfahren, mit ""!

  # Hilfsfunktion
  Cluster<-function(a,b){
    d <- dist(Datensatzscale2, method =a) # distance matrix
    Erg <- hclust(d, method=b)
    Erg
  }

  Zuordnung3<-function(groups,EchteGruppe,Label){
    perms = permutations(4,4)
    EchteGruppe1<-as.factor((GruppenmatrixDatensatz[,Label]))
    V<-c(1:24)
    for(i in 1:nrow(perms)){
      groups1<-as.factor(groups)
      levels(groups1)<-perms[i,]
      VR<-EchteGruppe1==groups1
      V[i]<-length(subset(VR,VR==TRUE))
    }
    S<-which.max(V)
    groups2<-as.factor(groups)
    levels(groups2)<-perms[S,]
    VR<-EchteGruppe==groups2
    Genauigkeit<-(length(subset(VR,VR==TRUE))/((nrow(Datensatz))))
    list(as.numeric(as.vector(groups2)),Genauigkeit)
  }
  ##### Single Linkage

  ### Verfahren 1.1 (single) mit  "euclidean"
  Verfahren1<-Cluster("euclidean","single")
  # Cluster Funktion wird durchgeführt und als Verfahren gespeichert
  groups <- (cutree(Verfahren1, k=4)) # Einteilung in 3 Clustern
  VR<-Zuordnung3(groups,EchteGruppe,1)
  #
  Gruppenmatrix[,1]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
  Genauigkeitsvektor[1]<-VR[[2]]

  # Genauigkeit wird ermittelt und im Genauigkeitsvektor gespeichert
```

```
### Verfahren 1.2 (single) mit  "maximum"
Verfahren2<-Cluster("maximum","single")
groups <- cutree(Verfahren2, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,2)
#
Gruppenmatrix[,2]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[2]<-VR[[2]]

### Verfahren 1.3 (single) mit  "manhattan"
Verfahren3<-Cluster("manhattan","single")
groups <- cutree(Verfahren3, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,3)
#
Gruppenmatrix[,3]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[3]<-VR[[2]]

###### Complete Linkage

### Verfahren 2.1 (complete) mit  "euclidean"
Verfahren4<-Cluster("euclidean","complete")
groups <- cutree(Verfahren4, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,4)
#
Gruppenmatrix[,4]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[4]<-VR[[2]]

### Verfahren 2.2 (complete) mit  "maximum"
Verfahren5<-Cluster("maximum","complete")
groups <- cutree(Verfahren5, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,5)
#
Gruppenmatrix[,5]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[5]<-VR[[2]]

### Verfahren 2.3 (complete) mit  "manhattan"
Verfahren6<-Cluster("manhattan","complete")
groups <- cutree(Verfahren6, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,6)
#
Gruppenmatrix[,6]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[6]<-VR[[2]]
###### Average Linkage

### Verfahren 3.1 (average) mit Euklid
Verfahren7<-Cluster("euclidean","average")
groups <- cutree(Verfahren7, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,7)
#
Gruppenmatrix[,7]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[7]<-VR[[2]]

### Verfahren 3.2 (average) mit  "maximum"
Verfahren8<-Cluster("maximum","average")
groups <- cutree(Verfahren8, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,8)
#
Gruppenmatrix[,8]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[8]<-VR[[2]]

### Verfahren 3.3 (average) mit  "manhattan"
Verfahren9<-Cluster("manhattan","average")
groups <- cutree(Verfahren9, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,9)
#
Gruppenmatrix[,9]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[9]<-VR[[2]]
##### Centroid Linkage

### Verfahren 4.1 (centroid) mit  "euclidean"
Verfahren10<-Cluster("euclidean","centroid")
groups <- cutree(Verfahren10, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,10)
#
Gruppenmatrix[,10]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[10]<-VR[[2]]
### Verfahren 4.2 (centroid) mit  "maximum"
Verfahren11<-Cluster("maximum","centroid")
groups <- cutree(Verfahren11, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,11)
#
Gruppenmatrix[,11]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[11]<-VR[[2]]

### Verfahren 4.3 (centroid) mit  "manhattan"
```

```
Verfahren12<-Cluster("manhattan","centroid")
groups <- cutree(Verfahren12, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,12)
#
Gruppenmatrix[,12]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[12]<-VR[[2]]
###### Ward Linkage

# Verfahren 5.1 , euklid und Ward ,
Verfahren13<-Cluster("euclidean","ward")
groups <- cutree(Verfahren13, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,13)
#
Gruppenmatrix[,13]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[13]<-VR[[2]]
### Verfahren 5.2 mit maximum
Verfahren14<-Cluster("maximum","ward")
groups <- cutree(Verfahren14, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,14)
#
Gruppenmatrix[,14]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[14]<-VR[[2]]
### Verfahren 5.3 mit manhattan
Verfahren15<-Cluster("manhattan","ward")
groups <- cutree(Verfahren15, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,15)
#
Gruppenmatrix[,15]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[15]<-VR[[2]]
###

#############################################################################
#############################################################################

##### Verfahren 16-19 divisives Clustering ######
# Vorsicht! Große Datensätze dauern lange!!!!!!!!
#
############### Divisiv

### 6.1 mit euklid
Verfahren16 <- diana(x=Datensatzscale2, metric="euclidean")
groups <- cutree(Verfahren16, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,16)
#
Gruppenmatrix[,16]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[16]<-VR[[2]]
### 6.2 mit maximum
Verfahren17 <- diana(x=Datensatzscale2, metric="maximum")
groups <- cutree(Verfahren17, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,17)
#
Gruppenmatrix[,17]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[17]<-VR[[2]]
### 6.3 manhattan metrik
Verfahren18 <- diana(x=Datensatzscale2, metric="manhattan")
groups <- cutree(Verfahren18, k=4) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,18)
#
Gruppenmatrix[,18]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[18]<-VR[[2]]
#############################################
##### Partionierende Verfahren Verfahren 19-21
# Quelle: http://www.statmethods.net/advstats/cluster.html
# http://statmath.wu.ac.at/courses/multverf2/tutorien/kmeans.pdf


### Kmeans Verfahren
Verfahren19 <- kmeans(Datensatzscale2, 4, nstart=25, iter.max=50) #

# append cluster assignment
groups<-as.vector(Verfahren19$cluster)
VR<-Zuordnung3(groups,EchteGruppe,19)
#
Gruppenmatrix[,19]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[19]<-VR[[2]]

### PAM

### mit Euklid
Verfahren20<-pam(Datensatzscale2,4, metric="euclidean")

# append cluster assignment
groups<-as.vector(Verfahren20$clustering)
VR<-Zuordnung3(groups,EchteGruppe,20)
```

```
#
Gruppenmatrix[,20]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[20]<-VR[[2]]

### mit Manhattan
Verfahren21<-pam(Datensatzscale2,4, metric="manhattan")

# append cluster assignment
groups<-as.vector(Verfahren21$clustering)
VR<-Zuordnung3(groups,EchteGruppe,21)
#
Gruppenmatrix[,21]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[21]<-VR[[2]]
################################################################################
##### Modelbasiert Verfahren 22
# http://cran.r-project.org/web/packages/mclust/mclust.pdf
### Model Based Clustering
Verfahren22<-Mclust(Datensatzscale2, G=4) #! Datensatz2ord AmeliaDatensatz2 -> 4 Cluster
groups<-Verfahren22$classification
VR<-Zuordnung3(groups,EchteGruppe,22)
#
Gruppenmatrix[,22]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[22]<-VR[[2]]
#####################################################
#####################################################

##### Nun wird die kategoriale Variable berücksichtigt. Die Berechnung basiert
##### bei allen auf der Gower Distanz.

#Es müssen die AmeliaDatensätze mit 3 Merkmal ausgewählt werden:
Datensatz1<-AmeliaDatensatze[[2]][[Zahl]]

### Skalierung des Datensatze
# skalierter Datensatz # Die Daten müssen standardisiert werden:

# skalierter Datensatz
Variable2<-scale(Datensatz1$Variable2)
Variable1<-scale(Datensatz1$Variable1)

# Datensatz nur mit numerischen Werten -> notwendig, weil nicht alle Verfahren mit
# unterschiedlichen Skalen umgehen können
Datensatzscale2<-cbind(Variable2,Variable1)
# Datensatzscale3 enthält zusätzlich zu den skalierten numerischen Variablen
# die dritte kategoriale Variable.
Datensatzscale3<-as.data.frame(cbind(Datensatzscale2,(as.factor(Datensatz1[,3]))))
Datensatzscale3[,3]<-Datensatz1[,3]


### mit Daisy - Berücksichtigung der 3.Variable, gower metrik
# Hilfsfunktion
Cluster<-function(b){
  Daisy <- daisy(Datensatzscale3[,1:3]) # distance matrix
  Erg <- hclust(Daisy, method=b)
  Erg
}

### Verfahren 23 aggl. single
Verfahren23<-Cluster("single")
groups <- as.vector(cutree(Verfahren23, k=4)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,1)
#
Gruppenmatrix[,23]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[23]<-VR[[2]]

