Universität Zürich[UZH]

Bachelorarbeit

Implementierung eines Tools zur statistischen Auswertung von Simulationsdaten

Bachelorarbeit im Studiengang Wirtschaftsinformatik

vorgelegt von

Sven Brunner
Gränichen, Aargau, Schweiz
Matrikelnummer 10-933-703

Angefertigt am

Institut für Informatik
der Universität Zürich

Betreuer:
Stefan Holm

Betreuender Professor:
Prof. Lorenz Hilty

Abgabe: 16. Juli 2014

Abstract

Zusammenfassung

Simulationen spielen in der heutigen Forschung eine immer zentralere Rolle. Die von den Simulationen generierten Daten, welche meist in grosser Menge anfallen, können Wissen und neue Erkenntnisse liefern. Die vorliegende Bachelorarbeit beschäftigt sich mit der Umsetzung einer speziell auf einen Anwendungsfall, der Simulation des Modells des Schweizer Energieholzmarktes, zugeschnittenen Analysesoftware. Ziel war es, dem Benutzer ein Werkzeug an die Hand zu geben, um Wissen und verborgene Muster aus den Daten zu extrahieren. Verwendet werden dazu vor allem Verfahren der multivariaten Statistik, mit besonderem Fokus auf Clusteranalysen. Um den Benutzer von der Verantwortung der Auswahl und Konfigurierung der komplexen Clusterverfahren zu entbinden, wurden Grundeinstellungen erarbeitet, die eine optimale Analyse ermöglichen. Zudem wurde ein Editor für die Festlegung logischer Regeln entwickelt und integriert, mit dem inhaltlich interessante Phänomene in den Daten entdeckt werden können. Die Arbeit beinhaltet das Programm selbst, den Beschrieb dessen Entwicklung und Funktionsweise, eine theoretische Fundierung der verwendeten statistischen Verfahren sowie eine Einführung in die Clusteranalyse.

Abstract

Simulations play an ever more central role in today's research. The massive amount of data generated by these simulations is likely to contain new knowledge. The present bachelor thesis engages in implementing a software tool for analyzing the data of a very specific case, the data resulting from the simulation of the model of the Swiss wood fuel market. The goal was to provide the user with a powerful tool in order to extract knowledge and hidden patterns from the data. Methods from multivariate statistics, with a special focus on cluster analysis, are used. Default settings has been developed to release the user from the responsibility of choosing and configuring the mostly complex cluster algorithms. Additionally, an editor for creating logical rules was developed and integrated into the program. With the help of this editor substantial phenomena (in terms of content) can be discovered. Along with the software tool itself and the description of its development and functionality, this thesis contains a theoretical foundation of the used statistical methods and an introduction to cluster analysis.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Ermöglicht durch die stetige Weiterentwicklung und Leistungssteigerung moderner Computerhardware, werden rechnergestützte Simulationen zu einem immer bedeutenderen Instrument in der heutigen Forschung. Typischerweise produzieren diese Simulationen eine grosse Menge an Datenmaterial. Um daraus jedoch neues Wissen zu generieren, bedarf es einer adäquaten Analyse der Daten. Dies ist genau der Punkt, an dem diese Arbeit anknüpft. Die Daten stammen aus der Arbeit „Design und Implementierung eines agentenbasierten Modells des Schweizer Energieholzmarktes“ von Stefan Holm. Im Rahmen dieser Bachelorarbeit soll ein Programm in Java entwickelt werden, welches eine umfassende statistische Analyse der Daten ermöglicht. Ein besonderes Augenmerk wird dabei auf die Thematik der Clusteranalyse gerichtet. Ziel ist es jedoch nicht, neue Verfahren zu entwickeln, sondern ein Tool zur Verfügung zu stellen, das einerseits stark auf den Anwendungsfall und die Daten zugeschnitten ist und andererseits die passenden Verfahren auswählt und soweit optimiert, dass sich der Benutzer um möglichst wenige Details kümmern muss. Zusätzlich soll ein Weg gefunden werden, um Phänomene in den Daten zu finden, denen eine inhaltliche Bedeutung beigemessen werden kann.

1.1 Ziel

Ziel dieser Arbeit ist die Implementierung eines Programms, das dem Forschenden erlauben soll, durch eine Simulation generierte Daten zu analysieren und darin enthaltene Strukturen und verborgenes Wissen aufzudecken. Die zur Verfügung gestellten Analysemethoden stammen aus der deskriptiven sowie multivariaten Statistik, wobei für diese Arbeit Clusteranalysen von besonderer Bedeutung sind. Dabei geht es nicht darum ein möglichst allgemeines Tool zu entwickeln, welches für jegliche Art von Daten und Anwendungsfälle verwendet werden kann, sondern ein ganz spezifisch auf die Simulation des Modells des Schweizer Holzmarktes angepasstes Analyseprogramm. Wo eine generelle Lösung Kompromisse eingehen muss, um ein breiteres Spektrum an Problemfällen abdecken zu können, kann sich eine Speziallösung vollumfänglich auf die Gegebenheiten eines spezifischen Falles einlassen. Dieser Prämisse folgt auch die vorliegende Arbeit. Die Analysen der zu entwickelnden Software sollen Resultate

liefern, die mit einem anderen Programm so nicht möglich oder viel schwieriger zu realisieren gewesen wären. Um die Analyse für den Benutzer möglichst einfach zu gestalten, wird eine Vorauswahl an besonders geeigneten statistischen Verfahren getroffen. Dies ist insbesondere bei den Clusteralgorithmen wichtig, da dort zahlreiche unterschiedliche Verfahren existieren, um deren Eigenschaften sich der Benutzer sonst kümmern müsste.
Ein Merkmal der Simulationsergebnisdaten ist die grosse Anzahl an unterschiedlichen Variablen (gegen 200). Zu deren Analyse soll das Programm einfach zugängliche und effiziente Methoden bereitstellen. Falls möglich, sollen die Resultate bei der Analyse (insbesondere bei der Clusteranalyse) sogleich gefiltert werden, um dem Benutzer möglichst nur die relevanten Fälle zu präsentieren. Diese sollen neben statistischen Kennzahlen auch weitere interessante Phänomene in den Daten aufdecken, die mit inhaltlicher Bedeutung versehen werden können (beispielsweise Auftreten von Unterversorgung etc. . .).
Alle Resultate und Auswertungen sollen ausserdem adäquat visualisiert werden. Es sollen also je nach Analyseverfahren passende Grafiken erstellt werden.

1.2 Einordnung der Arbeit

Diese Arbeit knüpft insbesondere an die Masterarbeit von Stefan Holm mit dem Titel „Design und Implementierung eines agentenbasierten Modells des Schweizer Energieholzmarktes“ an (Holm, 2011). Die hier vorliegende Bachelorarbeit setzt sich jedoch nicht mit dem Modell des Schweizer Energieholzmarktes auseinander, sondern soll ein Tool bereitstellen, um die Ergebnisse der Simulation genauer analysieren zu können. Obwohl dazu auch eine Vorsortierung und Evaluierung von statistischen Methoden vorgenommen werden muss, ist diese Arbeit keine detaillierte Gegenüberstellung vieler verschiedener Analyseverfahren und deren Stärken und Schwächen. Die Verfahren werden nur in ihren Grundzügen erörtert und diejenigen Eigenschaften beleuchtet, die auch relevant für das zu entwickelnde Programm sind. Für die statistische Datenanalyse existiert eine Vielzahl an Computerprogrammen mit zum Teil sehr grossem Funktionsumfang. Beispiele sind das von IBM entwickelte „SPSS Statistics“, die von StatSoft hergestellte Software „Statistica“, „Matlab“ von The MathWorks oder die Statistikprogrammiersprache „R“. Das im Rahmen dieser Arbeit entstehende Programm soll sich vor allem durch eine einfachere Handhabung und vor allem spezifischere Ausrichtung von solchen allgemeinen Statistikprogrammen unterscheiden.

1.3 Verhältnis von schriftlicher Arbeit und Programm

Das Programm stellt sicherlich das Hauptprodukt dieser Arbeit dar. Der schriftliche Teil ist jedoch keineswegs lediglich eine Dokumentation des Entwicklungsprozesses. Er soll die in

der Software verwendeten statistischen Methoden ausreichend erörtern und deren Implementierung rechtfertigen. Neben der theoretischen Fundierung, dient der schriftliche Teil dieser Arbeit ausserdem dem Beschrieb der konkreten Implementierung und der Programmarchitektur. Zudem ist so eine Diskussion über die Eignung des Programms für den angedachten Zweck möglich.

1.4 Aufbau

In Kapitel 2 wird kurz auf die zu analysierenden Daten eingegangen und auch ein grober Überblick über das Modell und die Simulation gegeben. Kapitel 3 und 4 dienen der theoretischen Fundierung der statistischen Verfahren, ein Schwerpunkt wird dabei auf Clusteranalysen gelegt. In Kapitel 5 wird dann die Verbindung der Verfahren mit dem Programm hergestellt und erklärt, welche Verfahren warum implementiert wurden. Kapitel 6 und 7 widmen sich der Architektur und konkreten Implementierung der Software, deren Eignung in Kapitel 8 auch kritisch gewürdigt wird. Als Abschluss folgt in Abschnitt 9 ein kurzer Ausblick.

2. Modell und Daten

Eines der Hauptziele dieser Arbeit ist es, eine massgeschneiderte statistische Auswertung und Analyse für ein ganz bestimmtes Modell bzw. dessen Simulationsergebnisse zu liefern. Dazu soll in einem ersten Schritt kurz auf die Simulation im Allgemeinen und die generierten Daten im Speziellen eingegangen werden.

2.1 Modell und Simulation

Wie bereits angedeutet, sind die Daten die Simulationsergebnisse eines Modells des Schweizer Energieholzmarktes. Es handelt sich also um Holz, das aufgrund qualitativer Gesichtspunkte nicht zu Möbeln oder Baumaterial weiterverarbeitet werden kann. Zusätzlich wird aber auch der Markt mit qualitativ hochwertigem Holz, sogenanntes Rundholz, miteinbezogen. Da das Energieholz meist als Nebenprodukt bei der Gewinnung von Rundholz anfällt, besteht zwischen den beiden Holzarten eine mengenmässige Abhängigkeit. Bestimmt wird der Markt durch externe Ereignisse und Grössen (Ölpreis) und vor allem durch die Akteure, die direkt am Markt teilnehmen. Die Akteure, unterteilt in 11 unterschiedliche Rollen, treten auf den Märkten als Verkäufer und/oder Käufer auf und handeln nach vordefiniertem Entscheidungsverhalten. Die Simulation umfasst 1000 Akteure und erstreckt sich über einen Zeitraum von 20 Jahren. Unterschiede in den Resultaten der einzelnen Durchläufe ergeben sich einerseits durch veränderbare Startbedingungen, andererseits sind sie dadurch bedingt, dass manche Ereignisse und Entscheidungen von Akteuren vom Zufall abhängen. Während eines Simulationsdurchlaufs werden die Informationen in einer Ontologie-Datenbank[1] gehalten. Nach jedem simulierten Monat werden die Werte aller verwendeten Variablen (z.B. Preise für Holzarten, Menge an gekauftem- bzw. verkauftem Holz nach Rolle und Holzart etc...) in eine CSV („comma separated value“)-Datei exportiert. Hier besteht auch der Anknüpfungspunkt zu dieser Arbeit, denn die in den CSV-Dateien enthaltenen Daten sind das Analysematerial für das neu zu entwickelnde Programm. (Holm, 2011)

[1] Eine Ontologie in der Informatik dient dazu, Begriffe, deren formale Beschreibung und die Relationen zwischen den Begriffen so zu speichern, dass ein Computer sie verstehen kann.

2.2 Datenmaterial

Jede CSV-Datei stellt genau einen Simulationsdurchgang von 20 Jahren dar. Die aufgezeichneten Variablen werden als Spalten abgetragen, die erste Spalte ist für das Datum reserviert. Insgesamt sind es 200 Variablen, wobei es jedoch zu beachten gilt, dass manche nicht oder nicht mehr verwendet werden und die zur Variable dazugehörige Spalte leer sein kann (bzw. mit dem „null"-Schlüsselwort versehen ist). Meist bewegt sich die Anzahl der nichtleeren Variablen im Bereich von 175. Nun werden nach jedem simulierten Monat die Werte aller verwendeten Variablen als neue Zeile hinzugefügt. Somit besteht eine typische Datendatei aus 201 Spalten (200 Variablen plus das Datum) und 240 Zeilen (20 Jahre x 12 Monate). Neben den gar nicht ausgewerteten Variablen, kann es auch vorkommen, dass lediglich gewisse Werte fehlen, also in manchen Zeilen „null"-Werte stehen. Dieser Umstand wird dann insbesondere bei der Implementierung von grosser Bedeutung sein. Weiter gilt es festzuhalten, dass die Daten einer zeitlichen Ordnung unterliegen und die Reihenfolge der Zeilen somit eine Rolle spielt. Es ist zu erwarten, dass für einen Analysedurchgang gegen 100 CSV-Dateien miteinbezogen werden, womit auch klar wird, dass die verwendeten Analysemethoden eine nicht allzu hohe Zeitkomplexität aufweisen dürfen.

Die inhaltliche Bedeutung der Variablen ist für die Analyse nicht weiter von Belang, sehr wohl jedoch die Form. So macht es für die meisten statistischen Methoden einen grossen Unterschied, ob eine Variable numerische Werte besitzt oder auch sprachliche (z.B. könnte eine Variable „Qualität" in „gut", „mittel" und „schlecht" eingeteilt werden). Deshalb werden im nächsten Kapitel, neben der Beschreibung wichtiger statistischer Methoden, die unterschiedlichen Typen statistischer Variablen vorgestellt und auch erklärt, welchen Typen die zu untersuchenden Daten angehören und was es bei diesen zu beachten gilt.

Tabelle 2.1: Schema CSV-Datei

Datum	**Variable1**	**Variable2**	**Variable3**	...
01.01.2005	Wert	Wert	Wert	...
01.02.2005	Wert	Wert	Wert	...
...	...	...	...	...

3. Statistische Verfahren

Bis anhin wurden die Ergebnisse der Simulation lediglich gemittelt. Es wurde also für jede Variable der arithmetische Mittelwert gebildet. Für eine gründliche Analyse und für die Entdeckung neuen Wissens oder Muster in den Daten, reicht dies natürlich nicht aus. Um genau das zu erreichen, wurden in der erstellen Software eine Vielzahl von statistischen Verfahren verwendet und integriert. Ergebnis dieser Analysen sind einerseits statistische Kennzahlen, wie z.B. die Standardabweichung und auf der anderen Seite das Vorfinden von Mustern oder Auffälligkeiten in den Daten. Letzteres wird vorwiegend durch sogannte Clusteranalysen erreicht, welche Gegenstand des nächsten Abschnittes 4 sind. Zunächst werden in diesem Unterkapitel einige statistische Grundlagen und die Berechnungen der erwähnten Kennzahlen erörtert. Dabei wird keine umfassende Einführung in das Arbeiten mit Daten oder die Disziplin der Statistik geliefert; es soll lediglich eine kurze Übersicht über die für diese Arbeit relevanten theoretischen Begriffe und Verfahren gegeben werden.

3.1 Variablen und Objekte

Eine Variable in der Statistik ist ein bestimmtes Merkmal eines Objekts (Moore, McCabe, Alwan, Craig & Duckworth, 2011). Andernorts wird das Objekt auch als Untersuchungseinheit oder Erhebungseinheit bezeichnet (Schlittgen, 2000). Ein Objekt kann beispielsweise ein Kunde oder ein Einwohner sein, ist jedoch keineswegs auf Personen beschränkt. So sind auch Tierarten, Aggregate (z.B. Unternehmen oder Nationen) oder Gesteine denkbar, je nachdem was untersucht und analysiert werden soll. Ein Objekt wird nicht einfach als solches untersucht, sondern anhand von gewissen Merkmalen oder eben Variablen. Dabei kann eine Variable eine gewisse Ausprägung (konkreter Wert) annehmen. So kann das Merkmal “Geschlecht“ ,die Ausprägung “männlich“ oder “weiblich“ annehmen, das Merkmal “Einkommen“ jedoch eine Zahl. An diesem kleinen Beispiel erkennt man bereits, dass Variablen sehr unterschiedliche Wertebereiche besitzen können, die offensichtlich nicht einfach ohne weiteres vergleichbar sind. (Moore et al., 2011)

3.2 Kategorisierung der Variablen

Aufgrund der Unterschiede ihrer Werteausprägungen sowie der Art und Weise der Messung, werden statistische Variablen in verschiedene Kategorien eingeteilt. Grundsätzlich kann zwischen qualitativen, auch kategorial genannt, und quantitativen Variablen unterschieden werden. Diese können wiederum anhand des Skalenniveaus untergliedert werden (Jann, 2005):

- Qualitative Variablen
  - Nominal bzw. nominalskaliert: Die Ausprägungen sind zwar eindeutig zu unterscheiden, ihnen liegt jedoch keine natürliche Rangfolge zu Grunde. Zudem können keinerlei arithmetische Operationen durchgeführt werden. Als klassisches Beispiel wird oftmals das Geschlecht herangezogen. So können die Ausprägungen "männlich" und "weiblich" zwar klar unterschieden werden, es ist allerdings nicht möglich, einem der beiden einen höheren Rang zuzuordnen. Es gilt also weder männlich>weiblich, weiblich>männlich noch männlich = weiblich.
  - Ordinal bzw. ordinalskaliert: Ordinale Variablen besitzen nun eine eindeutige, natürliche Rangfolge, jedoch sind auch mit ihnen Rechenoperationen nicht möglich. So ist beispielsweise ein Bewertungssystem mit den Ausprägungen "schlecht", "durchschnittlich" und "gut" ordinalskaliert. Es kann ohne weiteres eine Rangfolge bestimmt werden: gut>durchschnittlich>schlecht. Eine Aussage über den Abstand der Ausprägungen ist aber nicht möglich (z.B. gut ist zwei Mal grösser als durchschnittlich)

- Quantitative Variablen
  - Metrisch bzw. intervallskaliert und verhältnisskaliert: Zusätzlich zur Rangfolge besitzt auch der Abstand zweier Werte eine Bedeutung. Dieser Abstand kann numerisch spezifiziert werden und es ist möglich zu sagen, um wie viel eine Ausprägung von einer anderen abweicht. Beträgt der Verdienst einer Person beispielsweise 5000 Fr. und erhält eine zweite Person ein Einkommen von 10000 Fr., so ist das zweite Gehalt genau doppelt so hoch wie das erste. Während die ersten beiden vorgestellten Arten von Variablen relativ starken Einschränkungen ausgesetzt sind, kommen viele davon bei den metrischen Variablen nicht zum Tragen.

3.3 Variablenbegriff in dieser Arbeit

In dieser Arbeit wird der Begriff Variable mit unterschiedlichen Bedeutungen benutzt. Wie gerade beschrieben als Variable im statistischen Sinn, die auch immer, wenn nicht aus dem

Kontext klar ersichtlich, als statistische Variable bezeichnet wird. Bei der Beschreibung der Resultate der Simulation in Kapitel 2, wird ebenfalls von Variablen gesprochen. Diese sind grundsätzlich keine statistchen Variablen, da sie keine Merkmale eines Objektes beschreiben, sondern Eingaben, die für jeden Simulationsdurchlauf unterschiedliche Werte annehmen können. Bei der statistischen Auswertung können daraus aber statistische Variablen erzeugt werden, dann nämlich, wenn sie als Merkmal eines übergeordneten Untersuchungsgegenstandes dienen. Später, beim Beschrieb der Implementierung, wird der Begriff im Sinne der Informatik verwendet, wo eine Variable als „Verbindung zwischen Name und Wert“ (Harvey & Wright, 1999) angesehen werden kann.

3.4 Verwendete Analyseverfahren

Für die Analyse der Daten werden einige Metriken der deskriptiven Statistik berechnet. Alle verwendeten Methoden werden in diesem Abschnitt kurz umrissen und ihre Aussagekraft für die analysierten Daten erklärt.

Arithmetisches Mittel

Das arithmetische Mittel, auch Mittelwert oder Durchschnitt genannt, dessen Ergebnis durch das Addieren aller Werte und der anschliessende Division durch die Anzahl der Werte berechnet wird:

$$\overline{x} = \frac{x_1 + x_2 + ... + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.1}$$

Varianz, Standardabweichung

Die Standardabweichung und die Varianz haben zum Ziel die Verteilung der Werte zu beschreiben. Die Varianz wird mittels Addition der quadrierten Abweichungen vom Mittelwert berechnet.

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{3.2}$$

Zieht man daraus die Quadratwurzel, erhält man die Standardabweichung.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{3.3}$$

Median, Quartil

Der Median soll, ähnlich dem arithmetischen Mittel, einen mittleren Wert der Verteilung angeben. Bildlich gesprochen, ist der Median derjenige Wert, der sich genau in der Mitte der Liste aller Werte befindet. Das erste Quartil ist der Median der unteren Hälfte der

Verteilung, das dritte Quartil der Median der oberen Hälfte. Der Median gilt allgemein als robuster gegenüber Extremwerten als der Mittelwert.

$$\text{Position des Median} = \frac{n+1}{2} \tag{3.4}$$

Boxplot

Der Boxplot ist die grafische Darstellung des „Five-Number Summary". Das sind fünf wichtige Kennzahlen zur Beschreibung eines Datensatzes bestehend aus Minimum, erstes Quartil, Median, drittes Quartil und Maximum. Zwischen den Quartilen wird eine Box aufgespannt und die sogenannte Antenne verbindet Minimum und Maximum. Zudem können Ausreisser (Stark von der Norm abweichende Werte) zusätzlich als einzelne Punkte dargestellt werden.

Balkendiagramm

Die bisher vorgestellten Kennzahlen können natürlich alle in tabellarischer Form als Zahl dargestellt werden. Für den Vergleich zwischen verschiedenen Datensätzen oder Variablen eignet sich zudem das Balkendiagramm, womit Unterschiede sehr gut grafisch aufgezeigt werden können. Jeder Balken repräsentiert dabei eine statistische Kennzahl.

Histogramm

Um eine Häufigkeitsverteilung der Werte innerhalb eines Datensatzes darzustellen, ist das Histogramm die richtige Wahl. Zuerst werden gleich grosse Kategorien gebildet, welche die ganze Breite der Daten abdecken. Weiter werden alle Werte in eine Kategorie eingeteilt. Auf der Ordinatenachse (Y-Achse) wird die Häufigkeit der Kategorien, auf der Abszissenachse die Kategorien selbst (jede als eigener Balken) dargtellt. Auf einen Blick kann so abgelesen werden, welche Werte häufig und welche weniger häufig gemessen wurden. Dabei gilt es zu beachten, dass immer nur eine Variable pro Diagramm dargestellt werden kann.

Streudiagramm

Das Streudiagramm ist äusserlich ein kartesisches Koordinatensystem. Dargestellt werden statistische Objekte. Grundsätzlich können nur zwei Variablen des Objekts dargestellt werden. Auf der X-Achse sind die Werte der einen Variablen und auf der Y-Achse diejenigen der zweiten abgetragen. Um auch Objekte höherer Dimension in einem Streudiagramm darstellen zu können, muss eine Reduktion der Dimension durchgeführt werden (siehe dazu Kapitel 4). Das Streudiagramm ist ein häufig eingesetztes Mittel zur Darstellung der Ergebnisse einer Clusteranalyse.

Zeitliche Darstellung

Die Darstellung der Daten im zeitlichen Verlauf kann mittels eines einfachen Liniendiagramms realisiert werden. Dabei werden auf der X-Achse die Daten (im Sinne des Plurals von Datum) in einem bestimmten Intervall abgetragen (z.B. Woche, Monat oder Jahr) und auf der Y-Achse, den zum jeweiligen Datum gemessenen oder ermittelten Wert. Zwischen

Werten und Daten besteht also eine Abhängigkeit und die Daten weisen eine natürliche Reihenfolge auf. Diese Eigenschaften können dazu verwendet werden, um mit Hilfe von Regressionsanalysen, die oftmals Beziehungen oder Abhängigkeiten zwischen statistischen Variablen nachzuweisen versuchen, Voraussagen zu erstellen. Man spricht in diesem Zusammenhang von Zeitreihenanalyse. Eine ausführliche Beschreibung würde an dieser Stelle zu weit führen. Kurz zusammengefasst, wird auf Basis früherer Werte versucht, ein Vorhersagemodell zu generieren, mit welchem zukünftige Werte prognostiziert werden können. Trends und saisonale Abweichungen werden dabei miteinbezogen.

Bedeutung für Daten

Als Abschluss dieses Kapitels, bleibt die Klärung der noch offenen Frage nach der Kategorisierung der Daten des Simulationsprogramms. Alle ausgewerteten Variablen, mit einer Ausnahme, sind quantitative Variablen. Die Ausnahme bezieht sich auf das Datum, welches jedoch auch nicht in Berechnungen miteinbezogen wird, sondern dazu dient, die zeitliche Ordnung zu bestimmen. Das ausschliessliche Vorhandensein quantitativer Variablen erspart einen grossen Mehraufwand an Transformation, der mit qualitativen Variablen, insbesondere bei den Clusteranalysen, hätte durchgeführt werden müssen. Doch auch bei quantitativen Variablen ist Vorsicht geboten und je nach Berechnungsmethode, bedarf es auch dort einer Modifikation. Beispiele dafür werden bereits im nächsten Kapitel aufgeführt.

Nochmals wichtig zu betonen ist, dass es sich bei den Daten um Zeitreihen handelt, also jedem gemessenen bzw. ausgewerteten Wert ein Zeitpunkt zugeordnet werden kann. Für das Programm bedeutet das, dass die Daten nicht einfach als ein Haufen von Werten ohne Reihenfolge behandelt werden können. Die Zuordnung zwischen Wert und Zeitpunkt muss während der Datenverarbeitung immer zweifelsfrei nachvollzogen werden können. Eine CSV-Datei stellt dabei eine Liste von Zeitreihen dar, eine Zeitreihe pro ausgewertete Variable.

4. Clusteranalyse

4.1 Grundlagen

4.1.1 Definition

Als zentrales statistisches Analyseinstrument der zu entwickelnden Software wird die Clusteranalyse eingesetzt. Ziel ist es, ähnliche oder zusammengehörige Objekte zu gruppieren. (Stein & Vollnhals, 2011)

Die Gruppen werden auch als Typen, Klassen oder eben Cluster bezeichnet. Clusteranalysen sind insbesondere dann sinnvoll, wenn in den Daten eine Gruppenstruktur vermutet wird, jedoch keine Kenntnis darüber vorliegt, welche Gruppen oder Klassen dies sein könnten. Obwohl manchmal synonym verwendet, ist die Clusteranalyse klar von der Klassifikation abzugrenzen. Dabei wird nicht von unbekannten Klassen ausgegangen, sondern es wird versucht, Objekte aus neuen Daten bereits bekannten und definierten Klassen zuzuordnen (Pretzer, 2003).

Dazu ein fiktives Beispiel, das den Unterschied verdeutlichen soll: Eine Forschergruppe sammelt Daten zu Planeten in unserer Galaxie, beispielsweise über chemische Zusammensetzung, Temperatur und Oberflächenstruktur. Nach einigen Jahren sollen diese Daten analysiert werden. Dabei sollen Planeten mit ähnlicher Beschaffenheit zusammengefasst und die gefundenen Gruppen benannt werden. Dies wäre ein klassischer Anwendungsfall der Clusteranalyse. Sollen hingegen aus den erhobenen Daten Planeten gefunden werden, welche der Erde ähneln, sozusagen der Klasse „erdähnliche Planeten“ angehören, würde man von Klassifikation sprechen. Dieser Unterschied wird deshalb so stark hervorgehoben, weil sich Methoden und Analyseverfahren stark unterscheiden und eine jeweils komplett andere Vorgehensweise nötig ist. Im Rahmen dieser Arbeit werden ausschliesslich Clusteranalysen und dazu in Zusammenhang stehende Verfahren betrachtet, da nur diese für die Erfüllung der Anforderungen an die Software relevant sind.

Die Clusteranalyse ist Bestandteil des Data-Mining, also der Extraktion von Wissen aus meist sehr grossen Datenbeständen. Sie ist zudem klar von anderen statistischen Verfahren, wie beispielsweise der Regressionsanalyse, abzugrenzen, die oftmals Beziehungen oder Abhängigkeiten zwischen statistischen Variablen nachzuweisen versuchen. (Stein & Vollnhals, 2011)

Die grundlegende Absicht einer Clusteranalyse ist also die Einteilung von Objekten in möglichst ähnliche Gruppen. Etwas präziser lässt sich dies in zwei Teilzielen formulieren (Bacher, Pöge, & Wenzig, 2010) :

1. Innerhalb eines Clusters soll möglichst hohe Homogenität (Intracluster-Homogenität) vorherrschen.
2. Zwischen den Clustern soll eine möglichst geringe Homogenität (Intercluster-Homogenität) oder möglichst hohe Heterogenität bestehen, die Clusters sollen sich also klar voneinander unterscheiden.

Anders ausgedrückt bedeutet das auch, dass zwischen den Clustern eine hohe Varianz und innerhalb der Cluster eine niedrige Varianz vorliegen sollte. (Stein & Vollnhals, 2011)

Es können auch noch weitere Anforderungen an das Ergebnis einer Clusteranalyse gestellt werden, die jedoch auch vom Kontext abhängig und nicht in jedem Fall sinnvoll sind. Als sehr sinnvoll erweist sich jedoch die Forderung, die Zahl der Cluster möglichst klein zu halten, da es bei steigender Anzahl immer schwieriger wird, den Gruppen eine inhaltliche Bedeutung zuzuweisen. (Bacher, Pöge, & Wenzig, 2010)

4.1.2 Ähnlichkeit und Distanz

Zentrales Konzept in Zusammenhang mit Clusteranalysen ist dasjenige der Ähnlichkeit. Wie aus der Definition ersichtlich, beruht die Clustereinteilung auf der Ähnlichkeit zwischen Objekten. Nun stellt sich natürlich die Frage, wie diese definiert beziehungsweise berechnet werden kann. Die Antwort hängt stark von der Art der statistischen Variablen ab. Da die Ausgangsdaten ausschliesslich quantitative Variablen enthalten, wird auch lediglich die Ähnlichkeitsmessung bei solchen Variablen behandelt. Für qualitative Variablen können zum Teil identische Berechnungsmethoden verwendet werden, oftmals müssen die Daten jedoch noch in eine andere Form transformiert werden. (Bacher, Pöge, & Wenzig, 2010)

Bei metrischen Variablen wird als Ähnlichkeitsmass sehr oft die Distanz zwischen Objekten verwendet, welche grösser gleich 0 sein muss. Dabei sind sich zwei Objekte umso ähnlicher, je kleiner die Distanz zwischen den beiden ist (ein Objekt weist zu sich selbst eine Distanz

von 0 auf). Im mathematischen Sinne bezeichnet die Distanz nichts anderes als den geometrischen Abstand zweier Punkte im n-dimensionalen Raum. Die Dimension ergibt sich aus der Anzahl miteinbezogener statistischer Variablen pro Objekt. Würden bei unserem Planetenbeispiel also Temperatur, Masse und Durchmesser als den Planeten (das Objekt) beschreibende Variablen gewählt, befände man sich im dreidimensionalen Raum. Zur konkreten Berechnung liegen zahlreiche Möglichkeiten (Distanzmasse genannt) vor, welche jedoch meist Spezialfälle der Minkowski-Metrik darstellen. (Stein & Vollnhals, 2011)

$$d(q,p)_{ij} = \left[\sum_{i=1}^{n} |x_{ik} - x_{jk}|^p\right]^{\frac{1}{p}} \tag{4.1}$$

Der Parameter q gibt die Dimension des Objektes an und p wird Metrikparameter genannt. Dieser ist es auch, der bei den verschiedenen Ausprägungen der Minkowski-Metrik variiert wird.

City-Block- oder Manhattan-Metrik

Bei der City-Block-Metrik (p=1) wird die Summe des Absolutbetrags der Differenzen berechnet. Ihr Name rührt von der Tatsache her, dass Distanzen so gemessen werden, wie man sie bei der Fortbewegung in einer Grossstadt mit rechtwinkligem Strassennetz (wie eben New York bzw. Manhattan) auch beobachten würde.

$$d_1(x_i,x_j)_{ij} = \left[\sum_{i=1}^{n} |x_{ik} - x_{jk}|^1\right]^{\frac{1}{1}} = \sum_{i=1}^{n} |x_{ik} - x_{jk}| \tag{4.2}$$

Euklidischer Abstand

Der Euklidische Abstand ist die wohl häufigste Form der Distanzmessung. Er misst die „Luftlinie“ zwischen zwei Objekten; der Parameter p wird auf 2 festgesetzt. Ein Merkmal des Euklidischen Abstandes ist, dass Distanzen zwischen Variablen kompensiert werden können (eine grosse Distanz in einer Variable kann durch eine kleine Distanz in einer anderen aufgewogen werden).

$$d_2(x_i,x_j)_{ij} = \left[\sum_{i=1}^{n} |x_{ik} - x_{jk}|^2\right]^{\frac{1}{2}} = \sqrt[2]{\sum_{i=1}^{n} (x_{ik} - x_{jk})^2} \tag{4.3}$$

Canberra-Distanz

Die Canberra-Distanz berechnet sich im Prinzip gleich wie die City-Block-Metrik, die Differenz des Absolutbetrags wird jedoch noch durch die Summe der absoluten Werte der einzelnen Variablen geteilt, was einer Gewichtung gleichkommt.

$$d_{Canberra}(x_i, x_j)_{ij} = \sum_{i=1}^{n} \frac{|x_i - x_j|}{|x_i| + |x_j|} \tag{4.4}$$

Tschebyscheff

Bei der Tschebyscheff-Metrik wird nicht eine Summe gebildet, sondern es wird die maximale Differenz des Absolutbetrages als Distanz angenommen (p ist gleich unendlich).

$$d_{Tschebyscheff}(x_i, x_j)_{ij} = \lim_{g \to \infty} \left[\sum_{i=1}^{n} |x_{ik} - x_{jk}|^g \right]^{\frac{1}{g}} = \max_{i,j} \{|x_{ik}, x_{jk}|\} \tag{4.5}$$

Distanzmatrix

Unabhängig welches Distanzmass gewählt wird, muss in einem nächsten Schritt zur Bestimmung der Ähnlichkeit, die Distanz zwischen allen Objekten berechnet werden. Es wird davon ausgegangen, dass die Daten in tabellarischer Form mit den Objekten als Zeilen und den Variablen als Spalten vorliegen. Nun wird zwischen jedem Objektpaar die Distanz berechnet. Das Ergebnis ist die Distanzmatrix, woraus die Distanzen eines Objektes zu jedem anderen Objekt im Datensatz ersichtlich sind. Die Matrix ist symmetrisch und enthält in der Hauptdiagonalen lauter Nullen (Distanz des Objektes zu sich selbst). Die Dimension der Matrix entspricht natürlich der Anzahl der Objekte. Werden also 10 Objekte untersucht, enthält die Distanzmatrix 10 Zeilen und 10 Spalten (10 x 10 Matrix). Sind die Distanzen erst einmal berechnet und liegt die Distanzmatrix vor, kann mit der Clusteranalyse begonnen werden.

Tabelle 4.1: Beispiel einer Distanzmatrix

	Objekt 1	**Objekt 2**	**Objekt 3**	**Objekt n**
Objekt 1	0	Distanz	Distanz	Distanz
Objekt 2	Distanz	0	Distanz	Distanz
Objekt 3	Distanz	Distanz	0	Distanz
Objekt n	Distanz	Distanz	Distanz	0

4.1.3 Normalisierung der Daten

Betrachtet man das Konzept der Ähnlichkeit und der Distanzen wird schnell klar, dass gewisse Anforderungen an das Datenmaterial gestellt werden. Eine Berechnung der Distanz zwischen zwei Objekten ist nämlich nur dann erlaubt und auch sinnvoll, wenn sich die in

die Berechnung einbezogenen statistischen Variablen auch vergleichen lassen. Im vorhergehenden Beispiel wurden Temperatur, Masse und Durchmesser als Variablen verwendet. Alle drei werden in unterschiedlichen Masseinheiten gemessen und können auch von den absoluten Werten stark voneinander abweichen. Nehmen wir an, dass die Temperatur in Grad Celsius und die Masse in Kilogramm gemessen werden. Realistische Werte für einen Planeten wären beispielsweise 80 °C für die Temperatur und 639×10^{21} kg für die Masse. Für das Ermitteln der Distanz würde die Temperatur beinahe überhaupt keine Rolle spielen, da der Wert der Masse um ein Vielfaches grösser ist und somit die Berechnung dominiert (Cornish, 2007). Doch nicht nur bei verschiedenen Masseinheiten ist Vorsicht angebracht, sondern auch bei unterschiedlichen Mess- oder Skalenniveaus. Werden Distanzen einmal in Metern und bei einer anderen Variable in Millimetern angegeben, entsteht der gleiche Effekt. Um dem entgegenzuwirken, kann eine vorgängige Standardisierung der Daten vorgenommen werden. Dabei wird meist die z-Transformation eingesetzt, wobei vom Wert der Variable, der Mittelwert aller Werte der Variable subtrahiert und durch die Standardabweichung geteilt wird. Dies führt dazu, dass die Variable nach der Transformation eine Varianz von 1 und einen Erwartungswert von 0 aufweist.

$$z = \frac{x_i - \overline{x}}{s} \tag{4.6}$$

Diese Art der Standardisierung ist jedoch nicht ganz frei von Problemen. Wie in der Definition eines Clusters beschrieben, fördert eine grosse Varianz in den Daten die klare Ausbildung von Clustern. Durch die Standardisierung wird jedoch die Varianz und somit auch die Distanz zwischen den Clustern reduziert. Inwieweit sich dies negativ auf das Ergebnis auswirkt, bzw. eine andere Clusterzusammenstellung nach sich zieht, lässt sich von vornherein nicht zweifelsfrei bestimmen. Während beispielsweise Bachel et al. (2010) dazu raten, immer eine Standardisierung durchzuführen, schlägt Cornish (2007) vor, die Analyse zweimal durchzuführen, einmal mit und einmal ohne Transformation, und die Resultate zu vergleichen. Wird eine Standardisierung durchgeführt, muss dies natürlich geschehen, bevor die Distanzen berechnet werden und die Distanzmatrix aufgestellt wird.

4.1.4 Typen von Clusteranalysen

Es können drei Typen von Clusteranalysen anhand ihrer Zuordnungslogik unterschieden werden: Unvollständige Clusteranalysen, deterministische Clusteranalysen und probabilistische Clusteranalysen. Diese Typisierung, wie auch die weitere Untergliederung, ist keineswegs die einzige Möglichkeit Clusterverfahren in Kategorien einzuteilen. Sie folgt dem Vorschlag von Bacher et al. (2010), die eine gut strukturierte und nachvollziehbare Abgrenzung der Analysen vornehmen. Wenn bisher von „Clusteranalyse“ die Rede war, so waren damit die

deterministischen Clusteranalysen gemeint, welche für diese Arbeit von besonderem Interesse sind. Nachfolgend werden alle drei Kategorien kurz vorgestellt und wichtige Vertreter der jeweiligen Kategorie beschrieben. Ziel ist es, einen groben Überblick über die gängigen Verfahren zu geben und deren Eigenschaften genauer zu beleuchten. Die Aufzählung ist jedoch weder abschliessend, noch werden alle Analyseverfahren Eingang in die Software finden. Das Vorgehen unterscheidet sich also von dem des vorhergehenden Abschnittes, wo alle beschriebenen Methoden auch im Programm verwendet wurden. Um eine geeignete Wahl bezüglich der Clustering-Algorithmen treffen zu können, bedarf es aber zuerst einer Übersicht über die Möglichkeiten, sozusagen einer Palette an Algorithmen, aus der dann die geeignetsten implementiert werden. Anschliessend wird in Abschnitt 5 die Wahl der Verfahren für die Software beschrieben, begründet und der direkte Bezug zur Problemstellung hergestellt. Bezüglich der Terminologie sei an dieser Stelle noch darauf hingewiesen, dass die Begriffe Clusterverfahren, Clustermethode und Clustering-Algorithmus alle dasselbe bedeuten und so auch synonym verwendet werden.

4.2 Unvollständige Clusteranalysen

Die unvollständigen Clusteranalysen bilden die erste Gruppe der Clusteranalysen und verdanken ihren Namen der fehlenden Fähigkeit, Objekte in Cluster einzuteilen. Es mag im ersten Moment merkwürdig anmuten, dass diese Methoden überhaupt als Clusteranalysen bezeichnet werden, mangelt es doch genau an der Eigenschaft, die man als Hauptaufgabe erwarten würde. Tatsächlich fehlen den unvollständigen Clusteranalysen inhärente Entscheidungsregeln, wann ein Objekt einem bestimmten Cluster zugeordnet werden kann,beziehungsweise was überhaupt als Cluster gilt. Dennoch bedeutet dies aber keineswegs, dass sie für eine Clusteranalyse nicht empfehlenswert oder ungeeignet wären. Sie erfüllen zwei zentrale Aufgaben: Zum einen können sie als Vorbereitung für ein deterministisches Clusterverfahren dienen und zum anderen bieten sie die Möglichkeit einer räumlichen Darstellung der Objekte. Die hier vorgestellten Verfahren werden nicht allgemein, sondern nur im Kontext der Clusteranalyse beleuchtet, weitere Anwendungsmöglichkeiten werden nicht behandelt. (Bacher, Pöge, & Wenzig, 2010)

Wird eine unvollständige Clusteranalyse als vorbereitendes Instrument eingesetzt, spricht man auch von einer Dimensionsreduktion. Je höher die Dimension eines Problems, desto schwieriger ist es für statistische und mathematische Analyseverfahren (und somit auch Clusteranalysen), eine sinnvolle Lösung zu finden. Zudem kann oft beobachtet werden, dass nicht alle Dimensionen gleich wichtig und relevant für das Ergebnis sind. Im Kontext von Computerprogrammen bedeutet dies zusätzlich, dass die Durchführung der Datenanalyse mit einer

längeren Berechnungszeit und einer höheren Speicherbeanspruchung verbunden ist. Eine Reduktion der Dimension kann also einen positiven Einfluss auf die Aussagekraft einer Analyse haben. „Kann“ deswegen, weil durchaus die Möglichkeit besteht, dass durch die Reduktion wichtige Informationen verloren gehen. So ist es nicht möglich, im Vornherein mit Sicherheit festzustellen, ob eine Dimensionsreduktion das Ergebnis verbessern würde oder einen starken Verlust relevanter Informationen nach sich zöge. Dabei spielt vor allem die Beschaffenheit der Daten die zentrale Rolle. Die Verfahren treffen nämlich oftmals Annahmen bezüglich gewisser Eigenschaften der Daten. Sind diese Annahmen im untersuchten Datensatz falsch, leidet die Qualität des Ergebnisses. (Fodor, 2002)

Die zweite Einsatzmöglichkeit der unvollständigen Clusteranalysen (räumliche Darstellung) ist vom analytischen Standpunkt betrachtet identisch mit der Vorbereitung für weitere mustererkennende Methoden. Auch hier wird im Grunde wieder eine Dimensionsreduktion durchgeführt. Die Intention ist jedoch eine leicht andere. Ziel ist es nicht unbedingt das Gesamtverfahren zu beschleunigen, als vielmehr eine Projektion in eine darstellbare Dimension zu finden. Deshalb ist die Reduktion in diesem Fall auch nur sinnhaft, wenn als Ergebnis eine 1-3 dimensionale Lösung gefunden werden kann. Werden die Resultate entsprechend visualisiert, kann natürlich auch eine Clustereinteilung ”von Auge”vorgenommen werden. Dabei wird die Verantwortung auf den Benutzer übertragen und die Einteilung ist dadurch zu einem gewissen Teil auch subjektiv geprägt. (Bacher, Pöge, & Wenzig, 2010)

4.2.1 Multidimensionale Skalierung

Die Multidimensionale Skalierung (MDS) bezeichnet eine Vielzahl verschiedener statistischer Verfahren, die alle versuchen, eine räumliche Darstellung der Ähnlichkeiten von Objekten so präzise wie möglich zu finden. Ein wichtiger Vertreter der MDS ist die metrische multidimensionale Skalierung. Ausgangspunkt ist eine Distanzmatrix, welche aus den Ursprungsdaten berechnet wurde. Die MDS verwendet somit Distanzen als Ähnlichkeitsmass, je weiter die Objekte also voneinander entfernt sind, desto unähnlicher sind sie sich. Das Ergebnis der MDS, das als Konfiguration bezeichnet wird, ist eine Repräsentation der Distanzen zwischen den Objekten in einem tiefer-dimensionierten Raum. Da die MDS ein iteratives und kein analytisches Verfahren ist, besteht die Gefahr, dass sich der Algorithmus in lokalen Minima verfängt und die Konfiguration dementsprechend suboptimal ist. (Borg, 2000)

Die Güte der MDS wird mittels des sogenannten „standardized residual sum of squares“, kurz Stress, gemessen. Je tiefer dabei der Stress, desto exakter widerspiegelt die Konfiguration die tatsächlichen Distanzen zwischen den Objekten. Um den Stress einer Konfiguration absolut zu bewerten, kann nach Borg (2000) der Stress von Zufallsdaten herangezogen werden. Dieser beträgt ungefähr 0.24. Allgemein geht man davon aus, dass nur Konfigurationen mit einem

Stress kleiner 0.2 aussagekräftig sein können. Wichtig zu beachten ist die Tatsache, dass der Stress umso grösser wird, je stärker die Zieldimension von der ursprünglichen Dimension der Daten abweicht. Dies sollte bei der Entscheidung auf welche Dimension die Daten reduziert werden beachtet werden. (Bacher, Pöge, & Wenzig, 2010)

4.2.2 Hauptkomponentenanalyse

Ziel der Hauptkomponentenanalyse (PCA – englisch: principal component analysis) ist es, die relevantesten statistischen Variablen eines Objektes zu bestimmen, die sogenannten Hauptkomponenten. Relevant heisst in diesem Kontext, dass eine Variable möglichst viel von der Gesamtvarianz in den Daten erklären kann. Bestehen also die Objekte eines Datensatzes beispielsweise aus 10 statistischen Variablen, kann es vorkommen, dass nur drei Variablen für 90% der Varianz verantwortlich sind. Diese drei Variablen würden den Datensatz ausreichend beschreiben und die restlichen Variablen könnten weggelassen werden, ohne damit einen allzu grossen Informationsverlust zu verursachen. In vielen Fällen lässt sich somit die Dimension der Daten stark reduzieren. Für die räumliche Darstellung im zweidimensionalen Raum werden die ersten beiden Hauptkomponenten (die Hauptkomponenten mit dem grössten und zweitgrössten Anteil an der Gesamtvarianz) geplottet. Wie bei der MDS gilt auch hier, dass die Genauigkeit abnimmt, je weiter Ursprungsdimension und Zieldimension auseinander liegen, da die Wahrscheinlichkeit immer kleiner wird, die Information in den Daten adäquat mit nur zwei oder drei statistischen Variablen wiederzugeben. Ausgangspunkt für die konkrete Berechnung ist immer die Kovarianzmatrix (Paarweise Gegenüberstellung aller Kovarianzen). Die Hauptkomponenten können danach entweder über die Bestimmung der Eigenwerte und Eigenvektoren oder mittels einer Singulärwertzerlegung (SVD – englisch: Singular Value Decomposition) gefunden werden. (Shlens, A Tutorial On Principal Component Analysis, 2003)

4.3 Deterministische Clusterverfahren

Die Kategorie der deterministischen Clusterverfahren kann sicherlich als bedeutendste Gruppe bezeichnet werden. Oftmals wenn von Clusteranalyse gesprochen wird, ist gar nur diese Kategorie gemeint. Grund für die Namensgebung ist die Tatsache, dass die deterministischen Clusterverfahren, ganz im Gegensatz zu den unvollständigen, nun auch wirklich eine Einteilung von Objekten in voneinander verschiedene Cluster vornehmen können. Dabei gibt es Algorithmen, welche die Objekte genau einem Cluster zuweisen (überlappungsfreie Cluster) und solche, die auch die Zugehörigkeit von Objekten zu mehreren Clustern erlauben (überlappende Cluster). Für diese Arbeit sind vorwiegend erstere geeignet, da ein Interesse

besteht, möglichst klar voneinander getrennte Cluster zu finden. Die Problematik der Clustereinteilung wird also von den deterministischen Clusterverfahren gelöst, eine eindeutige Bestimmung der Clusteranzahl bleibt jedoch auch hier, im Umfang abhängig von der konkreten Methode, ein Problem.