### Verfahren 24 aggl. complete
Verfahren24<-Cluster ("complete")
groups <- as.vector(cutree(Verfahren24, k=4)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,24)
#
Gruppenmatrix[,24]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[24]<-VR[[2]]


### Verfahren 25 aggl. Average
Verfahren25<-Cluster ("average")
groups <- as.vector(cutree(Verfahren25, k=4)) # cut tree into 3 clusters
VR<-Zuordnung3(groups,EchteGruppe,25)
#
Gruppenmatrix[,25]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
Genauigkeitsvektor[25]<-VR[[2]]

### Verfahren 26 aggl. Centroid
Verfahren26<-Cluster ("centroid")
groups <- as.vector(cutree(Verfahren26, k=4)) # cut tree into 3 clusters
```

```
  VR<-Zuordnung3(groups,EchteGruppe,26)
  #
  Gruppenmatrix[,26]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
  Genauigkeitsvektor[26]<-VR[[2]]
  ### Verfahren 27 aggl. ward
  Verfahren27<-Cluster ("ward")
  groups <- as.vector(cutree(Verfahren27, k=4)) # cut tree into 3 clusters
  VR<-Zuordnung3(groups,EchteGruppe,27)
  #
  Gruppenmatrix[,27]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
  Genauigkeitsvektor[27]<-VR[[2]]
  ### Verfahren 28 divisiv

  Verfahren28 <- diana(daisy(Datensatzscale3))
  groups <- as.vector(cutree(Verfahren28, k=4)) # cut tree into 3 clusters
  VR<-Zuordnung3(groups,EchteGruppe,28)
  #
  Gruppenmatrix[,28]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
  Genauigkeitsvektor[28]<-VR[[2]]
  ### Verfahren 29 Partionierend

  Verfahren29<-pam(daisy(Datensatzscale3),4)
  # append cluster assignment
  groups<-as.vector(Verfahren29$clustering)
  VR<-Zuordnung3(groups,EchteGruppe,29)
  #
  Gruppenmatrix[,29]<-VR[[1]] # Vektor mit Gruppenzuordnung wird in Gruppenmatrix gespeichert
  Genauigkeitsvektor[29]<-VR[[2]]
  ########################################################################

  # Die einzelnen Verfahren sind durchgeführt, die Genauigkeiten im Genauigkeitsvektor gespeichert.
  # Dieser wird nun benannt, um die einzelnen Genauigkeiten den Verfahren zuordnen zu können.

  names(Genauigkeitsvektor)<-c("agglomerativ(Euklid/Single)","agglomerativ(Maximum/Single)",
                               "agglomerativ(Manhattan/Single)",
                               "agglomerativ(Euklid/Complete)","agglomerativ(Maximum/Complete)"
                               ,"agglomerativ(Manhattan/Complete)",
                               "agglomerativ(Euklid/Average)","agglomerativ(Maximum/Average)",
                               "agglomerativ(Manhattan/Average)",
                               "agglomerativ(Euklid/Centroid)","agglomerativ(Maximum/Centroid)",
                               "agglomerativ(Manhattan/Centroid)",
                               "agglomerativ(Euklid/Ward)","agglomerativ(Maximum/Ward)",
                               "agglomerativ(Manhattan/Ward)",
                               "divisiv(Euklid)","divisiv(maximum)","divisiv(Manhattan)",
                               "Kmeans(Hartigan)","Kmediods(Euklid)","Kmediods(Manhattan)",
                               "Modelbasiert",
                               "agglomerativ(Gower/Single)","agglomerativ(Gower/Complete)",
                               "agglomerativ(Gower/Average)",
                               "agglomerativ(Gower/Centroid)","agglomerativ(Gower/Ward)"
                               ,"divisiv(Gower)","Kmediods(Gower)")
  # Die Objekte der Verfahren (enthalten teilweise releveante Informationen) werden für eventuelle
  # spätere Verwendung gespeichert.

  Verfahren<-list(Verfahren1,Verfahren2,Verfahren3,Verfahren4, Verfahren5,
                  Verfahren7,Verfahren8,Verfahren3,Verfahren9, Verfahren10,
                  Verfahren11,Verfahren12,Verfahren13,Verfahren14, Verfahren15,
                  Verfahren16,Verfahren17,Verfahren18,Verfahren19, Verfahren20,
                  Verfahren21,Verfahren22,Verfahren23,Verfahren24, Verfahren25,
                  Verfahren26,Verfahren27,Verfahren28, Verfahren29)

  list("Genauigkeitsvektor des AmeliaDatensatzes"=Genauigkeitsvektor,
       "Gruppenmatrix des AmeliaDatensatzes"=Gruppenmatrix,
       "Clusterobjekte des AmeliaDatensatzes"=Verfahren)
  # Am Ende wird der Genauigkeitsvektor, die Gruppenmatrix und die einzelnen Verfahren in Form
  # einer Liste gespeichert.
}
#Funktionsende

##########################################################################################################
################################### Funktion 12: Analysefunktion2  ###########################

# Analysefunktion für Datensatz 5
# Da sich die Clusternfunktionen verändert haben, muss auch die Analysefunktion verändert
# werden. Ansonsten funktioniert die Funktion analog zur Analysefunktion.

Analysefunktion2<-function(DatensatzNA){
  # Die "Clustern" Funktion wird auf den ursprünglichen Datensatz angewendet:

  # Die "Clustern2" Funktion wird auf jede der 5 Datensätze angewendet. Man führt die verschiedenen
  # Clusterverfahren also bei jedem Datensatz durch.
  ClusterADatensatz1<-(Clustern4(AmeliaDatensatze,1))
  ClusterADatensatz2<-(Clustern4(AmeliaDatensatze,2))
  ClusterADatensatz3<-(Clustern4(AmeliaDatensatze,3))
  ClusterADatensatz4<-(Clustern4(AmeliaDatensatze,4))
```

```
ClusterADatensatz5<-(Clustern4(AmeliaDatensatze,5))

# Die Clusterobjekte werden für eventuelle spätere Verwendung gespeichert:
ClusterObjekte<-list("ClusterObjektA1"=ClusterADatensatz1,
                     "ClusterObjektA2"=ClusterADatensatz2,
                     "ClusterObjektA3"=ClusterADatensatz3,
                     "ClusterObjektA4"=ClusterADatensatz4,
                     "ClusterObjektA5"=ClusterADatensatz5,
                     "ClusterObjektU" =ClusterUDatensatz)



# Die ClusterADatensätze enthalten die Gruppenmatrizen der AmeliaDatensätze:
# Die Gruppenmatrizen bilden die Grundlage dieser Funktion.
GruppenmatrixDatensatz1<-as.data.frame(ClusterADatensatz1[2])
GruppenmatrixDatensatz2<-as.data.frame(ClusterADatensatz2[2])
GruppenmatrixDatensatz3<-as.data.frame(ClusterADatensatz3[2])
GruppenmatrixDatensatz4<-as.data.frame(ClusterADatensatz4[2])
GruppenmatrixDatensatz5<-as.data.frame(ClusterADatensatz5[2])

# Die Gruppenmatrizen werden für eventuelle spätere Verwendung gespeichert:
Gruppenmatrizen<-list("Gruppenmatrix ADatensatz1"=GruppenmatrixDatensatz1,
                      "Gruppenmatrix ADatensatz2"=GruppenmatrixDatensatz2,
                      "Gruppenmatrix ADatensatz3"=GruppenmatrixDatensatz3,
                      "Gruppenmatrix ADatensatz4"=GruppenmatrixDatensatz4,
                      "Gruppenmatrix ADatensatz5"=GruppenmatrixDatensatz5,
                      "Gruppenmatrix UDatensatz"= GruppenmatrixDatensatz)


# Hilfsfunktion
# Diese Funktion betrachtet die verschiedenen Verfahren über die 5 Datensätze. Sie gibt dann aus,
# wie oft die Merkmalsträger richtig eingeteilt wurden in den 5 Datensätzen.

RSF<-function(MA1){
  L<-(MA1[1:5]==MA1[6])
  length(subset(L,L==TRUE))
  if(length(subset(L,L==TRUE))==0)
  {MA1[7]<-"0"}
  if(length(subset(L,L==TRUE))==1)
  {MA1[7]<-"1"}
  if(length(subset(L,L==TRUE))==2)
  {MA1[7]<-"2"}
  if(length(subset(L,L==TRUE))==3)
  {MA1[7]<-"3"}
  if(length(subset(L,L==TRUE))==4)
  {MA1[7]<-"4"}
  if(length(subset(L,L==TRUE))==5)
  {MA1[7]<-"R"}

  MA1[7]
}

########################################
### Komplett
### Die Gruppenmatrizen enthalten die einzelnen Clustervektoren der verschiedenen Verfahren.
### Es werden die Vektoren der einzelnen Verfahren über alle 5 Datensätze hinweg
### verglichen. Für das erste Verfahren wählt man also die erste Spalte der Gruppenmatrixen aus.
### Die 5 Vektoren, die die Gruppenvektoren für die 5 AmeliaDatensatze mit dem jeweiligen
### Verfahren enthalten und der Vektor der echten Gruppe, werden zu einer Matrix zusammengefügt.
### Man erhält also eine Matrix mit 6 Vektoren. In den Zeilen sind die Zuordnungen in den
### verschiedenen AmeliaDatensätzen und die Echte Zuordnung.Bsp
### A1:3 A2:1 A3:3 A4:3 A5:3 EchteGruppe:3. Die RSF Funktion zählt dann in jeder Zeile,
### wie oft die ersten 5 Elemente übereinstimmen, in diesem Beispiel wäre es
### 4mal richtig geclustert worden.
### Für jeden Merkmalsträger erhält man also eine Anzahl, wie oft er richtig
### eingeteilt worden. Die Matrix hat die selbe Dimension wie eine
### der Gruppenmatrixen, nur das anstatt der zugeordneten Gruppe die
### erwähnte Anzahl gegeben ist.