4.3.1 Hierarchische Clusterverfahren

Die hierarchischen Clusterverfahren können als Urväter der clusteranalytischen Methoden bezeichnet werden und bilden auch die Grundlage für später entwickelte Algorithmen. Sie können wiederum in zwei Untergruppen mit unterschiedlicher Vorgehensweise unterteilt werden: Die agglomerativen und die divisiven Verfahren. (Bacher, Pöge, & Wenzig, 2010)

Die agglomerativen Methoden folgen dem „bottom-up"-Ansatz. Zu Beginn stellt jedes Objekt ein eigenes Cluster dar, welche danach kontinuierlich verschmolzen werden, bis nur noch ein, alle Objekte umfassendes Cluster besteht. Die Entscheidungsgrundlage, welche Cluster in einem Schritt verschmolzen werden sollen, liefert die Berechnung der Distanzen (nahe Cluster werden verschmolzen). Den entgegengesetzten Weg gehen die divisiven Verfahren („top-down"-Ansatz), alle Objekte befinden sich anfangs in einem Cluster, dieses wird so lange zerlegt, bis für jedes Objekt ein Cluster vorhanden ist. Beiden Verfahren gemein ist, dass getroffene Entscheidungen, also die Teilung oder das Zusammenfügen eines Clusters, nicht mehr rückgängig gemacht werden können. Auch liefern bei beiden Techniken die Endresultate offensichtlich nicht die gewünschten Erkenntnisse. Das optimale Resultat ist ein Zwischenergebnis, das sich irgendwann während der Durchführung des Verfahrens einstellt. Dazu muss ein sogenannter Schwellenwert gewählt werden. Dieser gibt die maximale Distanz zwischen zwei Clustern an. Cluster die näher beieinander liegen, werden während der Analyse verschmolzen. (Kopp & Lois, 2009)

Je tiefer der Schwellenwert, desto grösser die Anzahl der Cluster, da mit sinkendem Schwellenwert eine immer kleinere Maximaldistanz zwischen den Objekten verlangt wird. In der Praxis finden fast nur die agglomerativen Clusteranalysen Verwendung, da die divisiven Verfahren viel rechenintensiver und auch komplexer zu implementieren sind (Manning, Raghavan, & Schütze, 2009). Aus diesem Grund wird in der Folge auch nur auf die agglomerativen Verfahren eingegangen. Zur Visualisierung der Ergebnisse von hierarchischen (agglomerativen) Clusterverfahren wird meist das Dendrogramm verwendet, welches den Fusionierungsprozess gut darzustellen vermag. Ein Dendrogramm zeigt auf der Y-Achse die möglichen Ausprägungen des Schwellenwerts. Die Distanzen werden auf 1 normiert, 0 bedeutet dabei keine Distanz und 1 die grösstmögliche. Somit liegt auch der Wertebereich des Schwellenwerts zwischen und inklusive 0 (alle Objekte bilden ein eigenes Cluster) und 1 (es existiert noch ein Cluster

mit allen Objekten). Auf der X-Achse sind die einzelnen, durchnummerierten Objekte abgetragen. Ein horizontaler Strich bedeutet, dass zwei Cluster verschmolzen werden. Mit Hilfe des Dendogramms kann sehr schön abgelesen werden, wann welche Objekte verschmolzen werden. Dies unterstützt auch das Finden eines optimalen Schwellenwerts, der die Herausbildung einer Lösung mit grösster Homogenität innerhalb und grösster Heterogenität zwischen den Clustern ermöglicht.

Obgleich demselben Prinzip folgend, unterscheiden sich die verschiedenen Algorithmen der hierarchisch agglomerativen Clusterverfahren in der Berechnung der Distanz zwischen zwei bereits bestehenden Clustern. Dies ist nicht mit der Berechnung der Distanz zwischen zwei einzelnen Objekten zu verwechseln, die auch hier mit einem der bereits vorgestellten Distanzmasse erfolgt. Für die Ermittlung der Distanz zwischen zwei Clustern, müssen jedoch Berechnungen für ganze Mengen von Objekten vorgenommen werden, was auf unterschiedliche Art und Weise geschehen kann. (Stein & Vollnhals, 2011)

Single-Linkage
Als Abstand zwischen den Clustern wird die kleinste Distanz zwischen den zwei nächstgelegenen Objekten eines jeden Clusters definiert. Dieses Verfahren kann eingesetzt werden, um Ausreisser zu erkennen, da diese erst ganz am Ende einem Cluster zugeordnet werden.

$$D(X,Y) = \min(d(x,y)) \tag{4.7}$$

D(X,Y) ist als Distanz zwischen zwei Clustern X und Y zu verstehen, d(x,y) als Abstand zwischen zwei Elementen x aus dem Cluster X und y aus Cluster Y.

Complete-Linkage
Der Complete-Linkage-Algorithmus verwendet nicht die minimale, sondern maximale Distanz. Dabei werden die Cluster zusammengelegt, deren maximaler Abstand zwischen den Objekten am kleinsten ist.

$$D(X,Y) = \max(d(x,y)) \tag{4.8}$$

Average-Linkage
Wie der Name schon andeutet, werden die durchschnittlichen Abstände zweier Cluster als Verschmelzungskriterium herangezogen. Im Vergleich zu Single- und Complete-Linkage gehen alle Elemente eines Clusters in die Berechnung mit ein.

$$D(C_k, Cj) = \min \left\{ \frac{1}{n_k n_j} \sum_{n \in C_k} \sum_{m \in C_j} d_{nm} \right\} \tag{4.9}$$

Wobei n_k und n_j als die Anzahl in den jeweiligen Clustern enthaltenen Objekte zu verstehen sind.

Zentroid-Verfahren

Bei der Zentroid-Methode werden zuerst die Schwerpunkte eines jeden Clusters bestimmt. Die beiden Cluster, deren Schwerpunkte am Nächsten beieinander liegen, berechnet über den paarweisen Vergleich der quadrierten Abweichung, werden dann fusioniert.

$$D(C_j, C_k) = \min_{k \neq j} ||\bar{x}_j - \bar{x}_k||^2 \tag{4.10}$$

Mit dem Schwerpunkt eines Clusters:

$$\bar{x}_k = \frac{1}{n_k} \sum_{n \in C_k} x_n \tag{4.11}$$

Ward-Verfahren

Beim Ward-Verfahren werden diejenigen Cluster verschmolzen, deren Zusammenlegung den kleinsten Anstieg der Varianz im neu geschaffenen Cluster nach sich ziehen würde. Dies lässt sich durch den Vergleich der Fehlerquadratsummen zwischen jedem Clusterpaar bestimmen.

$$D_W(C_j, C_k) = \frac{n_j n_k}{n_j + n_k} ||\bar{x}_j - \bar{x}_k||^2 \tag{4.12}$$

4.3.2 Partitionierende Verfahren

Der wichtigste Vertreter der partitionierenden Verfahren ist sicherlich der k-Means-Algorithmus. Der Name wird oft etwas ungenau verwendet, da mit dem Begriff „k-Means“ eigentlich nur das generelle Minimierungsproblem gemeint ist. Dabei wird der Datensatz vom Algorithmus derart in k Partitionen zerlegt, dass der Abstand zwischen Objekten und Clustermittelpunkt minimal wird. Aufgrund der hohen Komplexität (NP-schwer), wird meist auf einen approximativen Algorithmus zurückgegriffen. Die an dieser Stelle präsentierte Variante, ist diejenige des Lloyd-Algorithmus.

1. Zu Beginn werden so viele Objekte als Mittelpunkte ausgewählt, wie Cluster gefunden werden sollen. Dies geschieht rein zufällig.
2. Alle restlichen Objekte werden dem Mittelpunkt mit dem kleinsten Abstand zugeordnet.
3. Nach abgeschlossener Verteilung, werden die Clusterzentren neu berechnet.
4. Die Schritte 2 und 3 werden wiederholt, bis die Cluster stabil sind, also von einer Iteration zur nächsten keine Objekte mehr einem anderen Cluster zugeordnet werden.

Aus dem Beschrieb des Algorithmus können auch schon die beiden gravierendsten Schwächen der Methode abgelesen werden: Einerseits ist die Lösung in gewissem Masse abhängig von der

gewählten Startkonfiguration (was je nachdem zu einer nicht optimalen Lösung führen kann) und andererseits muss die Anzahl der Cluster von aussen vorgegeben werden. Mit dem Ziel, diesen Problemen entgegenzuwirken, wurden auf Basis des k-Means verschiedene Variationen entwickelt. Eine davon ist der X-Means. Beim X-Means müssen lediglich die untere und obere Grenze der Clusterzahl angegeben werden, der Algorithmus berechnet für jede Clusterzahl ein Ergebnis und weist diesem eine Bewertung zu. Das Ergebnis mit der besten Bewertung wird dann als Endresultat festgelegt. (Pelleg & Moore, 2000)

4.3.3 Dichtebasierte Verfahren

Dichtebasierte Verfahren erkennen Cluster anhand einer Häufung von Punkten in einem gewissen Radius. Das wohl prominenteste Verfahren ist der DBSCAN (density-based spatial clustering of applications with noise). Der Algorithmus benötigt zwei Eingabeparameter: Die maximale Distanz, die zwei Punkte haben dürfen, um noch Nachbarn zu sein (genannt ε), sowie die minimale Anzahl Datenpunkte, die ein Cluster enthalten muss. Der Algorithmus wählt dann zufällig ein Objekt und prüft, ob in seiner Umgebung (auch Nachbarschaft genannt) mit Radius ε genügend Punkte (grösser-gleich der vorgegebenen Mindestanzahl) vorhanden sind, um ein Cluster zu bilden. Falls ja, wird diese Region als Cluster deklariert und es werden weitere Punkte gesucht, die zum selben Cluster gehören. Falls nein, gilt der Punkt als sogenannter Ausreisser, allgemein der statistische Begriff für einen Datenpunkt oder Messwert abseits der erwarteten Werte und hier im speziellen als Objekt, der sonst zu keinem Cluster gehört. Während die Ergebnisse von partitionierenden Verfahren durch Ausreisser verzerrt werden können, ist der DBSCAN-Algorithmus fähig, solche zu erkennen. Nach dem Auffinden eines Ausreissers wird wieder zufällig ein bisher noch nicht untersuchter Datenpunkt ausgewählt und dessen Nachbarschaft geprüft. Dies wird solange durchgeführt, bis alle Cluster gefunden sind. Neben der Fähigkeit zum Auffinden von Ausreissern, liegt der grosse Vorteil im Vergleich zum k-Means darin, dass keine Clusteranzahl vorgegeben werden muss. Der Nachteil darin, dass ein fixes ε festgelegt werden muss. Zwar existieren Faustregeln wie ε festgelegt werden könnte, beispielsweise als Mittelwert aller Distanzen in der Distanzmatrix, jedoch können generell nur Cluster mit ähnlicher Dichte gefunden werden, da ε nicht variiert werden kann. (Ester, Kriegel, Sander & Xu, 1996)

4.4 Probabilistische Verfahren

Im Gegensatz zu den deterministischen Verfahren, weisen die probabilistischen Algorithmen die Objekte einem Clustern nur mit einer gewissen Wahrscheinlichkeit zu. So können natürlich auch überlappende Cluster entstehen. Um überlappungsfreie Cluster zu erzwingen,

kann ein Objekt auch fix demjenigen Cluster zugewiesen werden, zu welchem es am wahrscheinlichsten gehört. Die Zuordnung erfolgt aufgrund der statistischen Verteilung der Objekte. Zu Beginn werden die statistischen Verteilungen zufällig geschätzt und in einem iterativen Prozess nach und nach optimiert. Dies führt jedoch dazu, dass verschiedene Durchläufe nicht unbedingt zum gleichen Ergebnis führen müssen. Ein häufig verwendetes Beispiel für ein probailistisches Verfahren ist der Expectation-maximization-Algorithmus (abgekürzt EM).

5. Wahl der Algorithmen

Nach der Übersicht über die Clusterverfahren in Kapitel 4, geht es nun um die Bestimmung eines Algorithmus, der standardmässig das Clustering im Programm vollziehen soll. Dieses Kapitel zeigt auf, welcher Algorithmus warum ausgewählt wurde. Zudem soll erklärt werden, wie das Clustering konkret umgesetzt wurde.

5.1 Wahl des Algorithmus

Bei der Wahl der Clusteralgorithmen musste nicht nur auf die Eignung, sondern auch auf die Verfügbarkeit Rücksicht genommen werden. Ziel war es natürlich ein Framework oder eine Library zu finden, die möglichst alle Ansprüche abdeckte, als stabile Version frei zur Verfügung steht und, zur optimalen Einbindung, in Java geschrieben ist. Leider war dies nicht vollumfänglich möglich, da keine Bibliothek alle Aspekte abdecken konnte. Für das (deterministische) Clustering wurde Weka (Akronym für Waikato Environment for Knowledge Analysis) ausgewählt. Ein an der Universität von Waikato entwickeltes Programm, welches vor allem für Klassifikationsaufgaben verwendet wird, zudem aber auch einige Clusteralgorithmen zu Verfügung stellt. Weka kann als eigenständiges Tool verwendet werden, bietet jedoch auch eine Java-Programmierschnittstelle.

Klar war somit, dass ein Algorithmus aus der Palette des Weka-Frameworks ausgewählt werden musste. Dies umfasst alle in Kapitel 4 beschriebenen deterministischen Verfahren sowie den EM-Algorithmus und noch einige weitere. Generell existiert nicht der vollkommene Clusteralgorithmus, welcher in jedem Fall bessere Resultate liefert als alle anderen. Vielmehr hängt es stark vom Datenmaterial und den gesuchten Resultaten ab. Clusteranalysen werden häufig in Verbindung respektive zur Bestätigung einer Hypothese eingesetzt. Vermutet wird beispielsweise eine gewisse Struktur oder eine gewisse Anzahl an unterschiedlichen Gruppen in den Daten, was dann mit Hilfe der Clusteranalyse bestätigt oder verworfen werden soll. Dafür sind Algorithmen wie z.B. der k-Means gut geeignet, da bereits eine Clusterzahl vermutet wird. (Bacher et al, 2010)

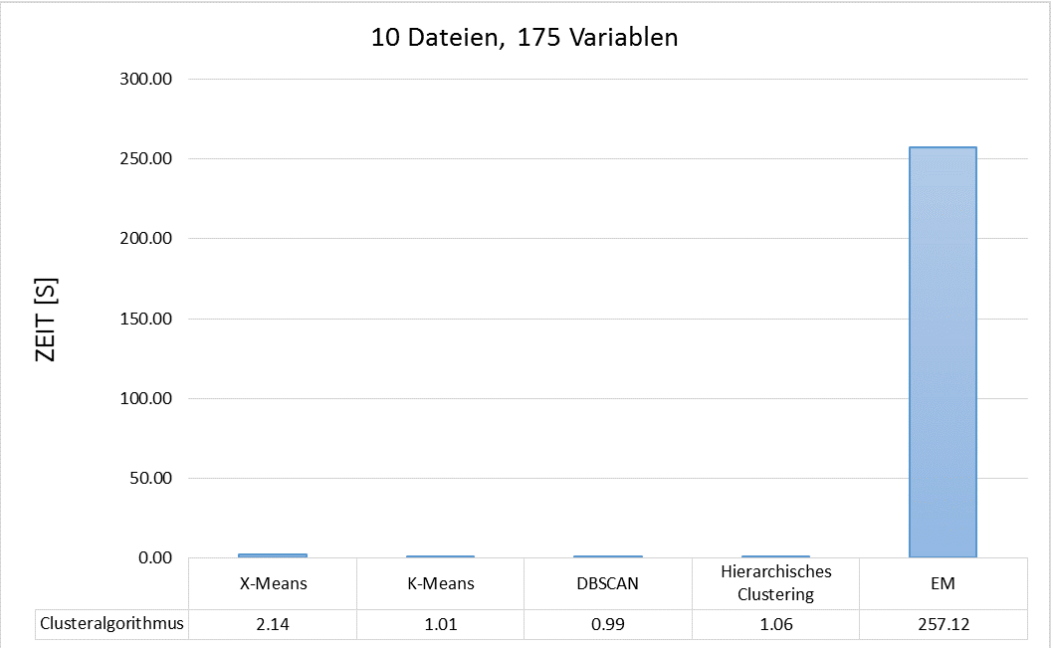


Abbildung 5.1: Zeitvergleich der Clusteralgorithmen 10 Dateien

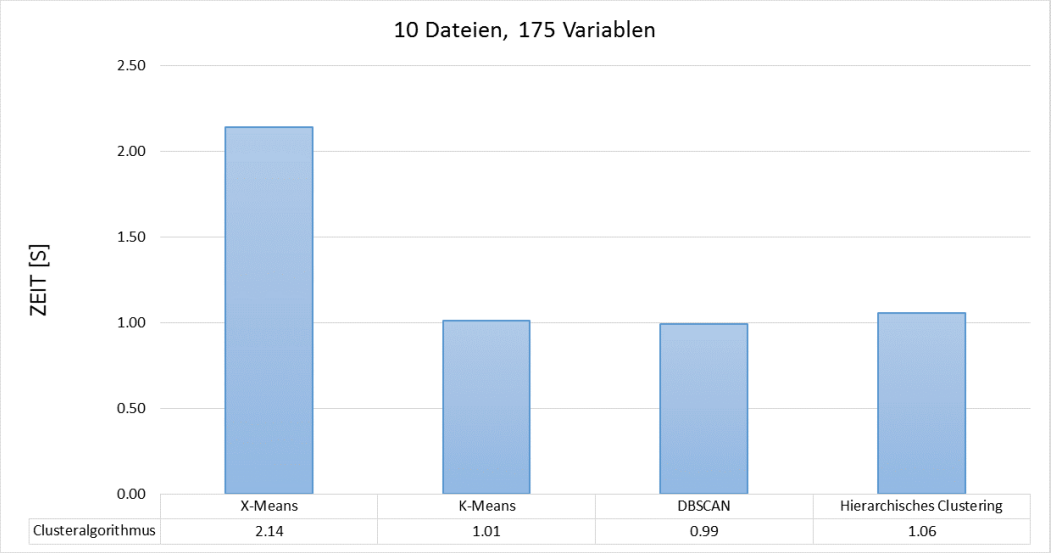


Abbildung 5.2: Zeitvergleich der Clusteralgorithmen 10 Dateien ohne EM

Bei den mit dem Programm dieser Arbeit zu untersuchenden Daten ist dies jedoch nicht der Fall. Es geht gerade darum, etwaige Muster überhaupt zu entdecken. Die Implementierung des EM-Algorithmus in Weka funktioniert, ohne vorgängig eine Clusterzahl angeben zu müssen. Dies geht jedoch stark auf Kosten der Geschwindigkeit, womit auch ein weiteres Kriterium für den auszuwählenden Algorithmus ersichtlich wird. Aufgrund der erwarteten zu analysierenden Datenmenge, muss ein performanter Algorithmus gewählt werden, um unzumutbar lange Wartezeiten zu vermeiden (länger als 1 Minute). Wie aus Abbildung 5.1 hervorgeht scheidet der EM-Algorithmus somit klar aus.

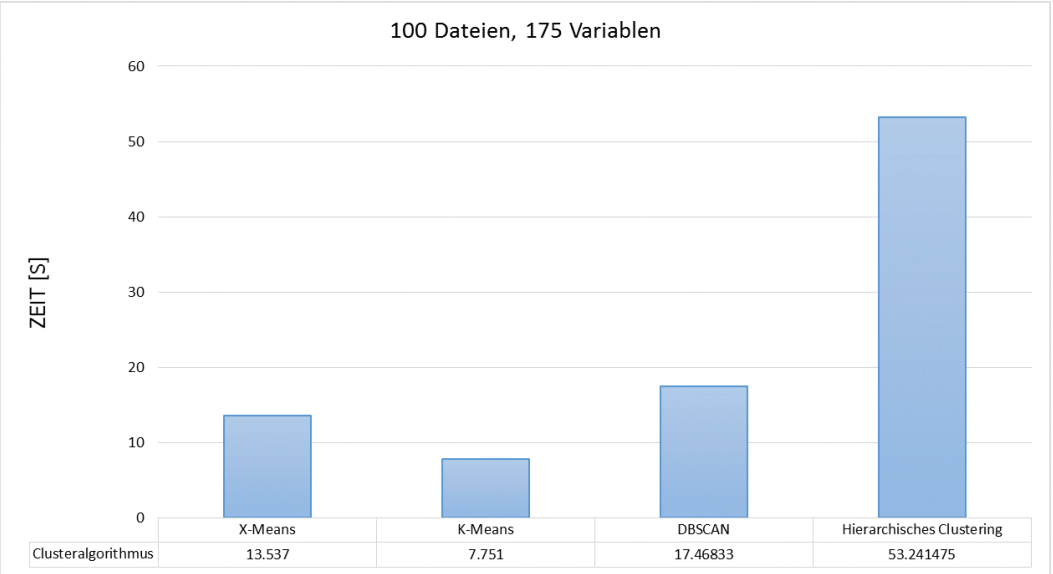


Abbildung 5.3: Zeitvergleich der Clusteralgorithmen 100 Dateien

Auch der DBSCAN kommt ohne Angabe der Clusterzahl aus, benötigt jedoch die beiden Parameter ε und die Mindestanzahl an Objekten für ein Cluster. Eine Herausforderung stellt dabei vor allem die Bestimmung von ε dar. Durch die Standardisierung der Daten vor dem Clustering wird dieses Problem etwas gemildert; trotzdem kann nicht garantiert werden, dass eine Einstellung befriedigende Resultate für alle Fälle (also verschiedene statistische Variablen und Objekte) liefert. Zudem ist die Gefahr gross, dass aussergewöhnliche und eigentlich interessante Objekte als Ausreisser taxiert werden und somit verloren gehen. Eine Möglichkeit dem entgegenzuwirken wäre zwar die Mindestzahl an Objekten auf 1 zu setzten, jedoch muss dann ε so gewählt werden, dass nicht zu viele Cluster mit nur einem Objekt entstehen.

Gewählt wurde schlussendlich der X-Means-Algorithmus. Ganz ohne Angabe zusätzlicher Parameter kommt auch er nicht aus, so müssen die minimale und maximale Anzahl an Clustern vorgegeben werden. Dies ist jedoch aus Sicht des Autors ein nicht allzu schwerwiegendes Problem. Die minimale Anzahl wurde auf 2 festgelegt, da eine Aufteilung in zwei Gruppen doch häufig vertretbar und wünschenswert ist. Für Situationen in denen dies überhaupt nicht zutrifft, ist vorgesorgt: Bei einem Clustering von völlig identischen Objekten (oder genauer mit einer Distanz von 0 zueinander), würden alle Objekte nur einem Cluster zugeteilt, das zweite Cluster enthielte dann einfach keine Objekte. Als maximale Clusterzahl wurde 10 gewählt, da dies genügend Raum für eine differenzierte Einteilung der Cluster lässt. Ausserdem ist davon auszugehen, dass bei einer noch höheren Clusterzahl die Interpretierbarkeit der einzelnen Cluster beinahe verunmöglicht wird. Die Bestimmung der beiden Grenzen mag auf den ersten Blick etwas beliebig anmuten. Doch haben einerseits zahlreiche Versuche mit den Daten gezeigt, dass damit gute Resultate erzielt werden können und andererseits rät

allgemein auch die Literatur zur Clusteranalyse eine Lösung mit relativ wenigen Cluster anzustreben. Die Performanz des X-Means-Algorithmus ist durchaus gut und wird bei grösseren Datenmenge nur durch diejenige des k-Means übertroffen.

5.2 Umsetzung

Alle präsentierten Clusteralgorithmen wurden für eine Gruppierung „normaler“ statistischer Objekte erstellt. Die zu analysierenden Daten sind jedoch Zeitreihen. Nebst dem herkömmlichen Clustering existieren noch weitere Verfahren zur Gruppierung von Zeitreihen, welche in dieser Arbeit jedoch nicht behandelt werden. Vergleiche der Verfahren haben aber gezeigt, dass mit dem Einsatz normaler Clusteralgorithmen die akkuratesten Resultate erzielt werden können. (Iglesias & Kastner, 2013)

Wie schon gesehen, besitzt ein typischer Datensatz 240 Messpunkte pro Zeitreihe. Für das Clustering wird die Zeitreihe als statistisches Objekt mit 240 Merkmalen aufgefasst, die Dimension liegt somit bei 240. Die Anzahl der Objekte ist gleich der Anzahl an CSV-Dateien, die analysiert werden sollen.

Grundsätzlich wird die Clusteranalyse für jede in den Daten vorhandene Variable separat durchgeführt. Bei beispielsweise 10 Dateien mit 175 Variablen wird also für jede Variable eine eigene Analyse durchgeführt (die in diesem Beispiel 10 Objekte enthielte). Damit ist sicherlich der Hauptzweck der Clusteranalyse in diesem Programm bereits abgedeckt, können so doch die gewünschten Muster innerhalb der Daten gefunden werden. Um jedoch noch einen Schritt weiter zu gehen, wurde auch die Möglichkeit geschaffen, mehrdimensionale Zeitreihen zu clustern. Zum Verständnis ist es nun sehr wichtig, auf die Begrifflichkeiten acht zu geben. Eine mehrdimensionale Zeitreihe ist eine Zeitreihe, bei der jedem Zeitpunkt mehrere Messwerte zugeordnet werden. Eine einzelne, 1-dimensionale Zeitreihe wird jedoch aus Sicht des Clusteralgorithmus bereits als 240-dimensionales Objekt betrachtet. Eine Möglichkeit wäre, das Objekt weiter aufzublähen, bei einer 2-dimensionalen Zeitreihe würde dann ein Objekt mit 480 Merkmalen vorliegen. Das hätte aber gravierende Nachteile: Mit höherer Dimension steigt die Berechnungskomplexität, die Resultate werden immer ungenauer und das Ergebnis wäre schwer als Zeitreihe darzustellen (mehrere Messwerte pro Zeitpunkt). Deshalb fiel die Entscheidung auf eine vorgängige Dimensionsreduktion. Es wird für jeden Zeitpunkt der mehrdimensionalen Zeitreihe eine Reduktion der Dimension auf 1 vorgenommen und erst danach das Clustering durchgeführt. Damit bleibt die Dimension eines Objekts aus der Warte des Clusteralgorithmus bei 240. Zudem kann so das Resultat ohne weitere Transformation wieder als Zeitreihe dargestellt werden.

Ein weiteres Ziel ist die möglichst starke Automatisierung der Clusteranalyse. Ein erster Teil dieser Automatisierung wurde gerade beschrieben und besteht aus der Vorauswahl und Konfiguration eines geeigneten Algorithmus. Weiter wäre es jedoch wünschenswert, wenn dem Benutzer zusätzlich nur diejenigen Clusterlösungen präsentiert würden, die auch eindeutig und relevant sind. Sowohl die Eindeutigkeit als auch die Relevanz einer Lösung sind jedoch sehr subjektive Attribute. Es existieren Metriken zur Messung der Qualität einer Clusterlösung, die meist angeben, wie stark sich die Objekte eines Clusters ähneln und wie stark sich die verschiedenen Cluster unterscheiden. Das Resultat ist aber lediglich eine Zahl. Diese ist nützlich, um beispielsweise zu vergleichen welcher Clusteralgorithmus welches Ergebnis erzielt. Die Entscheidung, ob eine Lösung annehmbar ist oder nicht, kann aber durch eine solche Kennzahl nicht automatisiert werden. Anstatt nun willkürlich eine Grenze für eine relevante und klare Lösung festzulegen, wurde entschieden, den Benutzer diese Grenze individuell setzen zu lassen. Im Programm besteht also die Möglichkeit, ein Kriterium auszuwählen und auch den Wert der Kennzahl festzulegen, den eine Clusterlösung mindestens erreichen muss (oder nicht überschreiten darf), um als „gutes Resultat“ zu gelten. Es wurden, angelehnt an die Definition der Ziele einer klaren Gruppierung aus Kapitel 4, zwei Kennzahlen festgelegt. Einerseits die Varianz innerhalb der Cluster[1] und die Varianz zwischen den Clustern[2], berechnet als Varianz zwischen den Clusterschwerpunkten. In Weka fehlen solche Kennzahlen leider gänzlich. In Apache Common Maths, einem Framework für statistische Analysen, existiert jedoch die Möglichkeit zur Bestimmung der Varianzen. Da für die Berechnung der Varianz zwischen den Clustern die Clusterschwerpunkte benötigt werden, kann dieses Kriterium nur in Verbindung mit Clusteralgorithmen verwendet werden, welche in der Analyse mit Schwerpunkten arbeiten, also der X-Means- und der k-Means-Algorithmus.

Die Durchführung einer Clusteranalyse beinhaltet mehr, als lediglich die Anwendung eines Clustering-Algorithmus auf einen Datensatz. Von zentraler Bedeutung ist vor allem die Aufbereitung des Datenmaterials in Form einer Standardisierung, die mit Hilfe von Weka durchgeführt werden kann. Auch die Wahl des Distanzmasses beeinflusst das Ergebnis stark. Verwendet wird durchgehend das Euklidische Distanzmass, da dies mitunter gerade bei Zeitreihen die beste Wahl unter den Minkowsi-Metriken darstellt (Iglesias & Kastner, 2013). Die Evaluierung der Cluster und eventuelle Anpassungen zwischen den Iterationen (beispielsweise des X-Means) übernimmt Weka automatisch.

[1] Die Varianz innerhalb eines Clusters repräsentiert die Intracluster-Homogenität und sollte möglichst klein sein

[2] Die Varianz zwischen den Clustern repräsentiert die Intercluster-Heterogenität und sollte möglichst gross sein

5.3 Visualisierung

Das Ergebnis der Clusteranalyse ist eine Liste von Clustern mit den dazugehörenden Datenobjekten. Nun stellt sich die Frage, wie die Cluster bestmöglich visualisiert werden können. Am sinnvollsten erscheint es, die Zeitreihen als solche darzustellen und alle zum selben Cluster gehörenden auch gleich einzufärben. Eine Visualisierung mittels Streudiagramm, der am häufigsten verwendeten Art der Darstellung von Ergebnissen einer Clusteranalyse, ist ohne weitere Transformation der Daten nicht realisierbar (Wiedenbeck & Züll, 2001). Als einzige Möglichkeit dies doch zu bewerkstelligen, bleibt wiederum eine Dimensionsreduktion. Konkret wird die komplette Zeitreihe in den 2-dimensionalen Raum projiziert. Verwendet wird dazu, wie auch für die übrigen Dimensionsreduktionen, die Hauptkomponentenanalyse.

6. Architektur

6.1 GUI-Architektur

Ein wichtiger Bestandteil eines Architekturkonzepts ist die Planung des Zusammenspiels von grafischer Benutzeroberfläche (oder graphical user interface, kurz GUI) mit Daten und Logik. Ein in diesem Zusammenhang oft verwendetes Design Pattern ist das Model-View-Controller (MVC) Entwurfsmuster. MVC ist eine Kombination der 3 bekannten Pattern Strategy, Observer und Composite, ein sogenanntes Compound Pattern. Wie jedoch von Freeman und Freeman (2004) betont, handelt es sich dabei nicht einfach um einen Einsatz der drei Muster im selben Programmteil, sondern wird durch die geschickte Verflechtung ein neues Pattern erzeugt, welches mehr leistet, als die Summe der drei Einzelpatterns. Ziel des MVC ist eine möglichst klare Trennung zwischen GUI und Logik. Wie der Name bereits erahnen lässt, besteht MVC aus drei Teilen, die je eine Funktion erfüllen sollen: In der View wird die ganze Benutzeroberfläche definiert und die Interkation mit dem Benutzer gehandhabt, der Controller entscheidet wie auf Benutzereingaben regiert werden soll und leitet sie gegebenenfalls an das Model weiter, welches die Daten bereitstellt bzw. die Programmlogik (z.B. Berechnungen) beherbergt. (Freeman & Freeman, 2004)

Daraus ergibt sich auch der Einsatz der drei Patterns. Die View registriert sich als Observer beim Model und wird so bei Änderung der Daten automatisch benachrichtigt. Der Controller fungiert als Strategy der View, welcher situationsbedingt ausgewechselt werden kann. Das Composite Pattern wird von der View implementiert, da sich einzelne Anzeigeobjekte oftmals ineinander verschachteln lassen; so beinhaltet ein Fenster beispielsweise eine Liste, die wiederum aus Buttons und Eingabefeldern besteht. (Freeman & Freeman, 2004)

MVC weist neben vielen Vorzügen aber auch gewisse Schwächen auf, so ist die Testbarkeit einzelner Komponenten nur sehr schwer realisierbar (vor allem, ob eine gewünschte Änderung der Daten aus Sicht der View auch wirklich durchgeführt wurde). Ende der 1990er Jahre wurde erstmals ein neues, von MVC abgeleitetes, Pattern mit dem Namen Model-View-Presenter von Mike Potel (1996), damals bei Taligent Inc. beschäftigt, vorgestellt, welches die Schwäche der mangelnden Testbarkeit etwas mildern sollte. Der Controller wird durch den

Presenter ausgetauscht. Die View ist immer noch Observer des Models, jedoch übernimmt der Presenter den grössten Teil der Synchronisation. Zudem besitzen das Model und die View keine gegenseitige Verbindung. (Potel, 1996)

Der Verantwortungsbereich der einzelnen Komponenten wurde in der Praxis aber oft sehr unterschiedlich interpretiert, was Martin Fowler, ein Experte und Autor auf dem Gebiet der Softwarearchitektur, dazu veranlasste das MVP Muster zu präzisieren und in zwei Typen aufzuteilen (Fowler, GUI Architectures, 2006):

Supervising Controller: Die Datensynchronisierung wird so umfassend wie möglich durch die View übernommen. Dem Presenter kommt dabei nur noch eine Vermittlungsfunktion zu, lediglich bei komplexeren Vorgängen muss er mehr Aufgaben übernehmen. Die Verbindung zwischen Model und View kann durch sogenanntes Data Binding realisiert werden. Beim Data Binding werden die graphische Benutzeroberfläche und das Datenmodell direkt miteinander verbunden. Die View und das Model bleiben also synchron, egal wie und wo die Daten geändert werden.

Passive View: Wie der Name schon sagt werden von der View keine Synchronisationsaufgaben wahrgenommen und sie kann dementsprechend einfach und „dünn“ gehalten werden. Der Presenter ist für den gesamten Austausch der Daten zwischen View und Model, als auch für die Statusänderungen der View verantwortlich.

Martin Fowler erarbeitete noch ein weiteres, vielleicht in seiner Ursprungsform weniger bekanntes Pattern, das Presentation Model. An die Stelle von Controller oder Presenter tritt das Presentation Model, welches das Schlüsselelement dieses Patterns darstellt. Es stellt die für die View benötigten Daten zur Verfügung, verwaltet aber auch die Eigenschaften der Bedienelemente der View, womit es einen Schritt weiter geht als dies Controller/Presenter normalerweise tun. Das Presentation Model kann auch als eigene, unabhängige Klasse bestehen, während beim MVP der Presenter meist über ein Interface an die View gekoppelt ist. Weiter existiert für jedes Bedienelement der View im Presentation Model ein korrespondierendes Datenfeld, welches möglicherweise an ein klassisches Model gekoppelt ist. Damit ist das Presentation Model prädestiniert für den Einsatz von Data Binding. Das bedeutet aber auch, dass die Umsetzung ohne Data Binding deutlich schwieriger ist und die Integration der Technik in das verwendete Framework schon beinahe eine Voraussetzung für den Einsatz des Musters ist. (Fowler, Presentation Model, 2004)

Zusammenfassend kann gesagt werden, dass sich alle drei – MVC, MVP und Presentation Model – relativ ähnlich sind, wenn auch mit spezifischen Stärken und Schwächen in gewissen Anwendungsfällen. Nun stellt sich natürlich die Frage, welche Implikationen die beschriebenen Architekturmodelle in Bezug auf das zu entwickelnde Programm nach sich ziehen. Wie bereits in der Beschreibung der Entwurfsmuster angeklungen, kann eine schlüssige Antwort

nur unter Berücksichtigung der verwendeten Technologien und Sprachen gegeben werden. Java als mächtige und objektorientierte Sprache stellt dabei keine Einschränkungen dar. Es ist ohne weiteres möglich, alle vorgestellten Design Patterns zu implementieren. Bleibt die Betrachtung des eingesetzten Frameworks für die grafische Benutzeroberfläche. Die Wahl fiel dabei auf JavaFX.

JavaFX wird von Oracle entwickelt und soll zum Standard bei der Entwicklung von Java Client Applikationen werden. Seit Version 2.2 ist es zudem in der Java Standard Edition (ab Version 7 Update 6) enthalten (Oracle, n.d.). Damit ist es auch für alle drei grossen Desktop Plattformen (Windows, Mac OS und Linux) verfügbar. Enthalten ist eine Vielzahl von Grafik- und Multimediaklassen. Da es eine Java Klassenbibliothek ist, steht die ganze Fülle der Java API zur Verfügung. Neuere Technologien, wie das erwähnte Data Binding, werden standardmässig unterstützt. Zudem ist es möglich, die Benutzeroberfläche vollkommen deklarativ mit einer XML-ähnlichen Sprache, genannt FXML, zu definieren. Alle Bedienelemente können dabei aber auch weiterhin über den Programmcode verändert, entfernt oder neu erstellt werden. Die grafischen Eigenschaften der Anzeige können sehr elegant mit CSS Stylesheets definiert werden, womit ein einheitliches grafisches Design enorm erleichtert wird. (Pawlan, 2013)

Mit JavaFX schliesst Oracle zu den Entwicklungswerkzeugen für die .NET Frameworks WPF, Silverlight oder Windows RT auf, welche die meisten Neuerungen bereits seit einigen Jahren anbieten. Gegenüber momentan noch weiter verbreiteten Frameworks wie Swing oder SWT, bietet JavaFX klare Vorteile in Bezug auf Trennung von Logik und View und ist von der Entwicklungsphilosophie besser an moderne Standards angepasst. In den kommenden Jahren ist sogar eine Portierung für die mobilen Plattformen mit ARM Prozessor[1] geplant (Vos, 2014). Auch wenn dies für diese Arbeit an sich kein Kriterium ist, so zeigt es doch, zusammen mit der grossen Unterstützung von Oracle, dass JavaFX eine zukunftssichere Technologie zu sein scheint. Obwohl dafür natürlich keine Garantie gegeben werden kann, so deutet die momentane Zielstrebigkeit seitens Oracle, JavaFX als Standard-GUI-Framework zu etablieren, doch stark darauf hin. In Anbetracht der geplanten längeren Lebensdauer des Programms, kann dieser Umstand durchaus von Bedeutung sein. Ein weiterer Punkt, der für JavaFX spricht, ist die relativ einfache Anpassung an MVC und die davon abgeleiteten Muster. Die Nachteile gegenüber etablierten Technologien sind sicherlich die faktisch nicht vorhandenen Erweiterungen von Drittanbietern sowie fehlende Hilfen und Beispiele aus der Entwicklergemeinde. Durch eine qualitativ hochwertige, von Oracle frei zur Verfügung gestellte Dokumentation (inkl. Tutorials und Beispielprogrammen), wird letzteres zumindest teilweise wieder wettgemacht.

[1] ARM Prozessoren basieren auf einer grundlegend anderen Architektur als PC- und Notebook-Prozessoren und sind durch ihren sehr niedrigen Stromverbrauch für den Einsatz in mobilen Geräten prädestiniert.

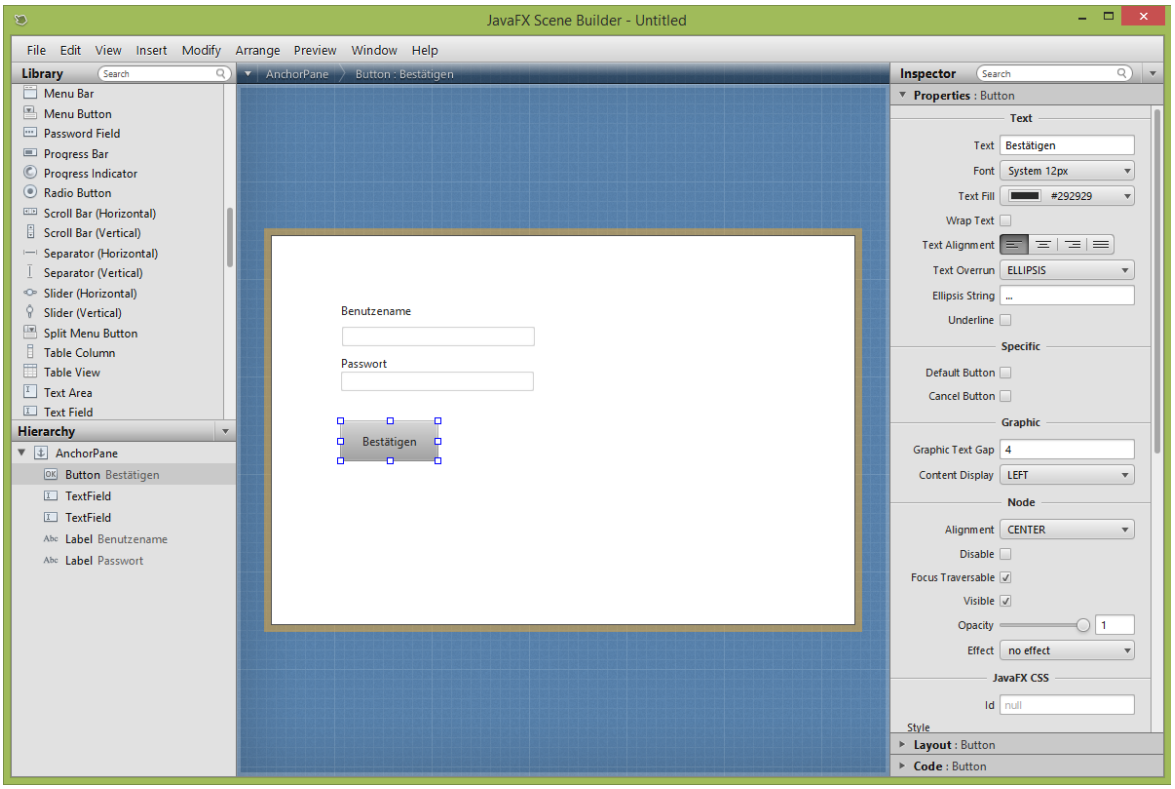


Abbildung 6.1: Beispielansicht einer Benutzeroberfläche im Scene Builder

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.paint.*?>

<AnchorPane id="AnchorPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2">
  <children>
    <Button layoutX="71.0" layoutY="190.0" mnemonicParsing="false" prefHeight="42.0" prefWidth="101.0" text="Bestätigen" />
    <TextField layoutX="71.0" layoutY="139.0" prefWidth="200.0" />
    <TextField layoutX="72.0" layoutY="93.0" prefWidth="200.0" />
    <Label layoutX="72.0" layoutY="69.0" text="Benutzename" />
    <Label layoutX="72.0" layoutY="123.0" text="Passwort" />
  </children>
</AnchorPane>
```

Abbildung 6.2: Generierter FXML-Code

In einem ersten Schritt fiel der Entscheid, die View komplett, oder so weit wie irgend möglich, in FXML zu erstellen. Dazu kann der von Oracle zur Verfügung gestellte Drag & Drop Editor „JavaFX Scene Builder“ oder ein Eclipse-Plugin zur direkten Bearbeitung des FXML-Codes verwendet werden. Da der Scene Builder lediglich ein Tool zur automatisierten Erstellung von FXML-Code darstellt, ist der Übergang zwischen den beiden Bearbeitungsmöglichkeiten fliessend. Wird beispielsweise ein Button im Scene Builder erstellt, erzeugt dieser auch gleich den korrespondierenden FXML-Abschnitt in der Quelldatei.

Diese Art der Erstellung der Benutzeroberfläche bietet den Vorteil, dass die Bedienelemente und vor allem deren grafische Eigenschaften nicht fest im Programmcode verankert werden

müssen und klar von der restlichen Programmlogik getrennt sind. Die einzelnen Eigenschaften wie Grösse, Farbe und Position können über XML-Attribute festgelegt werden. Natürlich kann in FXML kein Verhalten definiert werden, es ist also nicht möglich, auf gewisse Benutzereingaben zu reagieren oder die Elemente dynamisch zur Laufzeit zu verändern. Zu diesem Zweck kann jeder FXML-Datei eine in Java geschriebene Controller Klasse zugeordnet werden, die genau solche dynamischen Ereignisse verarbeitet. Die deklarativ erstellten Bedienelemente werden als private Klassenvariablen in die Controllerklasse eingebunden. Damit lassen sich alle Eigenschaften ändern und verwalten und auch neue hinzufügen, die in FXML gar nicht möglich wären. Beispiele dafür sind „Change Listener“ (also Methoden die ausgeführt werden, wenn sich eine bestimmte Eigenschaft des Elementes verändert) oder Überprüfung und Verarbeitung der Benutzereingaben.

Die Ähnlichkeit zwischen der Funktionsweise der Benutzeroberflächenerstellung in JavaFX und den zuvor vorgestellten Entwurfsmustern ist unschwer zu erkennen und erleichtert deren Einsatz ungemein. In der Praxis wird selten ein Pattern komplett eins zu eins nach Lehrbuch umgesetzt, vielmehr werden die Grundsätze übernommen und auf den konkreten Fall angepasst (Freeman & Freeman, 2004). Diese Vorgehensweise erwies sich auch für diese Arbeit als geeignet. Es wurde nicht versucht starr einem Muster zu folgen, sondern auch den speziellen Gegebenheiten des Frameworks und der Aufgabenstellung Rechnung zu tragen.

Als Grundlage dienten vor allem MVP und Presentation Model. Die View wird sicherlich durch die FXML-Dateien repräsentiert. Die in JavaFX „Controller“ genannte Klasse kann entweder als Presenter, Presentation Model oder gar als Code-Behind Datei der View (bezeichnet in den Microsoft Frameworks eine Datei, welche einfache Aufgaben einer deklarativ geschriebenen View übernimmt) betrachtet werden. Die Verwendung in dieser Arbeit kommt jedoch der Idee des Presentation Models sicherlich am nächsten. Die Klasse wird in der Folge auch weiterhin als Controller bezeichnet, da diese auch in JavaFX so genannt wird. Dieser Name ist generisch zu verstehen und nicht mit der Controller-Klasse des MVC gleichzusetzten. Auf eine genauere Auseinandersetzung, welche Aspekte von welchem Pattern entlehnt wurden, wird in der Folge verzichtet, da dies den Rahmen dieser Arbeit sprengen würde.

6.1.1 Konkrete Lösung

Die Verbindung zwischen der View und dem Controller ist bidirektional, das bedeutet, dass sich Controller und View „kennen“ bzw. eine Referenz auf den jeweils anderen halten. Es muss zwar nicht explizit angegeben werden, welcher View der Controller zugeordnet ist, doch müssen alle Klassenattribute im Controller, die ein Bedienelement in der View referenzieren, auch tatsächlich deklariert sein. Ein Controller könnte so theoretisch für zwei verschiedene Views eingesetzt werden, diese müssten dann aber über identische Bedienelemente verfügen.