SZ<-GruppenmatrixDatensatz1 # Dummy

for(i in 1:29){
  MA1<-cbind(GruppenmatrixDatensatz1[,i],
             GruppenmatrixDatensatz2[,i],
             GruppenmatrixDatensatz3[,i],
             GruppenmatrixDatensatz4[,i],
             GruppenmatrixDatensatz5[,i],EchteGruppe)

  SZ[,i]<-apply(MA1,1,RSF)
}

# SZ betrachtet die Kompletten Datensätze und zählt, wie oft die Merkmalsträger richtig bzw.
# falsch in den verschiedenen Datensätzen eingeteilt wurden.
```

```
################################################

### Betrachtung der Zeilen mit imputierten Daten
### Es werden nur die jenigen Teile der Datensätze betrachtet, bei den Zeilen imputiert wurden.


#Gruppenmatrizen Verfahren 1 bis 23
#C ist ist ein Vektor mit den Zeilen, in denen Daten imputiert wurden.
C1<-c(as.numeric(rownames(na.omit(DatensatzNA[[1]]))))
C2<-c(1:nrow(Datensatz))
C<-C2[-C1]


# Gruppenmatrizen nur mit imputierten Zeilen
GruppenmatrixImpDatensatz1<-GruppenmatrixDatensatz1[C,]
GruppenmatrixImpDatensatz2<-GruppenmatrixDatensatz2[C,]
GruppenmatrixImpDatensatz3<-GruppenmatrixDatensatz3[C,]
GruppenmatrixImpDatensatz4<-GruppenmatrixDatensatz4[C,]
GruppenmatrixImpDatensatz5<-GruppenmatrixDatensatz5[C,]

# EchteGruppe nur mit imputierten Zeilen
EchteGruppeImp<-EchteGruppe[C]


#Gruppenmatrizen für Verfahren 23 bis 29
#C4 ist ist ein Vektor mit den Zeilen, in denen Daten imputiert wurden.
C1<-c(as.numeric(rownames(na.omit(DatensatzNA[[2]]))))
C2<-c(1:nrow(Datensatz))
C4<-C2[-C1]


# Gruppenmatrizen nur mit imputierten Zeilen
GruppenmatrixImpDatensatz11<-GruppenmatrixDatensatz1[C4,]
GruppenmatrixImpDatensatz21<-GruppenmatrixDatensatz2[C4,]
GruppenmatrixImpDatensatz31<-GruppenmatrixDatensatz3[C4,]
GruppenmatrixImpDatensatz41<-GruppenmatrixDatensatz4[C4,]
GruppenmatrixImpDatensatz51<-GruppenmatrixDatensatz5[C4,]

# EchteGruppe nur mit imputierten Zeilen
EchteGruppeImp1<-EchteGruppe[C4]

####

SZ2<-GruppenmatrixDatensatz1  #Dummy

for(i in 1:29){
  if(i<=22){                                    #Für die Verfahren 1 bis 22
    MA1<-cbind(GruppenmatrixImpDatensatz1[,i],    #werden diese Gruppenmatrizen
               GruppenmatrixImpDatensatz2[,i],    #benötigt
               GruppenmatrixImpDatensatz3[,i],
               GruppenmatrixImpDatensatz4[,i],
               GruppenmatrixImpDatensatz5[,i],EchteGruppeImp)}

  if(i>22){
    MA1<-cbind(GruppenmatrixImpDatensatz11[,i],   #Für die Verfahren 23 bis 29
               GruppenmatrixImpDatensatz21[,i],   #werden diese Gruppenmatrizen
               GruppenmatrixImpDatensatz31[,i],   #benötigt
               GruppenmatrixImpDatensatz41[,i],
               GruppenmatrixImpDatensatz51[,i],EchteGruppeImp1)}


  M<-apply(MA1,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))

  SZ2[,i]<-MO
}
# SZ2 betrachtet die imputierten Zeilen der Datensätze und zählt, wie oft die Merkmalsträger
# richtig bzw. falsch in den verschiedenen Datensätzen eingeteilt wurden.


################################################

################################################

# Gruppenmatrix des kompletten, ursprünglichen Datensatzes

# Hilfsfunktion
# SC: Einfache Funktion, die einen Vektor ausgibt. Der vektor enthält die Zeilen,
# die richtig geclustert wurden.
SC<-function(x){
  which(x==EchteGruppe)
}
```

```
Richtig<-apply(GruppenmatrixDatensatz,2,SC) # SC Funktion wird bei jeder Spalte angewendet.


SZ3<-GruppenmatrixDatensatz1 #Dummy

for(i in 1:29){

  Test<-Richtig[i]
  names(Test)<-NULL
  x<-unlist(Test)
  MA2<-cbind(GruppenmatrixDatensatz1[x,i],
             GruppenmatrixDatensatz2[x,i],
             GruppenmatrixDatensatz3[x,i],
             GruppenmatrixDatensatz4[x,i],
             GruppenmatrixDatensatz5[x,i],EchteGruppe[x])

  M<-apply(MA2,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))
  # Damit wird jeder Vektor automatisch auf die Länge 500 gebracht
  SZ3[,i]<-MO
}
# SZ3: Es werden nur die Merkmalsträger betrachtet, die beim ursprünglichen Datensatz
#richtig geclustert wurden.
################################################################################
################################################
# Nun werden nun die Zeilen betrachtet, die ursprünglich falsch geclustert wurden:


# SC2: Einfache Funktion, die einen Vektor ausgibt. Der vektor enthält die Zeilen,
# die falsch geclustert wurden.
SC2<-function(x){
  which((x==EchteGruppe)==FALSE)
}
Falsch<-apply(GruppenmatrixDatensatz,2,SC2) # SC2 Funktion wird bei jeder Spalte angewendet.


SZ4<-GruppenmatrixDatensatz1 #Dummy

for(i in 1:29){

  Test<-Falsch[i]
  names(Test)<-NULL
  x<-unlist(Test)
  MA2<-cbind(GruppenmatrixDatensatz1[x,i],
             GruppenmatrixDatensatz2[x,i],
             GruppenmatrixDatensatz3[x,i],
             GruppenmatrixDatensatz4[x,i],
             GruppenmatrixDatensatz5[x,i],EchteGruppe[x])
  M<-apply(MA2,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))
  # Damit wird jeder Vektor automatisch auf die Länge 500 gebracht
  SZ4[,i]<-MO
}
# SZ4: Es werden nur die Merkmalsträger betrachtet, die beim ursprünglichen Datensatz
# falsch geclustert wurden.

################################################################################

# Nun werden die Zeilen betrachtet, die imputiert wurden und gleichzeitig ursprünglich richtig geclustert
# wurden.
SZ5<-GruppenmatrixDatensatz1 #Dummy

for(i in 1:29){
  if(i <=22){
    Test<-Richtig[i]
    names(Test)<-NULL
    x<-unlist(Test)
    x<-c(x,C)
    x<-x[which(duplicated(x))]
    MA2<-cbind(GruppenmatrixDatensatz1[x,i],
               GruppenmatrixDatensatz2[x,i],
               GruppenmatrixDatensatz3[x,i],
               GruppenmatrixDatensatz4[x,i],
               GruppenmatrixDatensatz5[x,i],EchteGruppe[x])}

  if(i >22){
    Test<-Richtig[i]
    names(Test)<-NULL
    x<-unlist(Test)
    x<-c(x,C4)
    x<-x[which(duplicated(x))]
    MA2<-cbind(GruppenmatrixDatensatz1[x,i],
               GruppenmatrixDatensatz2[x,i],
               GruppenmatrixDatensatz3[x,i],
```

```
              GruppenmatrixDatensatz4[x,i],
              GruppenmatrixDatensatz5[x,i],EchteGruppe[x])}

  M<-apply(MA2,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))
  # Damit wird jeder Vektor automatisch auf die Länge 500 gebracht
  SZ5[,i]<-MO
}
# SZ5 enthält Merkmalsträger, die ursprünglich richtig geclustert wurden
# und gleichzeitig imputiert wurden.
##############################################################################
##############################################################################
#Nun werden die Zeilen betrachtet, die imputiert wurden und gleichzeitig
#ursprünglich falsch geclustert wurden.