```
public class ClusterViewController implements ViewController,Initializable{


    @FXML private TableView<ClusterModelOneDim> variableTableView;
    @FXML private TableView<IndividualClusterModel> clusterResultTable;
    @FXML private TableColumn<ClusterModelOneDim,String> variableColumn;
    @FXML private TableColumn<IndividualClusterModel,String> nameColumn;
    @FXML private TableColumn<IndividualClusterModel,Integer> countColumn;
    @FXML private TableColumn<IndividualClusterModel,Double> vColumn;
    @FXML private TableColumn<IndividualClusterModel,Double> scoreColumn;
    @FXML private ScatterChart<Number,Number> clusterChart;
    @FXML private NumberAxis xAxis;
    @FXML private NumberAxis yAxis;
    @FXML private Rectangle zoomer;
    @FXML private Button zoomButton;
    @FXML private Label totalObjects;
    @FXML private Label totalScore;
    @FXML private Label totalCluster;
    @FXML private Label totalDimension;
```

Abbildung 6.3: Beispiel-Code einer Controller-Klasse

Noch strenger ist die Verbindung seitens der View, in der die zuständige Controller-Klasse fest verankert werden muss.

Die gewählte Lösung sieht für jede View genau einen Controller vor und jeder Controller ist für eine View zuständig. Jedes Paar aus View und Controller bilden eine Einheit, die erst einmal unabhängig von den restlichen Teilen funktioniert. Diese Unabhängigkeit wird jedoch zu einem Teil zu Gunsten einer reibungsloseren Einbettung in das Gesamtprogramm aufgegeben. Während die View dabei komplett unangetastet bleibt, werden im Controller die Verbindung zu Daten und Logik hergestellt. Der Controller übernimmt drei zentrale Aufgaben:

1. Benutzerinteraktion: Dies beinhaltet die Verarbeitung und Überprüfung der Benutzereingaben und als Reaktion allfällige Änderungen der Bedienelemente.

2. Die Herstellung der Datensynchronisation zwischen View und Model. Wann immer möglich wird dabei auf das bereits beleuchtete Konzept des Data Binding zurückgegriffen.

3. Der Aufruf der entsprechenden Logikklassen für Berechnungen und Analysen.

Das Data Binding in JavaFX wird häufig mit Hilfe von Reflection umgesetzt. Dies kann am einfachsten anhand eines Beispiels illustriert werden (Abbildung 6.3). Die Variable „variableTableView“ bezeichnet eine Tabelle im GUI. Zuerst wird die Datenquelle ausgewählt, in diesem Fall eine Liste von „ClusterModelOneDim“-Objekten, die aus dem Speicher bezogen wird. Die erste Spalte der Tabelle soll den Namen der Variablen anzeigen. Zu diesem Zweck wird der Inhalt dieser Spalte an das Klassenattribut „variable“ aus der „ClusterModelOneDim“-Klasse gebunden.

Ein Nachteil dieser Methode ist sicherlich, dass bei einer Umbenennung des Attributes in z.B. „variableName“, der Code nicht mehr funktioniert, da die Bindung wirklich mit dem Namen und nicht der Variablen selbst verknüpft ist. Auch die Vorteile der statischen Typenprüfung entfallen, da nicht überprüft werden kann, ob die Variable tatsächlich vom Typ „String“ ist. Das bedeutet auch, dass allfällige Fehler erst zur Laufzeit und nicht schon zur Kompilierzeit erkannt werden können. Durch sorgfältige Programmierung und ausgiebiges Testen, kann diese Gefahr jedoch minimiert werden. Allgemein überwiegen nach Meinung des Autors die positiven Aspekte, denn die Synchronisation wird vereinfacht und der Synchronisationsaufwand stark reduziert. Die GUI-Elemente werden unabhängig davon aktualisiert, wann und wo die gebundenen Daten verändert werden. Zudem können die unterliegenden Daten jederzeit und einfach ausgetauscht werden, dazu muss lediglich ein anderes Model verwendet werden. Da der Controller das Data Binding vornimmt, sind View und Model zudem vollkommen entkoppelt. Im Vergleich zum Presentation Model Muster, wo meist die Bindings direkt in der View festgelegt werden, erhöht dies zwar etwas die Komplexität und den Verantwortungsbereich des Controllers (oder eben des Presentation Models), ermöglicht jedoch die komplette Unabhängigkeit von View und Datenmodell. Um zwischen den einzelnen Views/Controllers Daten hin und her zu transportieren, implementieren alle Controller ein gemeinsames Interface, das aus einer Methode zur Übergabe eines Parameters des Typs „Object“ besteht. Der Controller wandelt den Parameter dann in den erwarteten Typ um. Im Regelfall beziehen die Controller die Daten aus einer Model-Klasse, die Funktionalität der direkten Parameterübergabe wird ausschliesslich verwendet, wenn die Einfachheit der Datenstruktur eine separate Erstellung eines Models nicht rechtfertig, so z.B. bei einem einfachen String oder einer Liste von Zahlen.

6.2 Gesamtstruktur

Generell wurde die Architektur sehr modular aufgebaut, so dass verschiedene Programmteile komplett unabhängig voneinander gehandhabt werden können. So lassen sich ganze Berechnungsteile austauschen oder hinzufügen, ohne dass andere davon tangiert würden.

Das Programm umfasst mehr als 9700 Codezeilen verteilt auf 122 Klassen und 14 Packages, dazu kommen ca. 1300 Zeilen FXML-Code in 28 View-Dateien. Während der Entwicklung wurde darauf geachtet, dass der Code stets auf einem qualitativ hohen Niveau bleibt. Dazu wurde Sonar Qube, ein statisches Code-Analysetool, verwendet. Die sogenannte „rules compliance“ lag zum Ende der Entwicklung dann auch bei über 95%. Allerdings darf man sich von solchen Metriken nicht zu stark blenden lassen, denn überprüft wird lediglich wie gut gewisse Programmierregeln eingehalten wurden und sagen wenig bis gar nichts über die Tauglichkeit eines Programms aus der Sicht eines Benutzers aus.

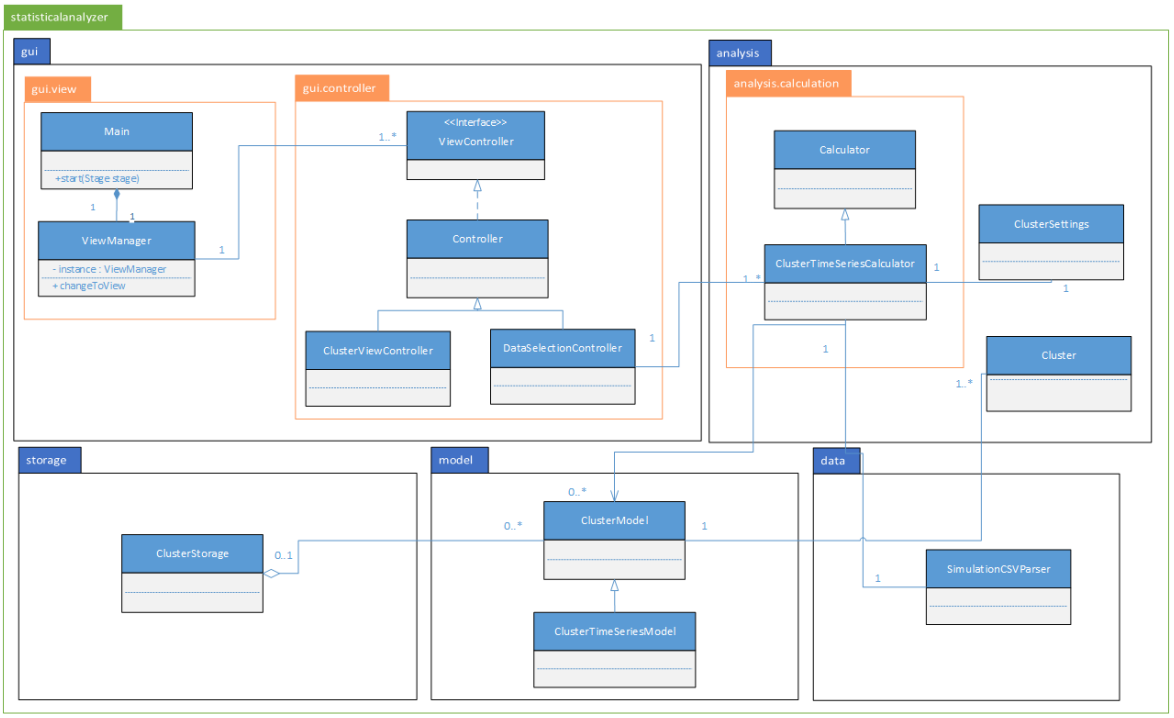


Abbildung 6.4: Grobübersicht der Architektur

Aufgrund der Grösse kann nicht die komplette Programm-Architektur im Detail besprochen werden. Um die Zusammenhänge innerhalb der Packages und Klassen dennoch erläutern zu können, wird exemplarisch der Vorgang zur Durchführung einer Clusteranalyse aus technischer Sicht beschrieben. Andere Analysen verlaufen aus architektonischer Sicht ähnlich, die konkrete Berechnung unterscheidet sich dann aber natürlich stark. Abbildung 6.4 zeigt ein stark vereinfachtes Klassendiagramm (auf Angabe von Attributen und Methoden wurde verzichtet) der wichtigsten an einer Clusteranalyse beteiligten Klassen. Dieses beinhaltet jedoch keineswegs alle Klassen. Das Fehlen einer View-Klasse erklärt sich damit, dass diese nicht als Klasse erstellt, sondern wie im Abschnitt über die GUI-Architektur dargelegt, in FXML deklariert wird.

Gestartet wird die Applikation aus der Main-Klasse, die von der JavaFX Startklasse „Application“ erbt. Danach übernimmt der „ViewManager“ die Erstellung der Views aus dem FXML-Code, das Navigieren zwischen den Views und das Laden der entsprechenden Controller-Klassen. Die Festlegung der Einstellungen und die Auswahl von Dateien und Variablen übernimmt der „DataSelectionClusterController“. Dieser ist es auch, der die Berechnungsobjekte erstellt. Konkret sind das Objekte der Klasse „ClusterTimeSeriesCalculator“, denen ausserdem noch spezifische Einstellungen für die Clusteranalyse mitgegeben werden (Klasse „ClusterSettings“). Der Calculator veranlasst das Parsing der Daten aus den CSV-Dateien und delegiert die Analyse an die neu erstellten „ClusterModel“-Objekte. Das ClusterModel

führt die eigentliche Clusteranalyse durch (inklusive Standardisierung der Daten und Evaluierung der Clusterlösung) und erstellt für jedes gefundene Cluster ein Objekt der Klasse „Cluster“, in dem die gruppierten Zeitreihen festgehalten werden. Die „ClusterModel“-Objekte werden im threadsicheren Objekt (Singleton) der Klasse „ClusterStorage“ in den Hauptspeicher geschrieben. Die Kontrolle geht nun auf den „ClusterViewController“ über, der für die grafische Darstellung der Resultate verantwortlich ist. Im Anhang ist zusätzlich noch ein Sequenzdiagramm des gerade geschilderten Vorgangs zu finden.

7. Implementierung

Der Beschrieb der konkreten Implementierung dient dazu, einen Einblick in die Funktionalität des Programms zu geben. Aufgrund des Umfangs des Programms kann auf Details nicht eingegangen werden. Zusätzlich ist jedoch eine Bedienungsanleitung im Anhang zu finden (Anhang C).

7.1 Modi

Das Programm besitzt vier grundlegende Modi zur Datenanalyse, die sich alle im Aufbau ähneln, um dem Benutzer einen einheitlichen Umgang mit dem Programm zu ermöglichen. Diese vier Programmteile sollen nun kurz beschrieben werden.

7.1.1 Variablenvergleich

Im Modus Variablenvergleich stehen primär die in Kapitel 3 beschriebenen statistischen Verfahren und Kennzahlen im Fokus. Wie in allen Modi legt der Benutzer zuerst die Einstellungen für die durchzuführenden Analysen fest. Immer können dabei die CSV-Dateien bestimmt werden, die in die Analyse miteinbezogen werden sollen. Wie der Name schon andeutet, geht es beim Modus Variablenvergleich darum, statistische Kennzahlen verschiedener statistischer Variablen miteinander zu vergleichen. Sowohl die Auswahl der Variablen als auch der Kennzahlen können in den Einstellungen festgelegt werden. Um zu verhindern, dass vor jeder Analyse immer wieder dieselben Schritte vom Benutzer vollzogen werden müssen, sind Standardeinstellungen festgelegt, bei denen alle Dateien, Variablen und Kennzahlen automatisch ausgewählt werden. Zusätzlich wird auch die Möglichkeit der Zeitreihenvorhersage (siehe Kapitel 3) bereitgestellt, allerdings ist eine Durchführung aufgrund der sehr hohen Berechnungskomplexität und dem damit verbundenen Zeitaufwand nur zu empfehlen, wenn wenige Variablen analysiert werden sollen. Nach der Durchführung der Analyse werden die Resultate präsentiert. Diese sind einerseits tabellarisch verfügbar und werden andererseits

auch grafisch visualisiert. Zur Verfügung stehen ein Balkendiagramm, auf dem alle Kennzahlen dargestellt werden, ein Box Plot, ein Histogramm sowie der zeitliche Verlauf. Für das Histogramm und das Balkendiagramm werden alle Daten, also aller CSV-Dateien zusammen, miteinander dargestellt. Bei den anderen beiden Diagrammen werden jeweils nur die Daten einer Datei angezeigt (natürlich mit Auswahlmöglichkeit). In allen Diagrammen (ausser dem Histogramm) können eine beliebe Anzahl an Variablen bzw. deren Daten verglichen werden, einzige Begrenzung stellt die Übersichtlichkeit dar, die mit jeder neuen Variablen etwas abnimmt. Dieser Modus ist die geeignete Wahl, um sich einen generellen Überblick über die Daten und die einzelnen Variablen zu verschaffen sowie Gemeinsamkeiten und Unterschiede zwischen den Variablen herauszufiltern.

7.1.2 Datensatzvergleich

Dieser Modus ist ähnlich aufgebaut wie der Variablenvergleich. Die Diagramme zur Betrachtung der Resultate sind exakt dieselben. Jedoch unterscheiden sich die beiden Modi im Analyseobjekt. Verglichen werden nämlich nicht verschiedene statistische Variablen, sondern unterschiedliche Datensätze, bestehend aus einer oder mehreren CSV-Dateien. Betrachtet wird dabei jeweils nur eine Variable. So wird dem Benutzer ermöglicht, Unterschiede in verschiedenen Simulationsdurchläufen festzustellen. Beispielsweise können so die Auswirkungen auf eine Variable durch Veränderung der Startbedingungen der Simulation beobachtet werden. Die Datensätze können vom Benutzer individuell zusammengestellt werden. Die statistischen Kennzahlen werden dann über einen ganzen Datensatz berechnet. Obwohl dieselben Diagramme verwendet werden, passen sich diese dem Analyseobjekt an. So werden beispielsweise im zeitlichen Verlauf alle Zeitreihen (eine CSV-Datei beinhaltet eine Zeitreihe) simultan dargestellt.

7.1.3 Szenarien

Der Szenarien-Modus ist einiges komplexer als die beiden vorhergehenden Modi und dient vor allem der Auffindung inhaltlich interessanter Phänomene. Diese Phänomene werden im Kontext des Programms als Szenarien bezeichnet. Konkret gemeint ist damit, dass sich eine oder mehrere Variablen in einem gewissen Wertebereich bewegen oder sich über gewisse Werteschranken hinaus bewegen. Dies kann dann mit einer inhaltlichen Interpretation versehen werden, beispielsweise wenn von einer gewissen Holzart eine Mindestmenge unterschritten wird, kann von einer Unterversorgung ausgegangen werden. Eine inhaltliche Interpretation setzt ein gewisses Mass an Kenntnissen über das Modell (hier das Modell des Schweizer Energieholzmarktes) voraus. Anstelle einer fest-codierten Auswahl an Szenarien zur Verfügung zu stellen, wurde deshalb ein Szenarien-Editor geschaffen. Damit kann sich der Benutzer,

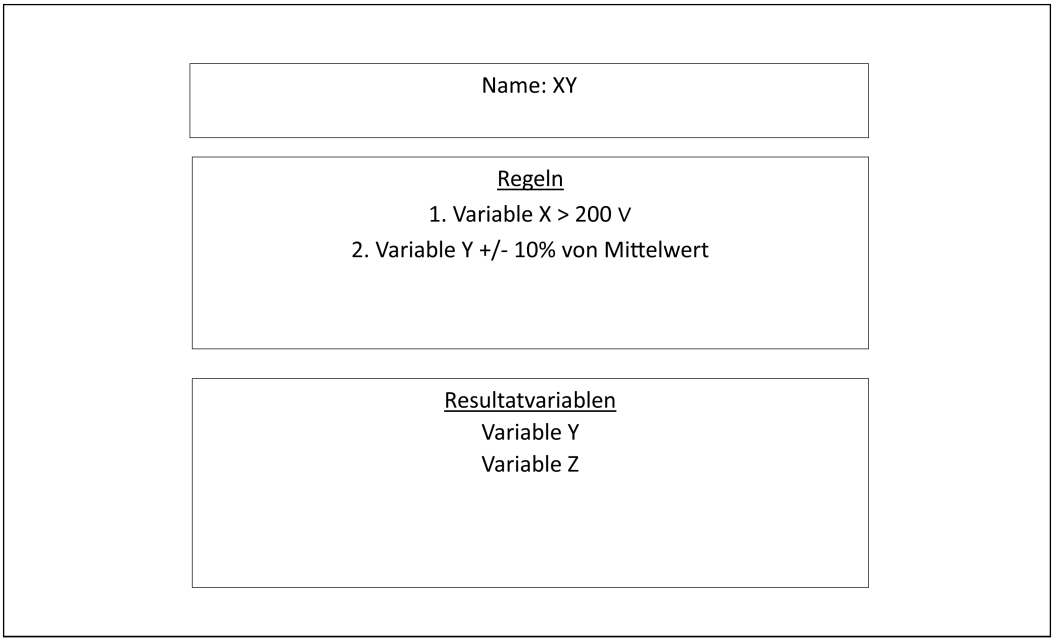


Abbildung 7.1: Schema eines Szenarios

der sich besser in der Domäne auskennt, die Szenarien individuell zusammenstellen, sichern, wieder laden und verändern.

Ein Szenario besteht aus Regeln. Jede Regel bezieht sich auf eine Variable und es können Kriterien festgelegt werden, wann die Regel angewendet werden soll. Dabei gibt es die Möglichkeit Grenzwerte festzulegen (Regel wird angewendet wenn der Wert der Variable X >200) oder auch minimale/maximale Abweichungen von einer statistischen Grösse (Regel wird angewendet wenn der Wert der Variable mehr als 10 % vom Mittelwert abweicht). Die Regeln werden mittels logischer Operatoren miteinander verknüpft. Weiter kann bei jedem Szenario festgelegt werden, welche Variablen ausgewertet werden, also welche Variablen als Resultate festgehalten werden sollen, wenn die Regeln des Szenarios Anwendung finden. Diese können dieselben oder auch andere sein, als die Variablen welche zur Bildung der Regeln verwendet werden. Die Analyse erfolgt für jede Zeile in den Daten (also für jeden Zeitpunkt) einzeln. Erst wird jede Regel zu einem Wahrheitswert evaluiert und in einem zweiten Schritt der logische Ausdruck, der durch Zusammensetzen der Wahrheitswerte mit den logischen Operatoren entsteht, ausgewertet. Falls der Ausdruck wahr (im Sinne der Logik) ist, werden alle Resultatvariablen zu dem untersuchten Zeitpunkt festgehalten, ansonsten wird die Zeile übersprungen.

Die Resultate können tabellarisch oder im zeitlichen Verlauf eingesehen werden, wobei die vom Szenario erfassten Zeitpunkte klar von den nicht erfassten abgehoben werden.

7.1.4 Clusteranalyse

Wie alle anderen Modi ist auch die Clusteranalyse zweiteilig aufgebaut. In einem ersten Schritt können dabei die Einstellungen festgelegt werden. Speziell dabei sind einerseits die Einstellungen für die Clusteralgorithmen und andererseits ein kleiner Editor für das Erstellen und Bearbeiten von statistischen Objekten. Der Objekt-Editor ermöglicht dem Benutzer, verschiedene Variablen zu Objekten zusammenzufassen. Die Einstellungen für das Clustering werden an und für sich alle automatisch vorgenommen, um den Benutzer von der Bürde einer genauen Kenntnis der Algorithmen und deren spezifischen Eigenschaften zu entbinden. Für den geübten Benutzer besteht aber selbstverständlich die Möglichkeit, die Einstellungen manuell vorzunehmen. Festgelegt werden können der Clusteralgorithmus selbst sowie je nach Algorithmus spezifische Einstellungen (beim DBSCAN können beispielsweise das ε und die Mindestzahl an Objekten pro Cluster bestimmt werden). Ausserdem können die Kriterien für die Filterung nicht relevanter Resultate festgelegt werden. Neben den bereits angesprochenen Kennzahlen der Varianzen innerhalb und zwischen den Clustern, beinhaltet dies auch eine Reduzierung auf die n besten Ergebnisse oder Lösungen mit einer Mindestanzahl an Clustern.

Die Berechnung folgt dem in Kapitel 5 vorgestellten Vorgehen. Für die Betrachtung der Ergebnisse stehen wiederum mehrere Diagramme zur Verfügung. Im Diagramm zur Darstellung des zeitlichen Verlaufs sind die Zeitreihen nach Clusterzugehörigkeit eingefärbt. Aus Gründen der Übersichtlichkeit kann auch jedes Cluster für sich betrachtet werden oder sogar bis zu vier Cluster nebeneinander (in eigenen Diagrammen). Durch eine Dimensionsreduktion können die Zeitreihen auch als Punkte in einem Streudiagramm visualisiert werden. Zudem steht natürlich auch eine tabellarische Auswertung der Clusteranalyse zur Verfügung.

7.1.5 Bemerkungen zur Berechnung und Multithreading

Bei komplexeren Berechnungen, die mehrere Sekunden bis Minuten dauern, ist es wichtig die Benutzeroberfläche nicht zu blockieren. Dies wäre der Fall, wenn die Berechnungen im selben Thread wie die Benutzerinteraktion erfolgen würden. Deshalb werden alle Berechnungen in einem separaten Hintergrundthread erledigt. Für die Modi Variablenvergleich, Datensatzvergleich und Szenario reicht dabei auch ein Thread aus. Für die Clusteranalyse wird allerdings auf Multithreading zurückgegriffen, sofern der Prozessor auf der das Programm bzw. die Java Virtual Machine ausgeführt wird, überhaupt mehr als einen Thread parallel bearbeiten kann. Die zu analysierenden statistischen Objekte werden dabei gleichmässig auf die bis zu vier Threads aufgeteilt (bei mehr als vier Threads übersteigt der Synchronisationsaufwand die durch die parallele Berechnung gewonnene Zeit zunehmend).

7.2 GUI

Bei der grafischen Benutzeroberfläche wurde darauf geachtet, eine möglichst einheitliche und intuitive Bedienung zu ermöglichen. Das Design orientiert sich eher an mobilen Applikationen mit wenigen Untermenüs und einer flachen Hierarchie. Zudem wurde auf eine Menü-Bar verzichtet, alle Funktionen sind direkt aufrufbar. Potenziell nicht sofort verständliche Bedienelemente wurden mit einem Info-Tooltip mit einer kleinen Erläuterung versehen. Je interaktiver ein Programm und je mehr Auswahlmöglichkeiten dem Benutzer gewährt werden, desto mehr Gewicht muss auf die Behandlung von Ausnahmefällen, etwa falsche oder unerwartete Benutzereingaben, gelegt werden. So wurde versucht, an allen Stellen an denen der Benutzer Entscheidungen oder eine Auswahl treffen kann, auch Codeblöcke für das Abfangen allfälliger Excpetions einzubauen. Nachfolgend werden einige Screenshots präsentiert, die sicherlich den anschaulichsten Weg zur Präsentation des entwickelten GUIs darstellen.

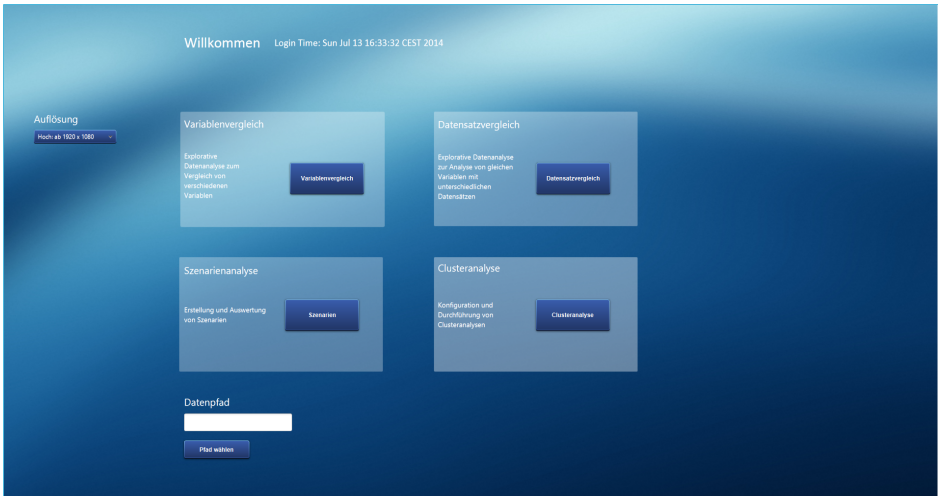


Abbildung 7.2: Start-Screen

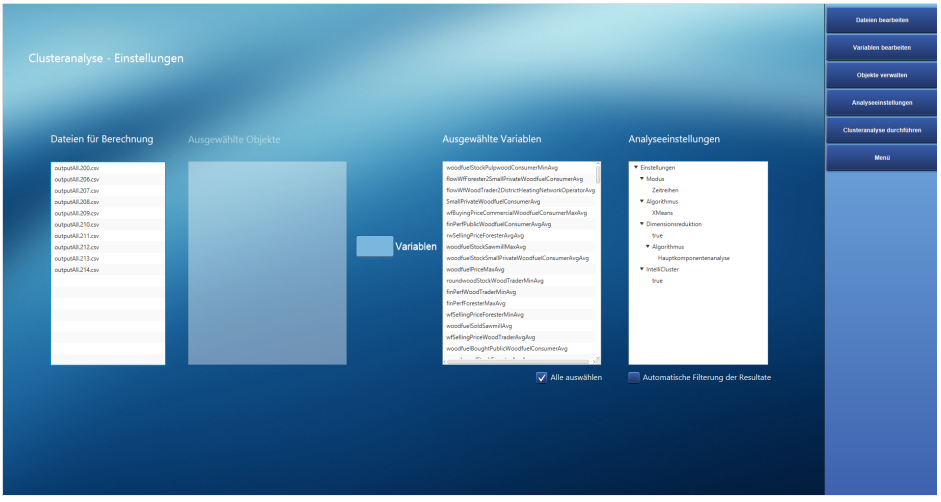


Abbildung 7.3: Einstellungen am Beispiel der Clusteranalyse

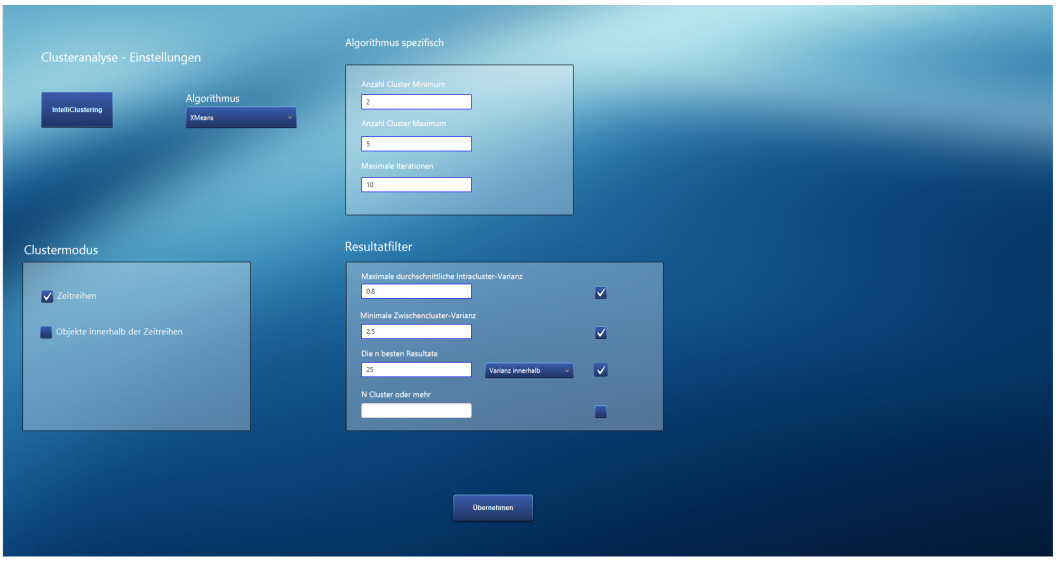


Abbildung 7.4: Einstellungen des Algorithmus und des Resultatfilters

Abbildung 7.5: Resultate am Beispiel der Clusteranalyse

7.3 Datenexport

Alle Daten die im Programm in Tabellen dargestellt werden, können nach Microsoft Excel exportiert werden. Dazu genügt ein Rechtsklick auf die entsprechende Tabelle und die Auswahl im erscheinenden Kontextmenü. Weiter können alle Diagramme sowohl als Bilddatei im PNG-Format gespeichert oder in den Zwischenspeicher zur weiteren Verwendung in einem beliebigen anderen Programm übertragen werden.

7.4 Datenspeicherung

Die meisten Daten werden dauernd im Hauptspeicher gehalten, was aufgrund der kurzen Zugriffszeiten der Performanz der Analysen zugutekommt. Daten die persistiert werden müssen, beispielsweise vom Nutzer erstelle Szenarien, werden in einer SQLite-Datenbank gespeichert, deren Schema im Anhang zu finden ist (Anhang A).

7.5 Anforderungen an die Dateien

Die Implementierung trifft Annahmen über das Format der zu analysierenden CSV-Dateien. So muss das Datum in der ersten Spalte stehen und die Variablennamen werden in der ersten Zeile erwartet. Sollte sich das Datenformat einmal ändern, zieht dies aber keine allzu grossen Änderungen im Programmcode nach sich. Es muss lediglich die für das Einlesen der Dateien zuständige Klasse geändert oder neu geschrieben werden, die Klasse „SimulationCSVParser“. Somit wäre es in Zukunft auch ohne weiteres möglich Daten aus anderen Quellen als CSV-Dateien, z.B. SQL-Datenbanken oder XML-Dateien, zu beziehen.

Wenige Probleme bereitete der Ausschluss von Variablen mit lauter „null“-Werten. Schwieriger zu behandeln war jedoch der Fall, bei dem nur in manchen Zeilen „null“ als Wert auftritt. Für einige Verfahren war es möglich, diese Werte einfach zu ignorieren, bei anderen mussten die Werte jedoch mit der Zahl 0 ersetzt werden. Die Berechnungen wurden so implementiert, dass sich für das Programm keine Probleme ergeben. Allerdings nimmt mit steigender Anzahl an „null“-Werten natürlich die Qualität des Resultates ab.

7.6 Verwendete Frameworks

7.6.1 Java FX

JavaFX wurde bereits intensiv in Kapitel 6 behandelt und dient als Grundgerüst für die grafische Benutzeroberfläche.

7.6.2 JFreeChart

JFreeChart dient zur dynamischen Erstellung von Diagrammen. Im Programm werden damit das Histogramm und der Box Plot generiert.

7.6.3 JUnit

JUnit ist ein Komponententest-Framework (sogenannte Unit-Tests) für Java-Code. Komponenten oder eben Units sind Klassen oder einzelne Methoden. Für diese Komponenten können Testklassen geschrieben werden, welche Teile des Programmcodes überprüfen. Nach jeder Änderung im Programmcode können die Tests erneut durchgeführt werden. Damit wird überprüft, ob sich mit der Änderung eventuell ein Fehler eingeschlichen hat. Zu beachten gilt es allerdings, dass diesen Tests nicht einfach blind vertraut werden darf, da vielleicht beim Schreiben des Tests an einen möglichen Ausnahmefall nicht gedacht wurde. JUnit eignet sich zudem auch eher für das Testen von Programmlogik und nicht für GUI-Tests.

7.6.4 Javadoc

Javadoc ist ein Framework zur automatischen Generierung einer Code-Dokumentation in HTML. Damit wurden alle wichtigen Klassen und Methoden im Programm dokumentiert.

7.6.5 Apache POI

Apache POI wurde für den Datenexport nach Microsoft Excel verwendet.

7.6.6 Apache Common Maths

Apache Common Maths ist eine Library für verschiedenste Berechnungen aus Mathematik und Statistik. Verwendet wurde es für alle Auswertungen statistischer Kennzahlen sowie die Berechnung der Qualität der Clusteranalyse.

7.6.7 Weka

Weka ist die zentrale Bibliothek für die Durchführung der Clusteranalysen und wurde bereits in Kapitel 5 kurz beschrieben.

7.6.8 Weitere Frameworks

Neben den gerade vorgestellten Frameworks wurden noch weitere Bibliotheken für kleinere Aufgaben verwendet. Zu nennen sind hier opencsv für die Unterstützung des Einlesens der CSV-Dateien, MDSJ für die Durchführung der Multidimensionalen Skalierung, Jama als Hilfsmittel zur Durchführung der Hauptkomponentenanalyse sowie SQLite-JDBC für die Datenbankkommunikation.

8. Diskussion der Resultate

In diesem Kapitel soll das entstandene Programm diskutiert und dessen Eignung für den Analysezweck kritisch gewürdigt werden. Am Ende des Kapitels wird anhand eines konkreten Anwendungsbeispiels in Form eines Szenarios exemplarisch gezeigt, wie mit dem Programm Wissen aus den Daten herausgefiltert werden kann.

8.1 Eignung des Programms

Ziel war es, ein Programm zur Analyse statistischer Eigenschaften (insbesondere Clusteranalyse) und sonstigen inhaltlich interessanten Phänomenen zu entwickeln. Zudem sollte das Programm möglichst gut auf die Daten angepasst werden. Um dies zu erreichen, wurden vier unabhängige Modi zur Datenanalyse geschaffen. Zwei Aspekte standen dabei im Vordergrund: Einerseits sollten die Analysen so auf den Anwendungsfall abgestimmt sein, dass sie einfacher, schneller und komfortabler zu realisieren wären, als dies mit einem Standardprogramm für statistische Analysen der Fall wäre. Andererseits sollte das Programm weitere Funktionen anbieten, die so vielleicht nicht in anderen Programmen zu finden oder viel besser automatisiert sind.

Es wurde versucht, den ersten Aspekt bei allen vier Modi umzusetzen. Ist das Programm einmal korrekt aufgesetzt, kann beispielsweise mit nur zwei Klicks in allen Modi die Analyse gestartet werden. Möglich ist dies aufgrund der vorgenommen Voreinstellungen und der sehr kurzen Pfade im Programm. Weiter können die Resultate danach sofort grafisch in verschiedenen Diagrammen betrachtet werden. Während die Modi Variablenvergleich und Datensatzvergleich noch einfache statistische Kennzahlen liefern, wird bei der Cluster- und Szenarienanalyse auch dem zweiten Aspekt Rechnung getragen.