SZ6<-GruppenmatrixDatensatz1 #Dummy

for(i in 1:29){
  if(i<=22){
    Test<-Falsch[i]
    names(Test)<-NULL
    x<-unlist(Test)
    x<-c(x,C)
    x<-x[which(duplicated(x))]
    MA2<-cbind(GruppenmatrixDatensatz1[x,i],
              GruppenmatrixDatensatz2[x,i],
              GruppenmatrixDatensatz3[x,i],
              GruppenmatrixDatensatz4[x,i],
              GruppenmatrixDatensatz5[x,i],EchteGruppe[x])}

  if(i>22){
    Test<-Falsch[i]
    names(Test)<-NULL
    x<-unlist(Test)
    x<-c(x,C4)
    x<-x[which(duplicated(x))]
    MA2<-cbind(GruppenmatrixDatensatz1[x,i],
              GruppenmatrixDatensatz2[x,i],
              GruppenmatrixDatensatz3[x,i],
              GruppenmatrixDatensatz4[x,i],
              GruppenmatrixDatensatz5[x,i],EchteGruppe[x])}

  M<-apply(MA2,1,RSF)
  MO<-c(M, rep(NA,(nrow(Datensatz)-length(M))))
  # Damit wird jeder Vektor automatisch auf die Länge 500 gebracht
  SZ6[,i]<-MO
}
# SZ6 enthält Merkmalsträger, die ursprünglich richtig geclustert wurden
# und gleichzeitig imputiert wurden.

#############################################################
# Funktion zur Erstellung der tables (apply funktioniert nicht)
Tabelle<-function(SZ){
  Tab<-matrix(NA,nrow=29,ncol=6) #Dummy
  for(i in 1:29){
    Vektor<-factor(SZ[,i], levels=c(0,1,2,3,4,"R"))
    Tab[i,]<-(as.vector(table(Vektor)))
  }
  Tab
}
### Tabelle 1 Betrachtung über alle Merkmalsträger
Table1<-Tabelle(SZ)

# Tabelle 3 Betrachtung über Merkmalsträger, die ursprünglich richtig geclustert wurden:
Table3<-Tabelle(SZ3)
# Tabelle 4 Betrachtung über Merkmalsträger, die ursprünglich falsch geclustert wurden:
Table4<-Tabelle(SZ4)

### Tabelle 2 Betrachtung der imputierten Zeilen
Table2<-Tabelle(SZ2)
# Tabelle 5 Betrachtung der imputierten Zeilen, die ursprünlich richtig geclustert wurden
Table5<-Tabelle(SZ5)
# Tabelle 6 Betrachtung der imputierten Zeilen, die ursprünlich richtig geclustert wurden
Table6<-Tabelle(SZ6)

############################
### Nun werden die einzelnen Tables gespeichert.
### Für jedes Verfahren wird ausgegeben , wie oft es über die 5 AmeliaDatensätze betrachtet
### richtig geclustert wurde(->Element 1). Dieses Ergebnis wird nochmal zusammengefasst
### (->Element 2).
### Schließlich wird noch mal ein Vektor mit den Genauigkeiten dargestellt(->Element 3).
### Für jedes Verfahren erhält man also eine Liste mit 3 Elementen.

Dummy<-rep( list(list(matrix(nrow=6, ncol=7),matrix(nrow=6, ncol=3)
```

```
                    ,matrix(nrow=2,ncol=3),matrix(nrow=1,ncol=(8)))
), 29 )  # Dummy Liste

for(i in 1:29){
  # Element 1 Vergleichstabelle 1
  GR<-t(cbind(Table1[i,],Table3[i,],Table4[i,],
              Table2[i,],Table5[i,],Table6[i,]))
  GR<-cbind(as.vector(c(sum(GR[1,]),sum(GR[2,]),sum(GR[3,]),sum(GR[4,]),sum(GR[5,]),
                        sum(GR[6,]))),GR)
  colnames(GR)<-c("Gesamtanzahl,davon","0xRichtig","1xRichtig","2xRichtig",
                  "3xRichtig","4xRichtig","Immer Richtig")
  rownames(GR)<-c("Gesamt","--davon ursprünglich Richtig geclustert",
                  "--davon ursprünglich Falsch geclustert", "Imputiert",
                  "--davon ursprünglich Richtig geclustert",
                  "--davon ursprünglich Falsch geclustert")
  ##############################################################################################
  # Zusammenfassung <3xRichtig -> Falsch ### >=3xRichtig -> Richtig (Element 2)
  a<-c(sum(GR[1,1]),sum(GR[1,2:4]),sum(GR[1,5:7]))
  b<-c(sum(GR[2,1]),sum(GR[2,2:4]),sum(GR[2,5:7]))
  c<-c(sum(GR[3,1]),sum(GR[3,2:4]),sum(GR[3,5:7]))
  d<-c(sum(GR[4,1]),sum(GR[4,2:4]),sum(GR[4,5:7]))
  e<-c(sum(GR[5,1]),sum(GR[5,2:4]),sum(GR[5,5:7]))
  f<-c(sum(GR[6,1]),sum(GR[6,2:4]),sum(GR[6,5:7]))
  GR2<-t(cbind(a,b,c,d,e,f))
  colnames(GR2)<-c("Gesamtanzahl,davon","-Falsch","-Richtig")
  rownames(GR2)<-c("Gesamt","--davon ursprünglich Richtig geclustert",
                   "--davon ursprünglich Falsch geclustert", "Imputiert",
                   "--davon ursprünglich Richtig geclustert",
                   "--davon ursprünglich Falsch geclustert")



  ####################################################################
  ### Element 3 Vergleichstabelle3
  #Genauigkeiten der AmeliaDatensätze
  Genauigkeiten<-t(as.matrix(c(as.vector(unlist(ClusterUDatensatz[1]))[i],
                               as.vector(unlist(ClusterADatensatz1[1]))[i],
                               as.vector(unlist(ClusterADatensatz2[1]))[i],
                               as.vector(unlist(ClusterADatensatz3[1]))[i],
                               as.vector(unlist(ClusterADatensatz4[1]))[i],
                               as.vector(unlist(ClusterADatensatz5[1]))[i])))

  ##Aus GR2 resultierende Genauigkeit
  Genauigkeit<-GR2[1,3]/GR2[1,1]


  Genauigkeiten<-cbind(Genauigkeiten, mean(Genauigkeiten), Genauigkeit)

  colnames(Genauigkeiten)<-c("UrsprungsGenauigkeit", "AD 1","AD 2",
                             "AD 3",
                             "AD 4","AD5", "Durchschnittlich",
                             "aus Tabelle")
  rownames(Genauigkeiten)<-"Genauigkeit"
  ####################################################
  # Element 4, Analysetabelle
  Tab<-GR2
  C1<-c(Tab[4,1],Tab[5,3]+Tab[6,2],(Tab[5,3]+Tab[6,2])/Tab[4,1])
  #nicht imputiert
  C2<-c((Tab[1,1]-Tab[4,1]),((Tab[3,2]-Tab[6,2])+(Tab[2,3]-Tab[5,3]))
        ,((Tab[3,2]-Tab[6,2])+(Tab[2,3]-Tab[5,3]))/(Tab[1,1]-Tab[4,1]))

  C<-t(cbind(C1,C2))

  rownames(C)<-c("imputiert","nicht imputiert")
  colnames(C)<-c("Zeilen, davon", "Clusterung wie ursprünglich", "% Übereinstimmung")

  #Schließlich werden diese Elemente in Dummy gespeichert:
  Dummy[[i]]<-list(
    "Vergleichstabelle 1:Anzahl, wie oft die Merkmalsträger über die
    5 Datensätze eingeteilt wurden:"=GR,
    "Vergleichstabelle 2:Zusammenfassung von Tabelle 1:Anzahl<3xRichtig-> falsch,
    Anzahl>=3xRichtig-> richtig"=GR2,
    "Analysetabelle"=C,
    "Genauigkeiten der einzelnen Datensätze sowie abgeleitet aus Tabelle 2"=Genauigkeiten)
}
# Benennung von Dummy
names(Dummy)<-c("agglomerativ(Euklid/Single)","agglomerativ(Maximum/Single)",
                "agglomerativ(Manhattan/Single)",
                "agglomerativ(Euklid/Complete)","agglomerativ(Maximum/Complete)",
                "agglomerativ(Manhattan/Complete)",
                "agglomerativ(Euklid/Average)","agglomerativ(Maximum/Average)",
                "agglomerativ(Manhattan/Average)",
                "agglomerativ(Euklid/Centroid)","agglomerativ(Maximum/Centroid)",
                "agglomerativ(Manhattan/Centroid)",
```

```
                "agglomerativ(Euklid/Ward)","agglomerativ(Maximum/Ward)",
                "agglomerativ(Manhattan/Ward)",
                "divisiv(Euklid)","divisiv(maximum)","divisiv(Manhattan)",
                "Kmeans(Hartigan)","Kmediods(Euklid)","Kmediods(Manhattan)",
                "Modelbasiert",
                "agglomerativ(Gower/Single)","agglomerativ(Gower/Complete)",
                "agglomerativ(Gower/Average)",
                "agglomerativ(Gower/Centroid)","agglomerativ(Gower/Ward)"
                ,"divisiv(Gower)","Kmediods(Gower)")

  list("Vergleichstabellen"=Dummy,"Clusterobjekte"=ClusterObjekte,"Gruppenmatrizen"=Gruppenmatrizen)
  # Schließlich wird Dummy (Liste mit den Elementen 1,2,3 und 4) ausgegeben, sowie für
  # eventuelle spätere Verwendung die Clusterobjekte sowie die Gruppenmatrizen.
}

#Funktionsende

##############################################################################################
```

B.3 Durchführung

Hier werden letztendlich die Funktionen auf die verschiedenen Datensätze angewendet.