Für den Benutzer unterscheidet sich die Handhabung des Programmteils Clusteranalyse kaum von den anderen Modi. Die zusätzlichen Schritte, die zur Durchführung einer Clusteranalyse vollzogen werden müssen, erledigt das Programm automatisch im Hintergrund. Im speziellen beinhaltet dies auch die Wahl und Konfiguration des Clusteralgorithmus. So ermöglicht

die Software auch Benutzern, die kaum Erfahrung mit Clusteranalysen besitzen, ebensolche durchzuführen und trotzdem brauchbare Resultate zu erhalten. Während der Entwicklung war die Auswahl eines geeigneten Algorithmus jedoch keine leichte Aufgabe. Normalerweise muss in der Clusteranalyse vom Benutzer viel Zeit investiert werden, verschiedene Algorithmen auszuprobieren und diese mit verschiedenen Konfigurationen laufen zu lassen. Es wurde versucht, diesen Aufwand soweit als möglich zu eliminieren. Trotzdem kann keine vollkommene Garantie darüber abgegeben werden, dass vielleicht für die eine oder andere Variable ein anderer Algorithmus eine noch bessere Lösung bringen würde. Die in Kapitel 5 bereits angesprochene Problematik mit der etwas freien Festsetzung der minimalen und maximalen Anzahl an Clustern beim X-Means-Algorithmus ist ein typischer Kompromiss, der getroffen werden musste, um überhaupt eine automatisierte Clusteranalyse ermöglichen zu können.

Das Ziel, dem Benutzer automatisch nur diejenigen Clusterlösungen zu präsentieren, die eindeutig sind, konnte nicht gänzlich umgesetzt werden. In der Literatur fehlen dazu klare und objektive Regeln, welche Clusterlösung annehmbar ist und welche nicht. Dies hängt sehr stark vom Betrachter ab und kann nur sehr schwer verallgemeinert werden. Es existiert zwar eine Reihe von Metriken, welche die Güte des Ergebnisses einer Clusteranalyse bestimmen, jedoch ist schwer festzulegen, wo die Grenze zwischen brauchbar und unbrauchbar zu finden ist. Die implementierte Lösung, bei welcher der Benutzer die Kriterien selbst festlegen kann, ermöglicht es jedoch trotzdem Clusterergebnisse vorgängig zu filtern.

Generell bietet das entwickelte Programm einen einfachen Zugang zu statistischen Analysen im Allgemeinen und der Clusteranalyse im Speziellen und darf als durchaus geeignet betrachet werden, die Aufgabenstellung zu erfassen. Gerade im Vergleich mit gängigen Statistikprogrammen, muss sich der Benutzer viel weniger um Einzelheiten kümmern. Mit dem Modus Szenario verfügt das Programm ausserdem über eine Analysemöglichkeit, mit der sehr gezielt und individuell nach inhaltlichen Phänomenen gesucht werden kann und in dieser Form in anderen Programmen schwer zu finden ist.

8.2 Eignung der Technologien

Grössere technische Probleme mit den eingesetzten Technologien und Frameworks sind keine aufgetreten. JavaFX erwies sich als gut geeignet, die Aufgabenstellung umzusetzen. Insbesondere das Vorhaben der klaren Trennung von Benutzeroberfläche und Logik liess sich sehr konsequent umsetzen. Teilweise wurden aber auch Nachteile sichtbar, die wohl der noch relativ kurzen Verfügbarkeit und Entwicklungshistorie geschuldet sind. So verfügen beispielsweise die Diagramme standardmässig über keine Zoom- und Datenexportfunktion. Dies musste alles selbst programmiert werden, während z.B. im auf Swing basierenden JFreeChart solche Funktionen längst zur Grundausstattung gehören. Einige der Diagramme wurden dann auch

mit JFreeChart umgesetzt (namentlich das Histogramm und der Box Plot), bei vielen Diagrammtypen entstanden jedoch Probleme und Konflikte zwischen JavaFX und dem in einem speziellen Swing-Panel eingebetteten JFreeChart-Diagramm. Alle übrigen Diagramme wurden wieder mit JavaFX erstellt. Im fertigen Programm entstehen teilweise beim Wechsel zwischen einem JFreeChart und einem JavaFX-Diagramm kleinere Grafikfehler, die jedoch bei einer Benutzeraktion sofort wieder verschwinden.

Weka und Apache Commons Maths, die wichtigsten Frameworks für die Analysen im Programm, sind beide gut in ein eigenes Programm einzubinden und stellten keine Probleme dar. An manchen Stellen bemerkte man jedoch, dass Weka nicht hauptsächlich auf Clusteranalysen ausgerichtet ist und bietet vor allem in Hinblick auf die Evaluierung der Clusterergebnisse nur wenige Möglichkeiten und Metriken. Zwei Alternativen wären einerseits die Statistikprogrammiersprache R oder das Java-Framework ELKI. Bei R muss jedoch mit einem Zusatzaufwand zur Einbindung in ein eigenes Java-Tool gerechnet werden, zudem muss natürlich auch die Sprache beherrscht werden. ELKI bietet eine grössere Auswahl an Clusteralgorithmen und auch Möglichkeiten zur Evaluierung der Ergebnisse, liegt jedoch zum Zeitpunkt des Schreibens dieser Arbeit nicht in einer stabilen Version vor. Auf der Projektwebsite wird sogar explizit davon abgeraten, ELKI in seinem momentanen Zustand in ein eigenes Programm einzubinden.

Wird die Menge an Daten grösser, kann es vorkommen, dass der Java Heap Space nicht mehr ausreicht (hängt natürlich von dessen eingestellter Grösse ab). Vor allem Weka erwies sich als relativ speicherhungrig. Werden beispielsweise mehr als 100 CSV-Dateien in die Analyse miteinbezogen und sollen alle 175 Variablen gleichzeitig untersucht werden, kann es nötig sein, den Java Heap Space auf mehr als 256 Megabyte zu erhöhen.

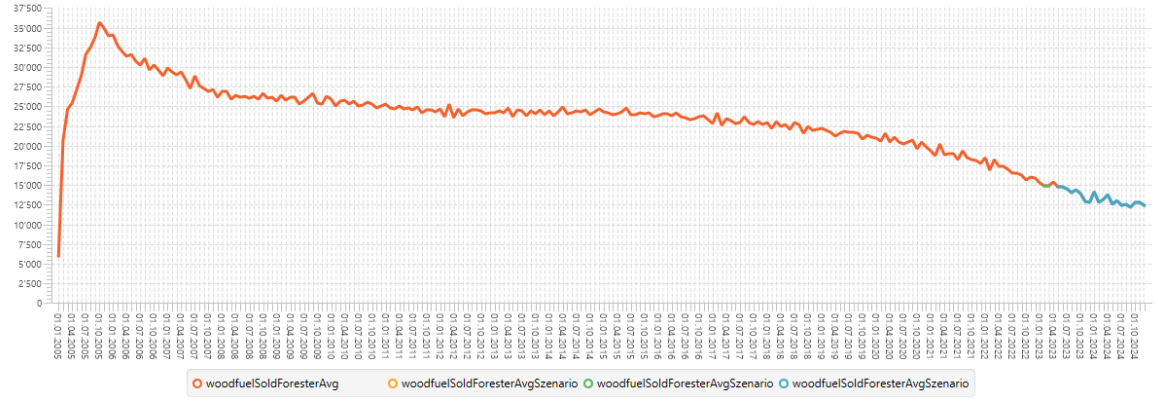
8.3 Anwendungsbeispiel

An einem kleinen Anwendungsbeispiel eines Szenarios soll aufgezeigt werden, wie mit dem entwickelten Programm Wissen aus den Daten extrahiert werden kann. Im Modell des Schweizer Energieholzmarktes sind die Förster unter anderem für die Holzernte zuständig und verkaufen Energieholz direkt oder indirekt (via einen Holzhändler) an andere Marktteilnehmer. Dies sind beispielsweise Unternehmen (Variablenname: Commercial Woodfuel Consumer) und Gemeinden (Public Woodfuel Consumer), die ihre Immobilien mit Holz heizen. Untersucht werden soll, wie sich eine Drosselung der Verkaufsmenge von Energieholz seitens der Förster auf die Lagermengen der anderen Akteure und den Preis von Energieholz auswirkt. Die Fragestellung ist natürlich trivial, es ist klarerweise zu erwarten, dass mit einer Reduktion der Verkaufsmenge auch die Lagerbestände sinken werden. Und wie die ökonomische Theorie von Angebot und Nachfrage lehrt, steigt der Preis eines Gutes bei der Verringerung des

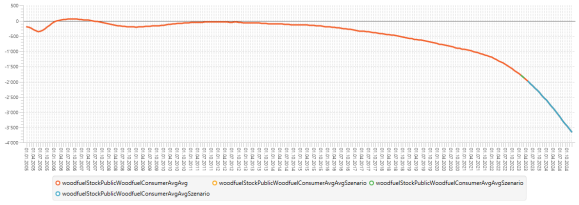
Angebots. Ziel dieses Beispiels ist es jedoch nur zu zeigen, wie so eine Analyse im Programm aussehen kann.

Zuerst muss das Szenario erstellt werden, als Ergebnisvariablen werden der Preis von Energieholz, die Verkaufsmenge der Förster und die Lagerbestände der Gemeinden und Unternehmen ausgewählt. Als einzige Regel wird eine Unterschreitung der Marke von 15000 Einheiten der Verkaufsmenge definiert. Um solch einen absoluten Betrag festsetzten zu können, bedarf es natürlich einer gewissen Kenntnis des Wertebereichs der Variablen. Alternativ hätte auch eine grosse negative Abweichung vom Mittelwert als Regel definiert werden können.

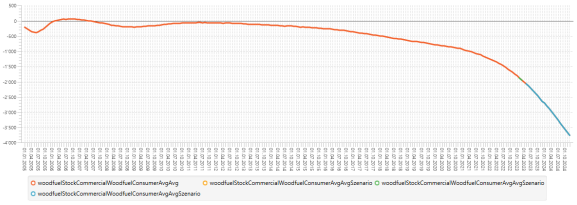
Danach kann die Analyse gestartet werden. In Abbildung 8.1a ist der Verlauf der Verkaufsmenge des Energieholzes zu sehen. Jeder zusammenhängende und vom Szenario erfasste Datumsbereich wird mit einer neuen Farbe markiert (auf den Abbildungen ist vor allem der blaue Bereich gut zu erkennen). Zu sehen ist eine kontinuierliche Reduktion der Verkaufsmenge. Die Lagerbestände der Gemeinden und der Unternehmen sinken erwartungsgemäss (Abbildungen 8.1b und 8.1c), während der Preis für Energieholz stark steigt (Abbildung 8.1d). Wie bereits erklärt, ist dies ein sehr einfaches Beispiel, durch eine ausgeklügelte Bildung und auch logische Verknüpfung von Regeln, können jedoch auch viel komplexere Phänomene entdeckt werden. Allerdings benötigt man dafür auch ein vertieftes Verständnis des Modells einerseits und der Daten andererseits.



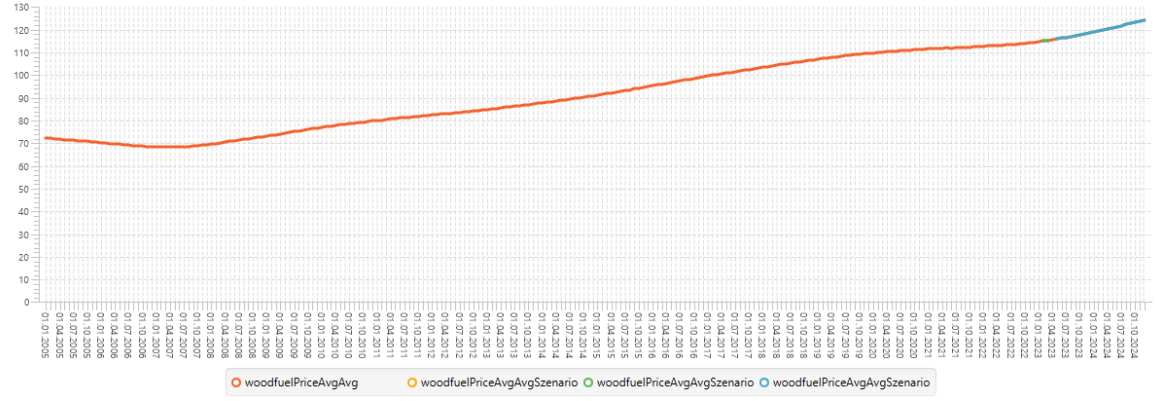
(A) Verkaufsmenge von Energieholz der Förster



(B) Lagermenge Gemeinden



(C) Lagermenge Unternehmen



(D) Preis von Energieholz

ABBILDUNG 8.1: Anschauungsbeispiel Szenario

9. Ausblick

Wie bei jeder geschriebenen Software ist es immer möglich, den Umfang zu vergrössern. Im Programm, das im Rahmen dieser Arbeit entstanden ist, könnten in den beiden Modi Variablen- und Datensatzvergleich die Anzahl an statistischen Kennzahlen und Auswertungsdiagrammen noch weiter erhöht werden. Auch der Szenarien-Editor, bzw. die Komplexität der Regeln eines Szenarios, könnten noch weiter ausgebaut werden. So werden momentan beispielsweise keine Klammern in den logischen Ausdrücken unterstützt oder es könnten noch weitere Typen von Regeln geschaffen werden, eventuell sogar in Verknüpfung mit der Clusteranalyse. Eine Erweiterung oder Verbesserung der Clusteranalyse sollte aber in einem ersten Schritt nicht unbedingt auf die Implementierung zielen, sondern eher auf eine Erarbeitung formaler Kriterien für die Eignung einer Clusterlösung. Ziel könnte beispielsweise sein, ein objektives Kriterium zu entwerfen, wann das Clusterergebnis dem Forschungsziel des Benutzers nützt oder je nach Anwendungsziel unterschiedliche Kriterien zu verwenden.

Die Gruppierung von Zeitreihen wird in der Arbeit mittels „normaler“ Clusteranalyse durchgeführt. Eine Erweiterung könnte sein, andere Verfahren aus der Zeitreihenanalyse mit der herkömmlichen Clusteranalyse zu verknüpfen. Konkret könnten die herkömmlichen Distanzmasse durch signalanalytische Verfahren wie zum Beispiel Dynamic-Time-Warping ersetzt werden, das auch zeitlich verschobene Zeitreihen noch als ähnlich erkennt. Ideen dazu sind beispielsweise im Artikel von Iglesias & Kastner (2013) zu finden.

Das Programm unterstützt momentan lediglich metrische statistische Variablen und könnte auch für die Verwendung von nominalen und ordinalen Variablen erweitert werden. Allerdings wäre das natürlich nur sinnvoll, wenn das Simulationsprogramm in Zukunft auch solche Variablen als Output generieren würde.

Wie in der Arbeit beschrieben, unterstützt die Clusteranalyse das Finden von bisher unbekannten Mustern in den Daten. In Zukunft könnte aber auch der entgegengesetzte Weg interessant werden, nämlich die Zuordnung von neuen Zeitreihen zu vorgegebenen Klassen. Diese Klassen könnten mit Hilfe der Clusteranalyse definiert werden. Für die Zuordnung zu den Klassen können Klassifikationsverfahren verwendet werden, von denen verschiedenste auch in Weka integriert sind.

A. Ergänzungen zur Implementierung

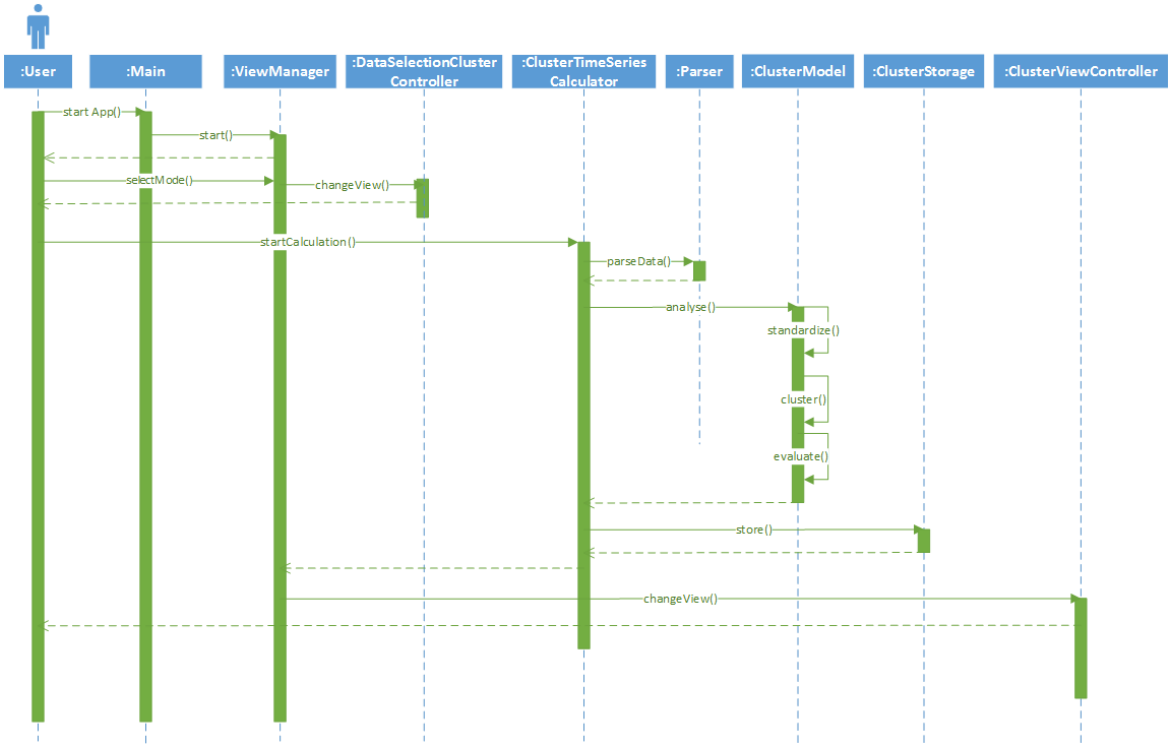


Abbildung A.1: Sequenzdiagramm der Durchführung einer Clusteranalyse aus technischer Sicht

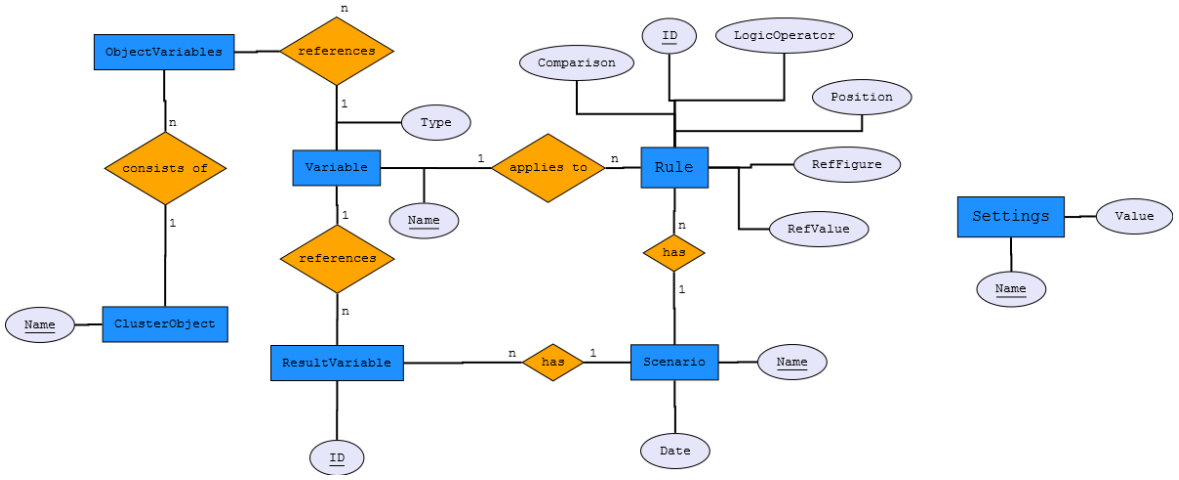


Abbildung A.2: Datenbankschema als ER-Diagramm

B. Systemvoraussetzungen

B.1 Installation

Das Programm kann auf zwei Arten verwendet werden:

- Als JAR-File ohne vorgängige Installation. Java muss mindestens in Version 8 (JRE 8) auf dem Computer installiert sein. Zudem muss sich der Ordner „libs“ im selben Verzeichnis befinden wie das JAR-File.
- Mittels Installation des mitgelieferten Installers. JRE wird, falls nicht vorhanden, mitinstalliert.

Es wird empfohlen das Programm zu installieren, da so auch sicher alle integrierten Frameworks korrekt laufen.

Die Voraussetzungen richten sich nach denen des Java Runtime Environment 8, die je nach Betriebssystem variieren.

Generell ist das Programm auf allen Betriebssystemen lauffähig, auf denen das JRE 8 installiert werden kann. Für das Betriebssystem Windows (ab Vista) wird ein Installer mitgeliefert.

Getestet wurde das Programm nur auf den Betriebssystemen Windows 7 und Windows 8.1.

Das Programm ist für drei Auflösungseinstellungen optimiert: 1920 x 1080 (Full-HD), 1600 x 900 und 1024 x 768.