```
#Teil 3 Durchführung

#############################################################################################################
###################################---- Datensatz 1 bis 3 - 30 % ########################################

### Analyse der Datensätze Datensatz1ord, Datensatz2ord und Datensatz3nom mit 30 % Fehlenden Daten
### Wieviel Daten sollen entfernt werden
AnzahlentfernterWerte<-30 #Entspricht 30 %

#########################################################################
#Datensatz 1 mit ordinalen Merkmal und 30 % entfernten Daten
#########################################################################
### Datensatz ist der Datensatz, bei dem später fehlende Werte erzeugt werden.
### Er wird deshalb öfter als der "ursprüngliche" Datensatz bezeichnet.
Datensatz<-Datensatz1ord
EchteGruppe<-Datensatz[,4]

#set.seed für Reproduzierbarkeit
set.seed(123)

### DatensatzNA
DatensatzNA<-(NAErzeugung(Datensatz,AnzahlentfernterWerte))

### Amelia Daten imputation
AmeliaDatensatze<-AmeliaDatensatzeErzeugung(DatensatzNA,x="ords")
# AmeliaDatensatze enthält eine Liste mit 6 Datensätzen. Die ersten 5 Datensätze (AmeliaDatensatz 1 bis 5)
# enthalten verschiedenen imputierten Datensätze. Der 6.te Datensatz (AmeliaDatensatzCom) stellt die
# Kombiniation mittels den Rubin Schätzer dar. Die imputierten Datensätze werden als
# "AmeliaDatensätze" bezeichnet.

##
ClusterUDatensatz<-Clustern1(Datensatz)
# ClusterUDatensatz enthält Gruppenmatrix des ursprünglichen Datensatzes
GruppenmatrixDatensatz<-as.data.frame(ClusterUDatensatz[2])
##

# Analyse der imputierten AmeliaDatensätze 1 bis 5
AnalyseDatensatz1<-Analysefunktion(DatensatzNA)
# Dieses Objekt enthält alle relevanten Informationen. Eine genauere Erläuterung dieses
# Objekts ist weiter unten zu finden.

# Speichern der AmeliaDatensätze und DatensatzNA
AmeliaDatensatze1<-AmeliaDatensatze
DatensatzNAD1<-DatensatzNA


#############################################################################################################
#Datensatz 2 mit ordinalen Merkmal und 30 % entfernten Daten
#########################################################################
### Datensatz ist der Datensatz, bei dem später fehlende Werte erzeugt werden.
### Er wird deshalb öfter als der "ursprüngliche" Datensatz bezeichnet.
Datensatz<-Datensatz2ord
EchteGruppe<-Datensatz[,4]

#set.seed für Reproduzierbarkeit
set.seed(123)
```

```
### DatensatzNA
DatensatzNA<-(NAErzeugung(Datensatz,AnzahlentfernterWerte))

### Amelia Daten imputation
AmeliaDatensatze<-AmeliaDatensatzeErzeugung(DatensatzNA,x="ords")

ClusterUDatensatz<-Clustern1(Datensatz)
# ClusterUDatensatz enthält Gruppenmatrix des ursprünglichen Datensatzes
GruppenmatrixDatensatz<-as.data.frame(ClusterUDatensatz[2])
##

# Möglichkeit 2: Analyse der imputierten AmeliaDatensätze 1 bis 5
# in Amelia5Cluster wird zunächst alles gespeichert:
AnalyseDatensatz2<-Analysefunktion(DatensatzNA)

# Speichern der AmeliaDatensätze und DatensatzNA
AmeliaDatensatze2<-AmeliaDatensatze
DatensatzNAD2<-DatensatzNA

save.image("~/Sicherung1.RData")
#############################################################################################################
#Datensatz 3 mit nominalen Merkmal und 30 % entfernten Daten
#############################################################################
### Datensatz ist der Datensatz, bei dem später fehlende Werte erzeugt werden.
### Er wird deshalb öfter als der "ursprüngliche" Datensatz bezeichnet.
Datensatz<-Datensatz3nom
EchteGruppe<-Datensatz[,4]

#set.seed für Reproduzierbarkeit
set.seed(123)

### Datensatz mit entfernen Werten ist DatensatzNA1
DatensatzNA<-(NAErzeugung(Datensatz,AnzahlentfernterWerte))

### Amelia Daten imputation
AmeliaDatensatze<-AmeliaDatensatzeErzeugung(DatensatzNA,x="noms")

##
ClusterUDatensatz<-Clustern1(Datensatz)
# ClusterUDatensatz enthält Gruppenmatrix des ursprünglichen Datensatzes
GruppenmatrixDatensatz<-as.data.frame(ClusterUDatensatz[2])
##
# Analyse der imputierten AmeliaDatensätze 1 bis 5
# in Amelia5Cluster wird zunächst alles gespeichert:
AnalyseDatensatz3<-Analysefunktion(DatensatzNA)

# Speichern der AmeliaDatensätze und DatensatzNA
AmeliaDatensatze3<-AmeliaDatensatze
DatensatzNAD3<-DatensatzNA

#############################################################################
#Datensatz 4 mit ordinalen Merkmalen und 30 % entfernten Daten
#############################################################################
### Datensatz ist der Datensatz, bei dem später fehlende Werte erzeugt werden.
### Er wird deshalb öfter als der "ursprüngliche" Datensatz bezeichnet.
Datensatz<-Datensatz4ord
EchteGruppe<-Datensatz[,4]

#set.seed für Reproduzierbarkeit
set.seed(123)

### Datensatz mit entfernen Werten ist DatensatzNA1
DatensatzNA<-(NAErzeugung2(Datensatz,AnzahlentfernterWerte))

### Amelia Daten imputation
AmeliaDatensatze<-AmeliaDatensatzeErzeugung2(DatensatzNA,x="ords")

##
ClusterUDatensatz<-Clustern1(Datensatz)
# ClusterUDatensatz enthält Gruppenmatrix des ursprünglichen Datensatzes
GruppenmatrixDatensatz<-as.data.frame(ClusterUDatensatz[2])
##

# Möglichkeit 2: Analyse der imputierten AmeliaDatensätze 1 bis 5
AnalyseDatensatz4<-Analysefunktion(DatensatzNA)

# Speichern der AmeliaDatensätze und DatensatzNA
AmeliaDatensatze4<-AmeliaDatensatze
DatensatzNAD4<-DatensatzNA
#############################################################################################################
#Datensatz 5 mit nominalen Merkmalen und 30 % entfernten Daten
#############################################################################
### Datensatz ist der Datensatz, bei dem später fehlende Werte erzeugt werden.
```

```
### Er wird deshalb öfter als der "ursprüngliche" Datensatz bezeichnet.
Datensatz<-Datensatz5nom
EchteGruppe<-Datensatz[,4]

#set.seed für Reproduzierbarkeit
set.seed(123)

### Datensatz mit entfernen Werten ist DatensatzNA1
DatensatzNA<-(NAErzeugung2(Datensatz,AnzahlentfernterWerte))

### Amelia Daten imputation
AmeliaDatensatze<-AmeliaDatensatzeErzeugung2(DatensatzNA,x="noms")

##
ClusterUDatensatz<-Clustern3(Datensatz)
# ClusterUDatensatz enthält Gruppenmatrix des ursprünglichen Datensatzes
GruppenmatrixDatensatz<-as.data.frame(ClusterUDatensatz[2])
##
# Analyse der imputierten AmeliaDatensätze 1 bis 5
AnalyseDatensatz5<-Analysefunktion2(DatensatzNA)

# Speichern der AmeliaDatensätze und DatensatzNA
AmeliaDatensatze5<-AmeliaDatensatze
DatensatzNAD5<-DatensatzNA

save.image("~/Sicherung1.RData")
##########
# Entfernung überflüssiger Objekte aus Workspace
# All diese Objekte sind in anderen Objekten gespeichert und dienten nur zur Vereinfachung
rm(AmeliaDatensatze,DatensatzNA1,GruppenmatrixDatensatz,ClusterUDatensatz,Datensatz,
   EchteGruppe,AnzahlentfernterWerte)

#############################################################################################################

# Man erhält für jeden Datensatz eine Liste "AnalyseDatensatz". Da dies sehr komplex
# aufgebaut ist, hier eine kurze Erläuterung, wie relevante Objekte/Informationen aus
# der Liste extrahiert werden können:

# Beschreibung des Inhalts von AnalyseDatensatz

#Hinweis:Für [[Verfahren]] muss die jeweilige Nummer des Verfahrens angegeben werden

### Liste mit 3 Elementen
## Element 1 [[1]]
# Das Element enthält eine Liste mit 29 Elementen, in denen jeweils die
# Vergleichstabellen für die verschiedenen Verfahren enthalten sind.
# mit [[1]][[Verfahren]] kann also auf die Tabellen der verschiedenen Verfahren
# zugegriffen werden. Um die Einzelnen Tabellen 1 - 4 zu betrachten, muss also
# [[1]][[Verfahren]][[Tabelle]] eingegeben werden.
#Vergleichstabelle 1 und 2 -> 1,2
#Genauigkeitstabelle 3 -> 4
#Analysetabelle ->3


## Element 2 [[2]]
# Elentent 2 enthält die verschiedenen Clusterobjekte.
# AmeliaObjekte kann man mit [[2]][[1-5,je nach Datensatz]][[3]] anwählen.
# Das UrsprungsClusterobjekt [[2]][[6]][[3]] mit anwählen.
# Je nach Verfahren noch [[Verfahren]] dranhängen, um einzelnen Verfahren auszuwählen.

## Element 3 [[3]]
# Element 3 enthält die Gruppenmatrizen.
# In [[3]][[1-5]] sind die AmeliaGruppenmatrizen gespeichert.
# In [[3]][[6]] ist die ursprüngliche Gruppenmatrix gespeichert.
# Um auf ein bestimmtes Verfahren zuzugreifen, muss zusätzlich zur gewählten Gruppenmatrix
# [,Verfahren] angehängt werden.
```

B.4 Erzeugung der Tabellen und Plots

Für die Erzeugung der Übersichtstabellen, der Tabellen mit den Veränderung der Genauigkeiten, der Vergleichstabellen sowie für die Vergleichsplots wurden spezielle Funktionen erstellt. Bei den Tabellen werden diese zusätzlich noch in Latex Form ausgegeben. Diese werden hier kurz aufgeführt. Die Erzeugung der restlichen Tabellen und Plots ist im elektronischen Anhang zu finden.

```
#Teil 4 #Hilfsfunktionen
```

```
#Pakete

library(gridExtra)
library(xtable)

# Hilfsfunktion Uberblick
# Überblick über Übereinstimmung eines bestimmten Verfahrens über die 5 Datensätze
# Es muss eingegeben werden, welches Verfahren untersucht werden soll (als Zahl).

# Übereinstimmung imputiert
Uberblick<-function(Verfahren){
D<-cbind(AnalyseDatensatz1[[1]][[Verfahren]][[3]][,3],
         AnalyseDatensatz2[[1]][[Verfahren]][[3]][,3],
         AnalyseDatensatz3[[1]][[Verfahren]][[3]][,3],
         AnalyseDatensatz4[[1]][[Verfahren]][[3]][,3],
         AnalyseDatensatz5[[1]][[Verfahren]][[3]][,3])
colnames(D)<-c("D1 ",
               "D2",
               "D3 ",
               "D4 ",
               "D5 ")
rownames(D)<-c("imputiert","nicht imputiert")
print(D)
xtable(D)
}

##############################################################################################
# Hilfsfunktion Tabelle

# Diese Funktion ermöglicht es, auf die Tabellen der Verfahren zuzugreifen und erzeugt zudem
# den Latex-Code.

# Der Tabelle Funktion muss das AnalyseDatensatz Objekt und das zu betrachtete
# Verfahren übergeben werden.


Tabelle<-function(AnalyseDatensatz1,Verfahren){

  Tab<-AnalyseDatensatz1[[1]][[Verfahren]]

  print(Tab)

  # Tabelle 1
  print("##### Vergleichs Tabelle 1 ######")
  print(xtable(data.frame(row = rownames(Tab[[1]]),data.frame(Tab[[1]]))),include.rownames = FALSE)

  # Tabelle 2
  print("##### Vergleichs Tabelle 2 ######")
  print(xtable(data.frame(row = rownames(Tab[[2]]),data.frame(Tab[[2]]))),include.rownames = FALSE)

  # Tabelle 3
  print("##### Analysetabelle ######")
  print(xtable(data.frame(row = rownames(Tab[[3]]),data.frame(Tab[[3]]))),include.rownames = FALSE)

  #Tabelle 4

  print("##### Genauigkeitstabelle  ######")
  print(xtable(data.frame(row = rownames(Tab[[4]]),data.frame(Tab[[4]]))),include.rownames = FALSE)

}
Die entstehende Warnung kann ignoriert werden.
##############################################################################################################

# Hilfsfunktion Dendo

# Mithilfe dieser Funktion werden die Dendogramme
# für die hierarchischen Verfahren erstellt.
# Es muss das AnalyseDatensatzObjekt sowie das zu
# untersuchende Verfahren übergeben werden.

Dendo<-function(AnalyseDatensatz1,Verfahren){


  # Man erhält die einzelnen Clusterobjekte für die 5 AmeliaDatensätze
  # sowie das des Ursprungsdatensatzes mit:
  COU<- AnalyseDatensatz1[[2]][[6]][[3]][[Verfahren]]
  COA1<-AnalyseDatensatz1[[2]][[1]][[3]][[Verfahren]]
  COA2<-AnalyseDatensatz1[[2]][[2]][[3]][[Verfahren]]
  COA3<-AnalyseDatensatz1[[2]][[3]][[3]][[Verfahren]]
  COA4<-AnalyseDatensatz1[[2]][[4]][[3]][[Verfahren]]
  COA5<-AnalyseDatensatz1[[2]][[5]][[3]][[Verfahren]]

  #Zugehörige Genauigkeiten
```

```
  GCOU<- AnalyseDatensatz1[[2]][[6]][[1]][[Verfahren]]
  GCOA1<-AnalyseDatensatz1[[2]][[1]][[1]][[Verfahren]]
  GCOA2<-AnalyseDatensatz1[[2]][[2]][[1]][[Verfahren]]
  GCOA3<-AnalyseDatensatz1[[2]][[3]][[1]][[Verfahren]]
  GCOA4<-AnalyseDatensatz1[[2]][[4]][[1]][[Verfahren]]
  GCOA5<-AnalyseDatensatz1[[2]][[5]][[1]][[Verfahren]]
  #Für die hierarchischen Verfahren werden die Dendogramme verglichen:

  par(mfrow=c(3,3))
  plot(COU, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="Datensatz", hang=-1, sub=as.name(GCOU))
  rect.hclust(COU, 3 )
  plot(COA1, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz1", hang=-1,sub=as.name(GCOA1))
  rect.hclust(COA1, 3 )
  plot(COA2, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz2", hang=-1,sub=as.name(GCOA2))
  rect.hclust(COA2, 3 )
  plot(COA3, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz3", hang=-1,sub=as.name(GCOA3))
  rect.hclust(COA3, 3 )
  plot(COA4, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz4", hang=-1,sub=as.name(GCOA4))
  rect.hclust(COA4, 3 )
  plot(COA5, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz5", hang=-1,sub=as.name(GCOA5))
  rect.hclust(COA5, 3 )
}

#########################################################################

#############################################################################################################

# Hilfsfunktion Dendo 5
# Dendo für Datensatz 5
# Mithilfe dieser Funktion werden die Dendogramme
# für die hierarchischen Verfahren erstellt.
# Es muss das AnalyseDatensatzObjekt sowie das zu
# untersuchende Verfahren übergeben werden.

Dendo5<-function(AnalyseDatensatz1,Verfahren){


  # Man erhält die einzelnen Clusterobjekte für die 5 AmeliaDatensätze
  # sowie das des Ursprungsdatensatzes mit:
  COU<- AnalyseDatensatz1[[2]][[6]][[3]][[Verfahren]]
  COA1<-AnalyseDatensatz1[[2]][[1]][[3]][[Verfahren]]
  COA2<-AnalyseDatensatz1[[2]][[2]][[3]][[Verfahren]]
  COA3<-AnalyseDatensatz1[[2]][[3]][[3]][[Verfahren]]
  COA4<-AnalyseDatensatz1[[2]][[4]][[3]][[Verfahren]]
  COA5<-AnalyseDatensatz1[[2]][[5]][[3]][[Verfahren]]

  #Zugehörige Genauigkeiten
  GCOU<- AnalyseDatensatz1[[2]][[6]][[1]][[Verfahren]]
  GCOA1<-AnalyseDatensatz1[[2]][[1]][[1]][[Verfahren]]
  GCOA2<-AnalyseDatensatz1[[2]][[2]][[1]][[Verfahren]]
  GCOA3<-AnalyseDatensatz1[[2]][[3]][[1]][[Verfahren]]
  GCOA4<-AnalyseDatensatz1[[2]][[4]][[1]][[Verfahren]]
  GCOA5<-AnalyseDatensatz1[[2]][[5]][[1]][[Verfahren]]
  #Für die hierarchischen Verfahren werden die Dendogramme verglichen:

  par(mfrow=c(3,3))
  plot(COU, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="Datensatz", hang=-1, sub=as.name(GCOU))
  rect.hclust(COU, 4 )
  plot(COA1, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz1", hang=-1,sub=as.name(GCOA1))
  rect.hclust(COA1, 4 )
  plot(COA2, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz2", hang=-1,sub=as.name(GCOA2))
  rect.hclust(COA2, 4 )
  plot(COA3, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz3", hang=-1,sub=as.name(GCOA3))
  rect.hclust(COA3, 4 )
  plot(COA4, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz4", hang=-1,sub=as.name(GCOA4))
  rect.hclust(COA4, 4 )
  plot(COA5, labels=FALSE, xlab="Genauigkeit",ylab="",
       main="AmeliaDatensatz5", hang=-1,sub=as.name(GCOA5))
  rect.hclust(COA5, 4 )
}

#########################################################################
```

```
#Hilfsfunktion: Vergleich Plots vorher und nach Imputation


Plots<-function(AnalyseDatensatz,Verfahren,AmeliaDatensatze,Datensatz1ord){
  if(Verfahren <=22){
  GruppeU<-cbind(Datensatz1ord[,1:2],
                 (AnalyseDatensatz[[3]][[6]][,Verfahren]))
  Gruppe1<-cbind(AmeliaDatensatze[[1]][[1]],
                 (AnalyseDatensatz[[3]][[1]][,Verfahren]))
  Gruppe2<-cbind(AmeliaDatensatze[[1]][[2]],
                 (AnalyseDatensatz[[3]][[2]][,Verfahren]))
  Gruppe3<-cbind(AmeliaDatensatze[[1]][[3]],
                 (AnalyseDatensatz[[3]][[3]][,Verfahren]))
  Gruppe4<-cbind(AmeliaDatensatze[[1]][[4]],
                 as.factor(AnalyseDatensatz[[3]][[4]][,Verfahren]))
  Gruppe5<-cbind(AmeliaDatensatze[[1]][[5]],
                 (AnalyseDatensatz[[3]][[5]][,Verfahren]))
  #Plots
  par(mfrow=c(3,3))

  p1<-xyplot(Gruppe1$Variable1~Gruppe1$Variable2,
             groups=Gruppe1[,3],
             xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 1")
  p2<-xyplot(Gruppe2$Variable1~Gruppe2$Variable2,
             groups=Gruppe2[,3],
             xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 2")
  p3<-xyplot(Gruppe3$Variable1~Gruppe3$Variable2,
             groups=Gruppe3[,3],
             xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 3")
  p4<-xyplot(Gruppe4$Variable1~Gruppe4$Variable2,
             groups=Gruppe4[,3],
             xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 4")
  p5<-xyplot(Gruppe5$Variable1~Gruppe5$Variable2,
             groups=Gruppe5[,3],
             xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 5")
  p6<-xyplot(GruppeU$Variable1~GruppeU$Variable2,
             groups=GruppeU[,3],
             xlab="Variable2",ylab="Variable1",main="Datensatz")
  grid.arrange(p6,p1,p2,p3,p4,p5, ncol=3)
  }

  if(Verfahren >22){
    GruppeU<-cbind(Datensatz1ord[,1:3],
                   (AnalyseDatensatz[[3]][[6]][,Verfahren]))
  Gruppe1<-cbind(AmeliaDatensatze[[2]][[1]],
                 (AnalyseDatensatz[[3]][[1]][,Verfahren]))
  Gruppe2<-cbind(AmeliaDatensatze[[2]][[2]],
                 (AnalyseDatensatz[[3]][[2]][,Verfahren]))
  Gruppe3<-cbind(AmeliaDatensatze[[2]][[3]],
                 (AnalyseDatensatz[[3]][[3]][,Verfahren]))
  Gruppe4<-cbind(AmeliaDatensatze[[2]][[4]],
                 as.factor(AnalyseDatensatz[[3]][[4]][,Verfahren]))
  Gruppe5<-cbind(AmeliaDatensatze[[2]][[5]],
                 (AnalyseDatensatz[[3]][[5]][,Verfahren]))

    #Plots
    par(mfrow=c(3,3))

    p1<-xyplot(Gruppe1$Variable1~Gruppe1$Variable2,
               groups=Gruppe1[,4],
               xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 1")
    p2<-xyplot(Gruppe2$Variable1~Gruppe2$Variable2,
               groups=Gruppe2[,4],
               xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 2")
    p3<-xyplot(Gruppe3$Variable1~Gruppe3$Variable2,
               groups=Gruppe3[,4],
               xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 3")
    p4<-xyplot(Gruppe4$Variable1~Gruppe4$Variable2,
               groups=Gruppe4[,4],
               xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 4")
    p5<-xyplot(Gruppe5$Variable1~Gruppe5$Variable2,
               groups=Gruppe5[,4],
               xlab="Variable2",ylab="Variable1",main="Amelia Datensatz 5")
    p6<-xyplot(GruppeU$Variable1~GruppeU$Variable2,
               groups=GruppeU[,4],
               xlab="Variable2",ylab="Variable1",main="Datensatz")
    grid.arrange(p6,p1,p2,p3,p4,p5, ncol=3)
  }

}
################################################################
#Hilfsfunktion
#Genauigkeiten
#Genauigkeit vor und nach Imputation
```

```
Genauigkeit<-function(Verfahren){
  G<-cbind(AnalyseDatensatz1[[2]][[6]][[1]][[Verfahren]],
           AnalyseDatensatz2[[2]][[6]][[1]][[Verfahren]],
           AnalyseDatensatz3[[2]][[6]][[1]][[Verfahren]],
           AnalyseDatensatz4[[2]][[6]][[1]][[Verfahren]],
           AnalyseDatensatz5[[2]][[6]][[1]][[Verfahren]])


  GNI<-cbind(AnalyseDatensatz1[[1]][[Verfahren]][[4]][8],
             AnalyseDatensatz2[[1]][[Verfahren]][[4]][8],
             AnalyseDatensatz3[[1]][[Verfahren]][[4]][8],
             AnalyseDatensatz4[[1]][[Verfahren]][[4]][8],
             AnalyseDatensatz5[[1]][[Verfahren]][[4]][8])

  G<-rbind(G,GNI)}
  colnames(G)<-c("D1","D2","D3","D4","D5")
  rownames(G)<-c("Genauigkeit vor Imputation","Genauigkeit nach Imputation")
  print(G)
  xtable(G)
}
```

C CD Inhalt

Die beiliegende CD enthält die digitale Ausgabe dieser Arbeit und das zugehörige Literaturverzeichnis. Zudem ist sämtlicher relevanter R-Code sowie ein Workspace, der alle wichtigen Objekte beinhaltet, enthalten.

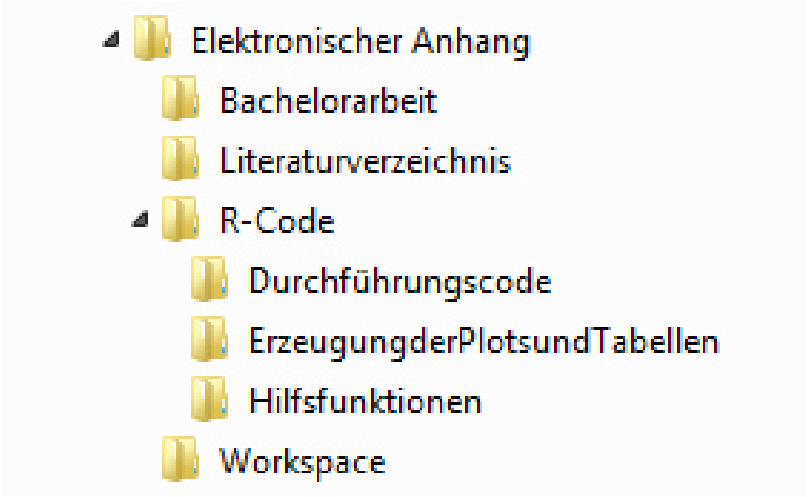

Abbildung 26: Ordnungstrukur der CD

D Literaturverzeichnis

[Alexandrowich 2011] Grigory Alexandrovich:

Analytische Eigenschaften von Mischungen elliptischer Verteilungen und deren Anwendung in der Clusteranalyse

URL*: https://www.mathematik.uni-marburg.de/~alexandrovich/diplomarbeit.pdf*

Universitaet Marburg, 2011

[Deng 2012] Deng:

Missingness Mechanism (MCAR, MAR, and MNAR) - Great Explanation of These Terms

URL: http://onbiostatistics.blogspot.de/2012/10/missingness-mechanism-mcar-mar-and-mnar.html

[Engel et al. 2008] Joachim Engel und Rudolf Gruebel:

Bootstrap — oder die Kunst, sich selbst aus dem Sumpf zu ziehen

Url: http://www.stochastik.uni-hannover.de/fileadmin/institut/pdf/MathSemBer2008preprint.pdf

Universitaet Hannover, 2008

[Fairmeir et al. 1996] Fairmeir L.;Brachinger W.; Hamerle A.; Tutz G.:

Multivariate statistische Verfahren

Verlag: De Gruyter, 1996

[Fairmeir et al. 2010] Ludwig Fairmeir und Christian Heumann:

Vorlesungsskript Schätzen und Testen I, Wintersemester 2009/2010

URL: *http://www.statistik.lmu.de/institut/lehrstuhl/semwiso/schaetzentesten1-ws0910/skript/ST1-ws0910-kap05.pdf*

Ludwig Maximilian Universitaet Muenchen, 2010

[Fairmeir et al. 2011] Fairmeir L.; Kuenstler R.; Pigeot I.;Tutz G.:

Statistik - Der Weg zur Datenanalyse

Springer Verlag, 2011

[Feilke et al. 2009] Martina Feilke:
SOLAR II - Komplexe Modellierung von beruflichen Allergierisiken und Simulation zur Anwendung von Imputationsmethoden
URL: http://epub.ub.uni-muenchen.de/11250/1/BA_Feilke.pdf
Ludwig Maximilian Universitaet Muenchen, 2009

[Fraley et al. 2007] Chris Fraley und Adrian Raftery:
Model-based Methods of Classification:Using the mclust Software in Chemometrics
URL*: http://www.jstatsoft.org/v18/i06/paper*
Journal of Statstical Software, 2007

[Fraley et al. 2012] Fraley C.; Raftery A., Murphy B, Scrucca L.:
mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation
URL: http://www.stat.washington.edu/research/reports/2012/tr597.pdf
Universitaet Washington, 2012

[Greutert 2004] Andreas Greutert:
Methoden zur Schätzung der Clusteranzahl
URL: ftp://ftp.stat.math.ethz.ch/Masters-Theses/Andreas_Greutert-Clusteranz-Schaetzung.pdf
Swiss Federal Institute of Technolgy Zurich, 2004

[Handl 2002] Andreas Handl:
Multivariate Analysemethoden
Springer Verlag, 2002

[Hortensius 2012] Lian Hortensius*:*

Drichtlet Distribution

URL: http://www.tc.umn.edu/~horte005/docs/

Dirichletdistribution.pdf

Universitaet Minnesota, 2012

[Hueftle 2006] Mike Hueftle:

Methoden zur Segmentierung von Daten: Cluster und

Selbstorganisierende Karten

URL: http://www.optiv.de/Methoden/ClustMet/index.htm?12

OptiV, 2006

[Honaker et al. 2013] Honaker J.; King.G.; Blackwell M.:

AMELIA II : A Program for Missing Data

URL*: http://cran.r-project.org/web/packages/Amelia/vignettes*

/amelia.pdf#page=1&zoom=auto,0,690

Cran R-project, 2013

[Igl o.D] Wilmar Igl:

Behandlung fehlender Werte

URL: http://www.rehawissenschaft.uni-wuerzburg.de/

methodenberatung/Igl_040604_Halle_Fehlende_Werte.pdf

Universitaet Wuerzburg, o. D

[Kahn et al. 2001] Thomas Kahn und Olaf Bruel:

Analyse der Standortqualitaet zur Beurteilung der wirtschaftlichen

Leistungsfaehigkeit im interregionalen Vergleich

GRIN Verlag, *2001*

[Kalisch 2012] Markus Kalisch:

Mesuring Distances

URL: `http://stat.ethz.ch/education/semesters/ss2012/ams/slides/v4.2.pdf`

Swiss Federal Institute of Technolgy Zurich, 2012

[Landscape Ecology Lab(1) o.D] Landscape Ecology Lab:

Cluster Analysis

URL*:* `http://www.umass.edu/landeco/teaching/multivariate/schedule/cluster1.pdf`

Universitaet Massachusetts Amherst, o. D

[Landscape Ecology Lab(2) o.D] Landscape Ecology Lab:

Polythetic Divisive Hierarchical Clustering

URL*:* `http://www.umass.edu/landeco/teaching/multivariate/schedule/cluster2.pdf`

Universitaet Massachusetts Amherst, o. D

[Lintorf 2011] K. Lintorf:

Wie Vorhersagbar sind Grundschulnoten? Prädikatikraft individueller und kontextspezifischer Merkmale

VS Verlag für Sozialwissenschaften GmbH, 2011

[Masayoshi et al. 2012] Takahashi Masayoshi und Ito Takayuki:

Multiple Imputation off Turnover in Edinet Data:

Toward the Improvment of Imputation for the economic Census

URL*:* `http://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.44/2012/35_Japan.pdf`

National Statistics Center Japan, 2012

[Mayer 2010] Benjamin Mayer:

Fehlende Werte in klinischen Verlaufstudien – Der Umgang mit Studienabbrechern

URL: http://vts.uni-ulm.de/docs/2011/7633/vts_7633_10939.pdf

Universitaet Ulm, 2010

[Minka 2012] Thomas Minka:

Estimating a Dirichlet distribution

URL: http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet/minka-dirichlet.pdf

Microsoft, 2012

[Mirkes 2011] E.M. Mirkes:

K-means and K-medoids applet

URL: http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html

Universitaet Leicester, 2011

[Oellinger 2011] Birgit Oellinger:

Funktionales Clustern von Transaktionsverlauefen

URL: http://epub.ub.uni-muenchen.de/11716/1/MA_Oellinger.pdf

Ludwig Maximilan Universitaet Muenchen, 2011

[Rahnenfuehrer 2008] Joerg Rahnenfuehrer:

Die multivariate Normalverteilung

URL: http://www.statistik.tu dortmund.de/fileadmin/user_upload/Lehrstuehle/Genetik/MV0809/Vorlesung20081020.pdf

Technische Universitaet Dortmund, 2008

[R Dokumentation(1) o.D] R Dokumentation:
K-means Clustering
URL: http://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html
Swiss Federal Institute of Zurich, o. D

[R Dokumentation(2) o.D] R Dokumentation:
Partitioning arround Medoids
URL: http://stat.ethz.ch/R-manual/R-devel/library/cluster/html/pam.html
Swiss Federal Institute of Zurich, o.D

[R Dokumentation(3) o.D] R Dokumentation:
DIvisive ANAlysis Clustering
URL: http://stat.ethz.ch/R-manual/R-patched/library/cluster/html/diana.html
Swiss Federal Institute of Zurich, o.D

[Rohrschneider 2007] Lars Rohrschneider:
Behandlung fehlender Daten
URL: http://edoc.hu-berlin.de/master/rohrschneider-lars-2007-07-23/PDF/rohrschneider.pdf
Humboldt Universitaet Berlin, 2007

[Rohwer et al. 2011] Rohwer und Dudel:
Methoden der Datenrepräsentation
und Klassifikation Kapitel 6: Hierarchische Klassifikation
URL: http://www.stat.ruhr-uni-bochum.de/teaching-archiv/teaching-wise2010_11/teaching/drk/drk6.pdf
Universitaet Bochum, 2011

[Schaefer o.D] Josep Schafer:

The multiple imputation FAQ page

URL: http://sites.stat.psu.edu/~jls/mifaq.html#minf

Everly College of Science, o.D

[Spiess 2008] Martin Spiess:

Missing-data-Techniken: Analyse von Daten mit fehlenden Werten

Lit, 2008

[Stein et al. 2011] Petra Stein und Sven Vollnhals:

Grundlagen clusteranalytischer Verfahren

URL: https://www.uni-due.de/imperia/md/content/soziologie/stein/skript_clusteranalyse_sose2011.pdf

Universitaet Duisburg-Essen, 2011

[Toutenburg et al. 2002] Helge Toutenburg und Christian Heumann:

Lineare Modelle

Springer Verlag, *2002*

[Unbekannt(1) o.D] Unbekannt:

Divisive Analysis(Diana)

URL: http://www.unesco.org/webworld/idams/advguide/Chapt7_1_5.htm

Unesco, o.D

[Unbekannt(2) o.D] Unbekannt:

Partitioning Around Medoids (Pam)

URL: http://www.unesco.org/webworld/idams/advguide/Chapt7_1_1.htm

Unesco, o.D

[Unbekannt(3) o.D] Unbekannt:

Distance

URL*: http://chessprogramming.wikispaces.com/Distance#Chebyshev %20Distance*

Chessprogramming, o.D

[Unbekannt 2007] Unbekannt:

Beta distribution

URL: http://www.vosesoftware.com/ModelRiskHelp/index.htm# Distributions/Continuous_distributions/Beta_distribution.htm

Vose Software, 2007

[Unbekannt 2010] Unbekannt:

Clusteranalyse

Vorlesung zur Einführung in die multivariate Statistik

Universitaet Goettingen, 2010

Anmerkung: Dokument ist im elektronischer Anhang

[Wiedenbeck et al. o.D] Michaeal Wiedenbeck und Cornelia Zuell:

Klassifikation mit Clusteranalyse: Grundlegende Techniken hierarchische und K-means Verfahren

URL: http://www.gesis.org/fileadmin/upload/forschung/ publikationen/gesis_reihen/howto/how-to10mwcz.pdf

Zentrum für Umfragen, Methoden und Analysen, Mannheim, o.D