Werden 100 CSV-Dateien oder mehr für eine Analyse mit **allen** Variablen verwendet, so muss der Java Heap Space vergrössert werden. Empfohlen wird generell eine Mindestgrösse von 512 MB.

B.2 Inhalt der CD

Die CD beinhaltet folgendes:

- Arbeit im PDF-Format
- Zusammenfassung als Textdatei
- Abstract als Textdatei
- Programm als JAR-File
- Installer für Windows
- Javadoc

C. Bedienungsanleitung

C.1 Start

C.1.1 Start-Screen

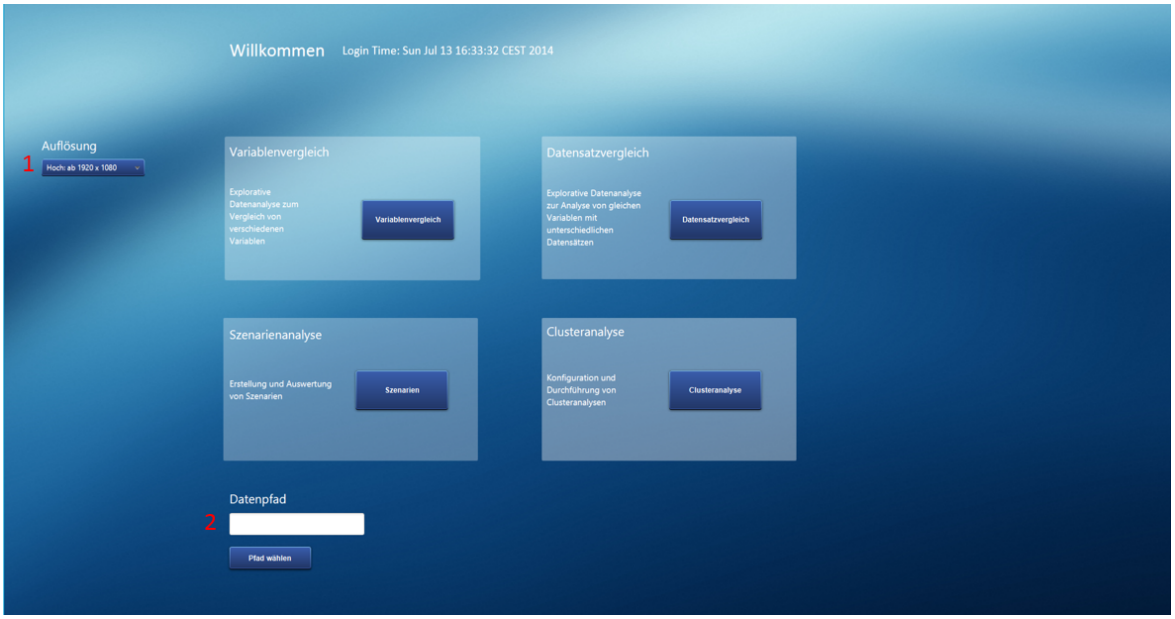


Abbildung C.1: Start-Screen

Der Startbildschirm ist der Einstiegspunkt ins Programm, von wo in einen der vier Berechnungsmodi gewechselt werden kann.

1 - Auswahl der Bildschirmauflösung.
2 - Auswahl des Datenpfades, von wo die CSV-Dateien geladen werden sollen.

C.2 Auswahl

C.2.1 Dateiauswahl

In der Dateiauswahl wird festgelegt, welche Dateien aus dem Datenordner für die Analyse ausgewählt werden.

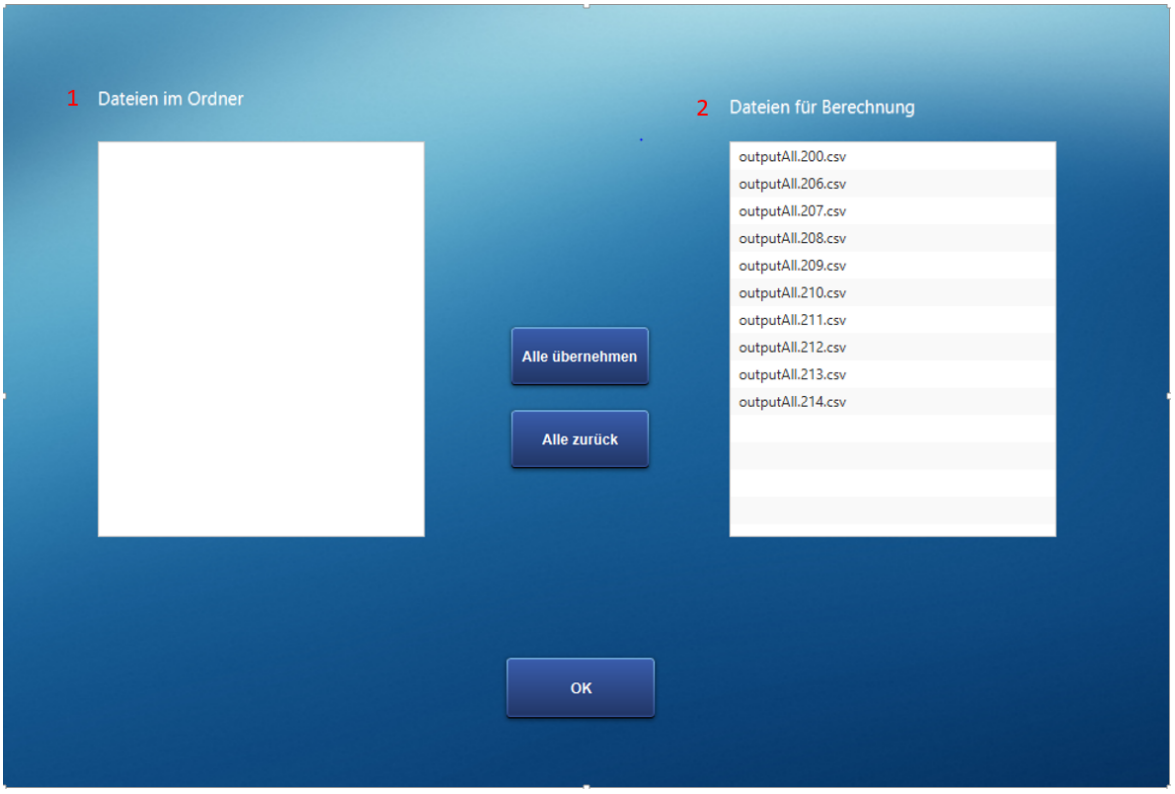


Abbildung C.2: Dateiauswahl

1. Die Dateien, die sich im Ordner des Datenpfades befinden, jedoch nicht für die Analyse ausgewählt werden sollen.

2. Die Dateien, die in die Analyse miteinbezogen werden sollen.

C.2.2 Variablenauswahl

In der Variablenauswahl wird festgelegt, welche Variablen für die Analyse ausgewählt werden. Es werden alle in den Dateien verfügbaren Variablen angezeigt, unabhängig davon, ob eine Variable auch in jeder Datei vorhanden ist.

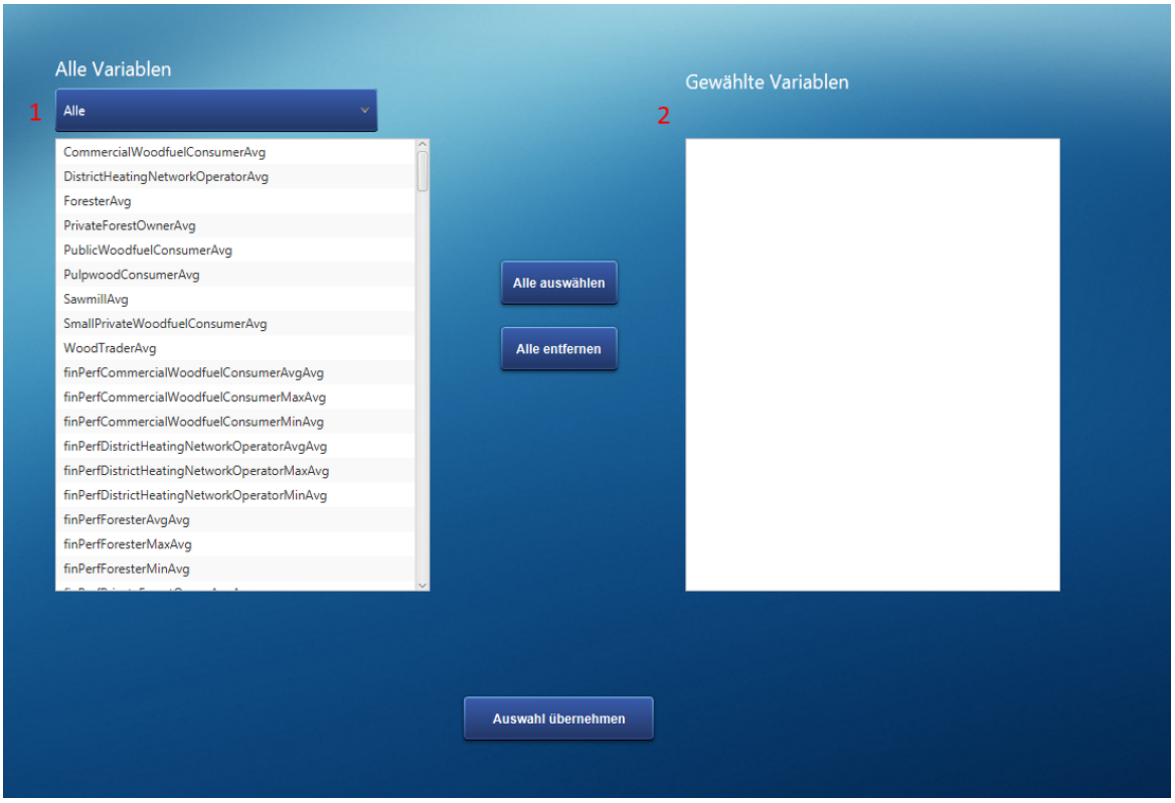


Abbildung C.3: Variablenauswahl

1. Die Variablen, die dem gewählten Filter entsprechen. Die Wahl “Alle“ bedeutet, dass alle in den Daten enthaltenen Variablen ausgewählt werden können. Andere Filter orientieren sich an den Akteuren des Modells und dienen zur besseren Übersichtlichkeit.

2. Die Variablen, die in die Analyse miteinbezogen werden sollen.

C.3 Variablenvergleich

Hier können die Einstellungen für die Analyse festgelegt werden.

C.3.1 Einstellungen

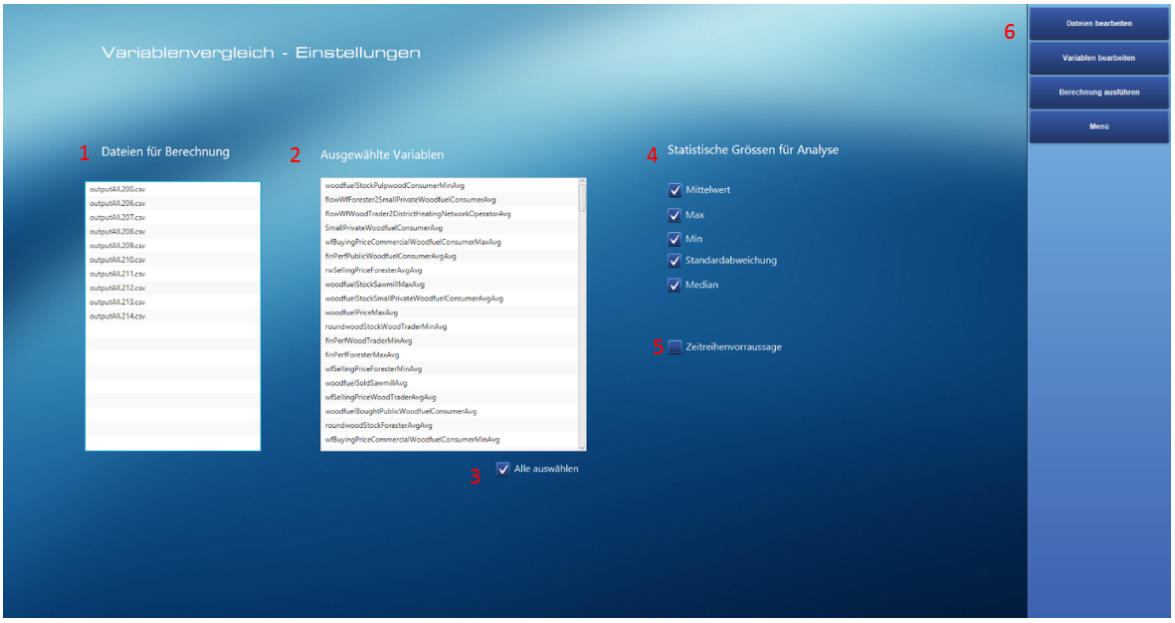


Abbildung C.4: Variablenvergleich - Einstellungen

1. Die Dateien, die in die Berechnung miteinbezogen werden.
2. Die Variablen, die in die Berechnung miteinbezogen werden.
3. Wenn angewählt, werden alle in den Dateien vorhandenen Variablen ausgewählt.
4. Die angewählten Kennzahlen werden analysiert.
5. Zeitreihenvoraussage zur Voraussage von 24 Monaten über den letzten gemessenen Zeitpunkt hinaus (Achtung hohe Zeitkomplexität, nur bei wenigen Variablen aktivieren).
6. Die Navigationsleiste
   - Daten bearbeiten: Auswahl der „Dateien für Berechnung“
   - Variablen bearbeiten: Auswahl der „Ausgewählte Variablen“
   - Berechnung ausführen: Starten der Analyse
   - Menü: Zurück zum Hauptmenü

C.3.2 Resultate

Diese Ansicht zeigt die Resultate nach erfolgter Analyse.

Abbildung C.5: Variablenvergleich - Resultate

1. Die Analyseergebnisse in tabellarischer Form. Durch Anklicken einer Zeile werden die entsprechenden Daten im aktiven Diagramm angezeigt. Es können beliebig viele Zeilen aus der Tabelle angewählt werden, welche dann - sofern vom Diagrammtyp unterstützt - simultan angezeigt werden.

2. Durch Klicken auf den "Zurück"-Button gelangt man zur Einstellungsansicht.

3. Der momentan aktive Diagrammtyp, der durch Klicken gewechselt werden kann. Nach einem Wechsel des Diagrammtyps, werden dieselben Daten im neuen Diagramm angezeigt.

4. Anzeige der gewählten Daten im zeitlichen Verlauf.

5. Die momentan aktive Datei aus der die Daten angezeigt werden. Beim erstmaligen Wechsel auf diese Ansicht liegt der Eingabefokus auf der Tabelle. Durch einmaliges Betätigen der TAB-Taste, wechselt der Fokus auf dieses Kontrollelement. Mit den Pfeiltasten kann danach zwischen den Dateien gewechselt werden.

6. Wird aktiviert, sobald ein Zoom-Bereich durch Klick und Halten der Maus auf dem Diagramm festgelegt wurde. Durch Klick wird auf den ausgewählten Bereich vergrössert.

7. Die Ansicht wird wieder zurückgesetzt.

C.3.3 Weitere Diagramme

C.3.3.1 Balkendiagramm

Das Balkendiagramm zeigt die Werte der zuvor ausgewählten statistischen Kennzahlen. Alle gewählten Variablen werden nebeneinander angezeigt.

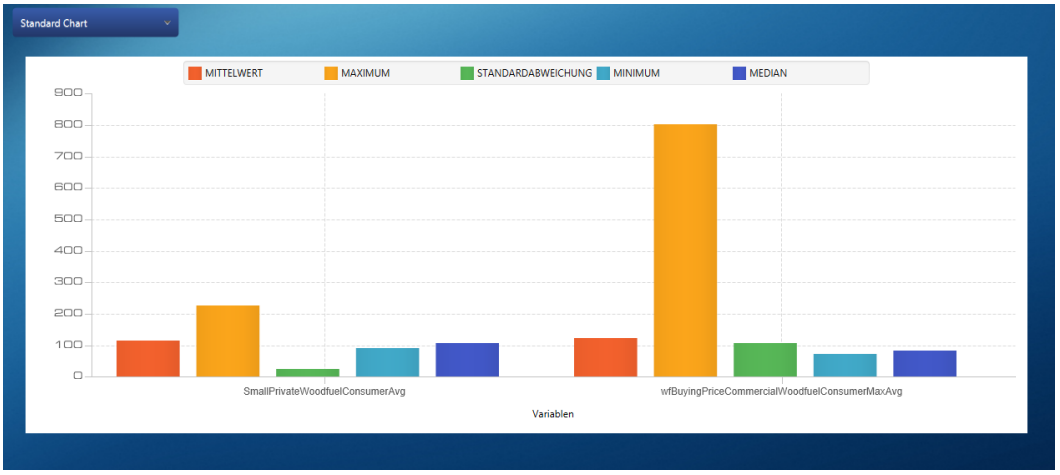


Abbildung C.6: Balkendiagramm

C.3.3.2 Box Plot

Auch in diesem Diagramm werden alle Variablen nebeneinander angezeigt. Aus Gründen der Übersichtlichkeit wird allerdings nur eine Auswahl der Dateien angezeigt (wenn mehr Variablen angezeigt werden, geht dies auf Kosten der Anzahl an gezeigten Dateien):

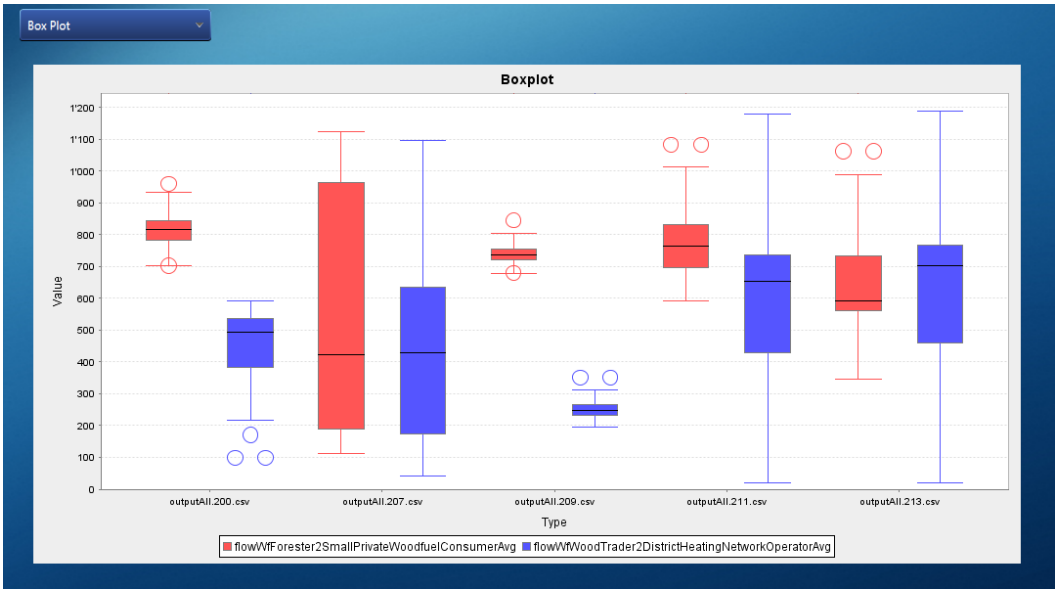


Abbildung C.7: Box Plot

C.3.3.3 Histogramm

Im Histogramm kann nur eine Variable gleichzeitig angezeigt werden. Ausserdem müssen die statistischen Kennzahlen Minimum und Maximum angewählt worden sein, um das Histogramm zu generieren.

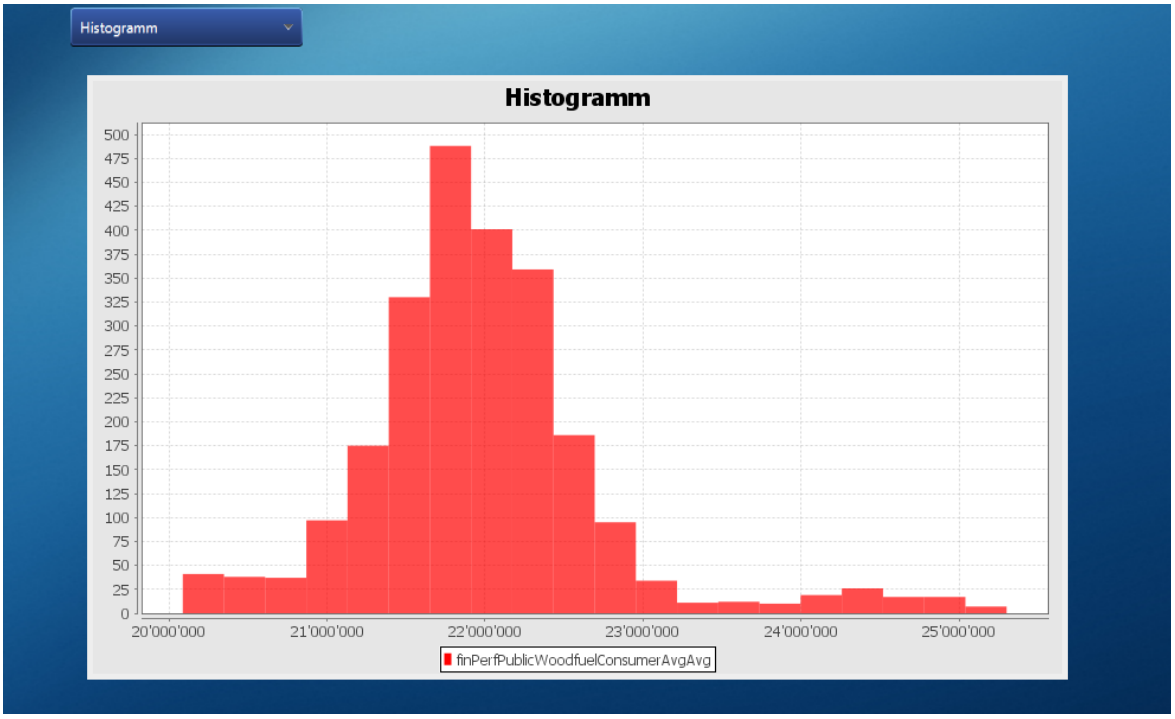


Abbildung C.8: Histogramm

C.4 Datensatzvergleich

Hier können die Einstellungen für die Analyse festgelegt werden.

C.4.1 Einstellungen

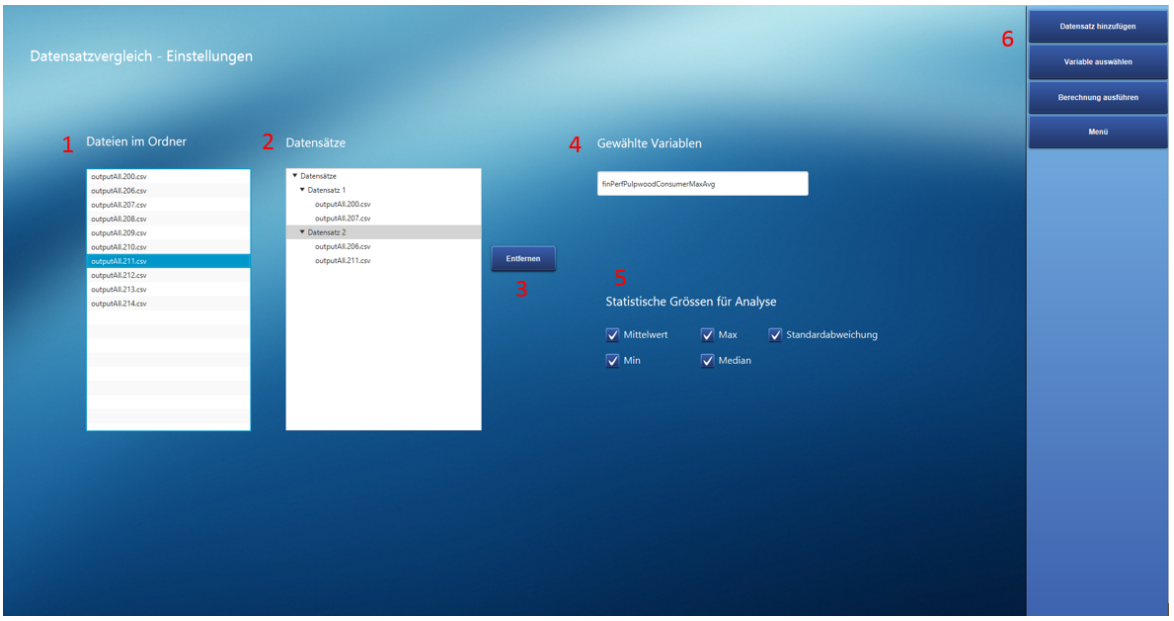


Abbildung C.9: Datensatzvergleich - Einstellungen

1. Die Dateien, die im Datenordner zur Verfügung stehen.
2. Die Datensätze für die Berechnung. Durch Drag&Drop können Dateien aus 1 zum einem beliebigen Datensatz hinzugefügt werden.
3. Der angewählte Datensatz wird entfernt.
4. Die Variable, die in die Berechnung miteinbezogen wird.
5. Die angewählten Kennzahlen werden analysiert.
6. Die Navigationsleiste
   - Datensatz hinzufügen: Ein leerer Datensatz wird zu 2 hinzugefügt
   - Variablen bearbeiten: Auswahl der “Gewählte Variablen“
   - Berechnung ausführen: Starten der Analyse
   - Menü: Zurück zum Hauptmenü

C.4.2 Resultate

Diese Ansicht zeigt die Resultate nach erfolgter Analyse. Die Ansicht unterscheidet sich in der Funktionalität nicht von der Resultatansicht des Variablenvergleichs. Es werden jedoch immer alle Daten und Dateien aus dem Datensatz miteinander angezeigt.

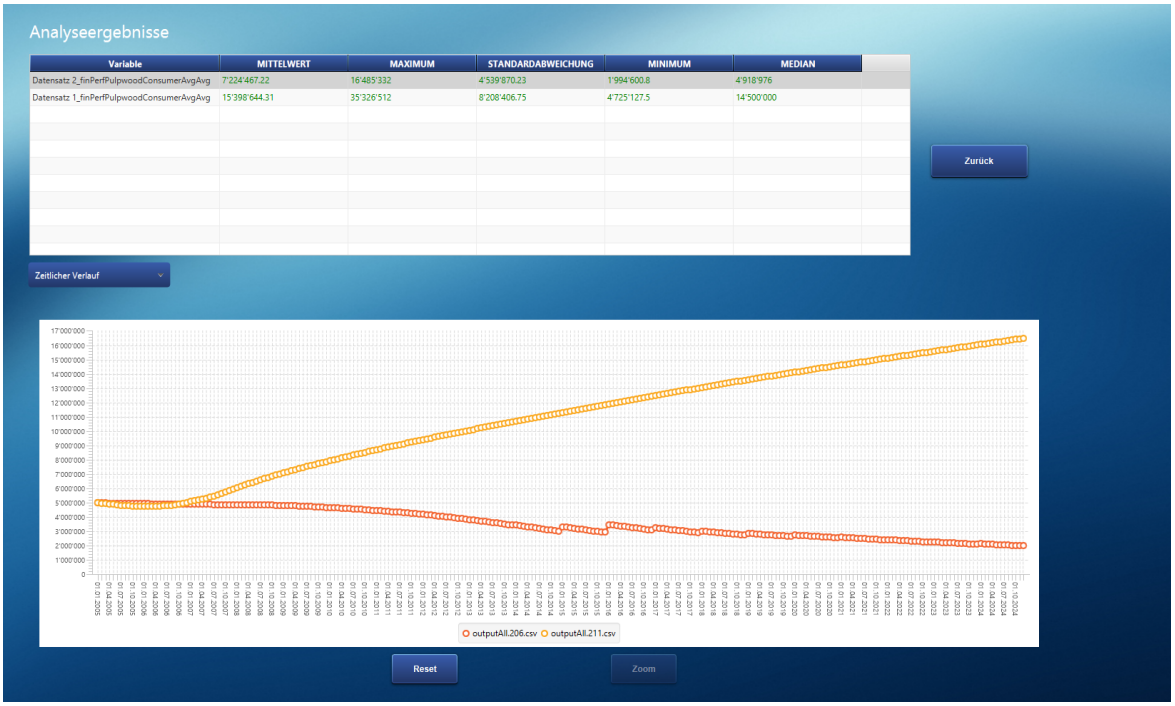


Abbildung C.10: Datensatzvergleich - Resultate

C.5 Szenarien

C.5.1 Überblick

Die Übersichtseite des Szenario-Modus.

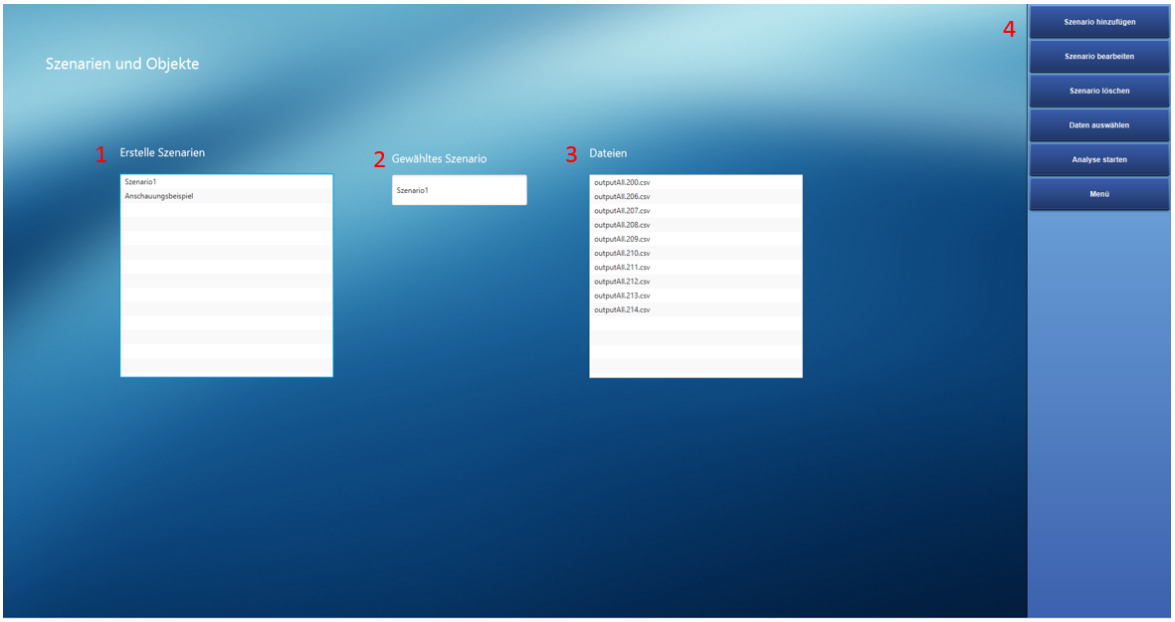


Abbildung C.11: Szeanrio - Übersichtsseite

1. Die bereits erstellten Szenarien, die in der Datenbank gespeichert sind.
2. Das Szenario, welches analysiert werden soll.
3. Die Dateien, die in die Berechnung miteinbezogen werden.
4. Die Navigationsleiste
    - Szenario hinzufügen: Der Szenario-Editor wird gestartet
    - Szenario bearbeiten: Klicken, um das in 1 angewählte Szenario zu bearbeiten
    - Szenario löschen: Klicken, um das in 1 angewählte Szenario zu löschen
    - Daten auswählen: Auswahl der Dateien für die Analyse
    - Analyse starten: Starten der Analyse des ausgewählten Szenarios
    - Menü: Zurück zum Hauptmenü

C.5.2 Erstellung/Bearbeitung

Der Szenario-Editor zum Erstellen und Bearbeiten von Szenarien. Wird ein neues Szenario erzeugt, sind alle Felder leer. Wird ein Szenario bearbeitet, werden die gespeicherten Daten in die jeweiligen Felder geladen. Die Erstellung eines Szenarios erfolgt in 4 Schritten.

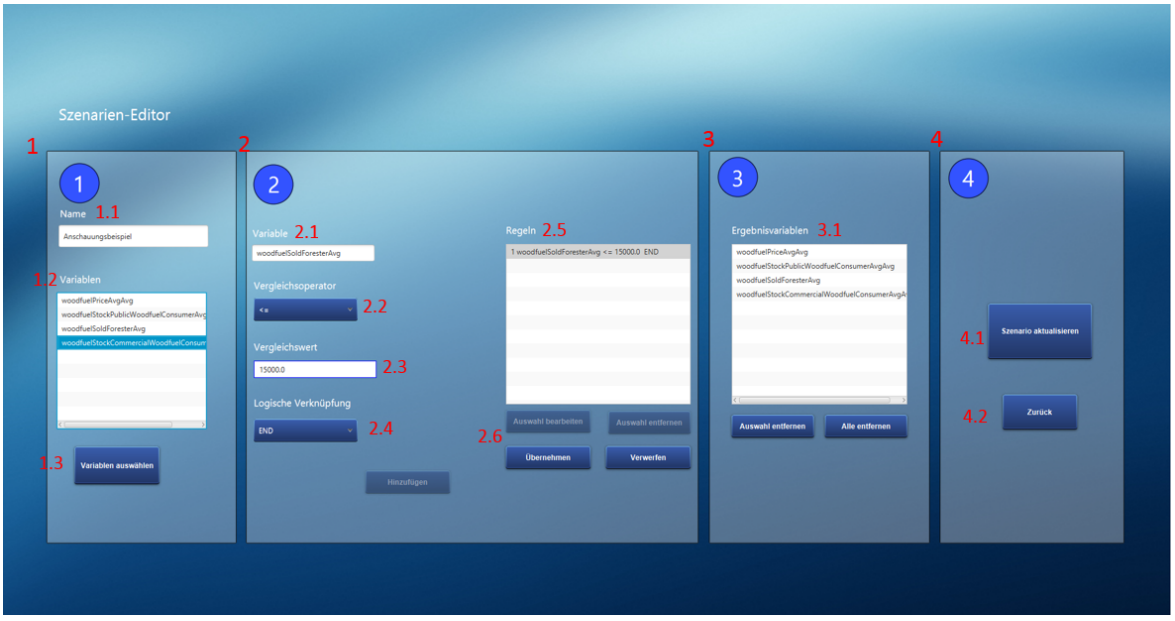


Abbildung C.12: Szeanrio-Editor

1. Der erste Schritt besteht daraus, dem Szenario einen eindeutigen Namen zu geben und die Variablen, die für dieses Szenario verwendet werden sollen, zu bestimmen.
   1.1. Der Name des Szenarios. Kann bei einer Bearbeitung des Szenarios nicht mehr gewechselt werden.
   1.2. Die Variablen, die für das Szenario verwendet werden können.
   1.3. Zur Variablenauswahl.
2. Im zweiten Schritt werden die Regeln für das Szenario erstellt oder verändert.
   2.1. Die Variable, auf die sich die Regel beziehen soll. Wird durch Drag&Drop aus 1.2 hinzugefügt.
   2.2. Der Vergleichsoperator wird festgelegt.
   2.3. Der Vergleichswert, der in Zusammenhang mit dem Vergleichsoperator verwendet werden soll, wird festgelegt. Falls als Vergleichsoperator eine Abweichung gewählt wurde, muss hier die maximale oder minimale prozentuale Abweichung angegeben werden. Es erscheint dann auch zusätzlich ein Feld zur Angabe, auf welche statistische Kennzahl sich die Abweichung beziehen soll.

2.4. Die logische Verknüpfung mit der nächsten Regel. Bei der letzten Regel sollte END gewählt werden.

2.5. Die Übersicht über die erstellten Regeln.

2.6. Schaltflächen zum Ändern oder Löschen bereits erstellter Regeln.

3. Der dritte Schritt dient zur Festlegung der Ergebnisvariablen, die bei Anwendung des Szeanrios ausgewertet werden.

3.1. Übersicht über die Ergebnisvariablen. Hinzufügen durch Drag&Drop aus 1.2.

4. Als letztes muss die Erstellung oder Bearbeitung des Szenarios bestätigt werden. Erst dann werden die Änderungen in die Datenbank geschrieben.

4.1. Bestätigung der Erstellung/Bearbeitung.

4.2. Zurück zur Übersichtsseite. Alle Änderungen werden verworfen.

C.5.3 Resultate

Die Resultate der Analyse des Szenarios.

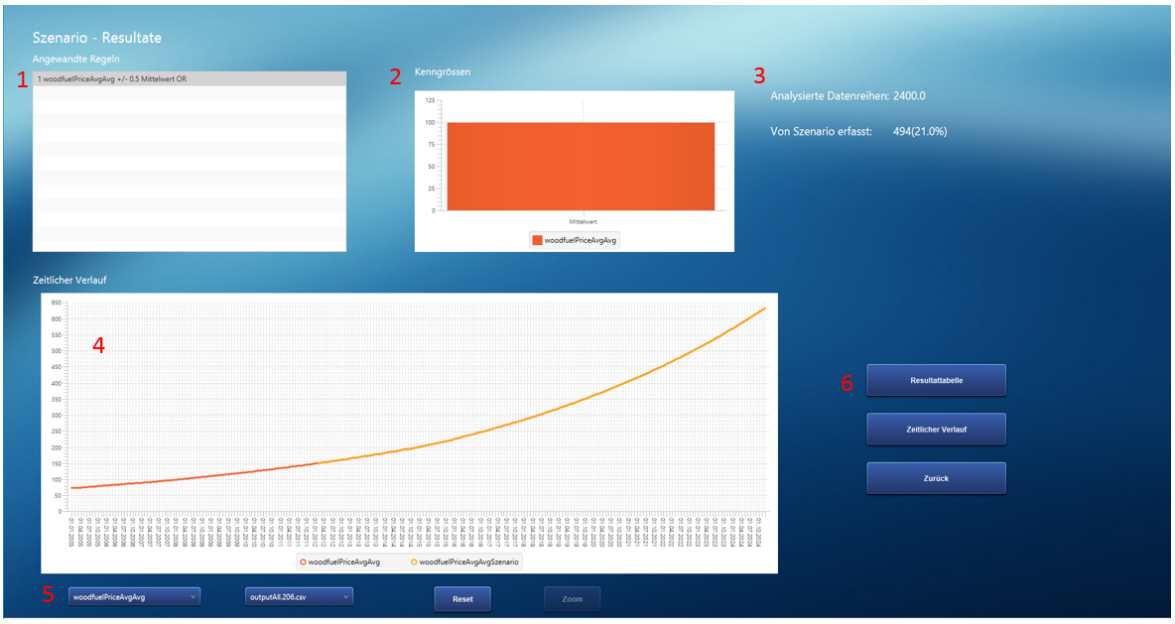


Abbildung C.13: Szeanrio-Resultate

1. Die für das Szenario erstellten Regeln in der Übersicht
2. Wird bei einer Regel die Abweichung von einer Kenngrösse miteinbezogen, so wird der tatsächliche Wert oder die Werte (im Beispiel nur der Mittelwert) in diesem Diagramm angezeigt.
3. Anzahl der analysierten Datenreihen insgesamt und wie viele davon vom Szenario erfasst wurden.
4. Die Werte der gewählten Variable im zeitlichen Verlauf. Gelb eingefärbt der Bereich, der vom Szenario erfasst wurde.
5. Die Wahl der Ergebnisvariable, die angezeigt werden soll.
6. Es kann zwischen der Ansicht des zeitlichen Verlaufs oder einer tabellarischen Darstellung gewechselt werden. In der Tabelle sind nur die Zeitpunkte und Werte eingetragen, die auch vom Szenario erfasst wurden.

C.6 Clusteranalyse

C.6.1 Einstellungen

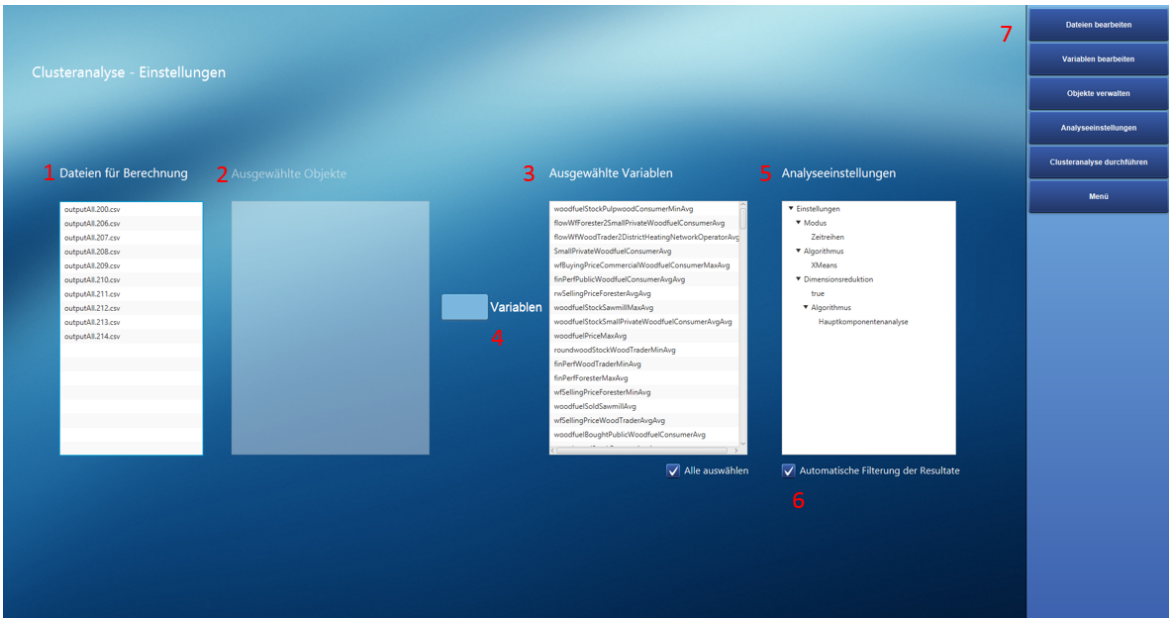


Abbildung C.14: Clusteranalyse-Einstellungen

1. Die Dateien, die in die Berechnung miteinbezogen werden.
2. Die Objekte, die in die Berechnung miteinbezogen werden.
3. Die Variablen, die in die Berechnung miteinbezogen werden.
4. Mit der Schaltfläche kann zwischen Variablen und Objekten umgeschaltet werden.
5. Die gewählten Analyseeinstellungen.
6. Wenn angewählt, werden die Clusterergebnisse nach abgeschlossener Berechnung, unter Berücksichtigung der Kriterien für relevante Resultate, gefiltert.
7. Die Navigationsleiste
   - Daten bearbeiten: Auswahl der „Dateien für Berechnung“
   - Variablen bearbeiten: Auswahl der „Ausgewählte Variablen“
   - Objekte verwalten: Aufrufen des Objekt-Editors
   - Analyseeinstellungen: Aufrufen des Editors zur Festlegung der Analyseeinstellungen
   - Clusteranalyse durchführen: Starten der Clusteranalyse
   - Menü: Zurück zum Hauptmenü

C.6.2 Objekt-Editor

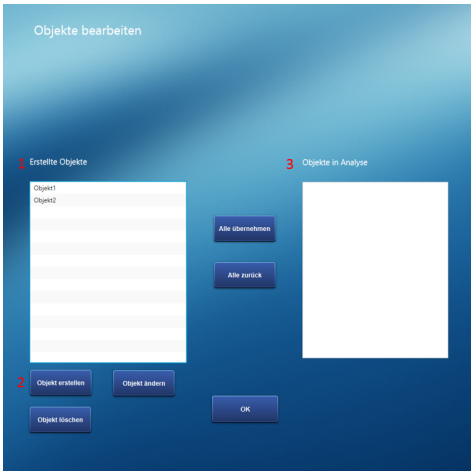


Abbildung C.15: Objekt-Übersicht

1. Bereits erstellte Objekte, gespeichert in der Datenbank.
2. Schaltflächen zum Erstellen, Bearbeiten oder Löschen von Objekten.
3. Objekte, die in die Berechnung miteinbezogen werden sollen.

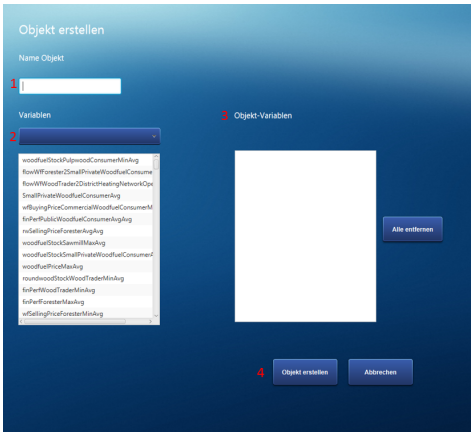


Abbildung C.16: Objekt-Erstellung

1. Eindeutiger Name des Objekts.
2. Schaltfläche zum Filtern der Variablen.
3. Variablen, aus denen das Objekt bestehen soll.
4. Erst nach dem Klick werden die Änderungen in die Datenbank geschrieben und das Objekt erstellt.

C.6.3 Analyseeinstellungen

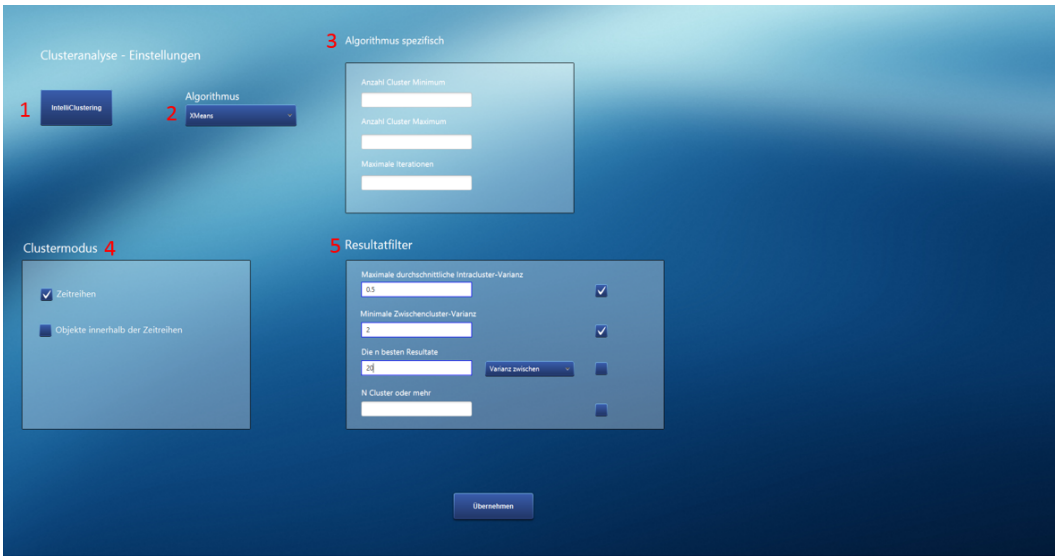


Abbildung C.17: Clusteranalyse-Analyseeinstellungen

1. Wenn eingeschaltet, wird die Konfiguration des Algorithmus (X-Means) vom Programm übernommen. Wenn ausgeschaltet, muss die Wahl und die Konfiguration des Clusteralgorithmus selbst durchgeführt werden.
2. Wahl des Clusteralgorithmus.
3. Die Konfigurationseinstellungen des gewählten Algorithmus.
4. Wahl des Clustermodus:
   - Zeitreihen: Clustern von Zeitreihen (empfohlen). Cluster bestehen aus kompletten Zeitreihen.
   - Objekte innerhalb der Zeitreihen: Jedes Objekt oder Variable wird einzeln behandelt. Es wird keine Rücksicht auf zeitliche Abhängigkeiten genommen und dient der Strukturentdeckung innerhalb einer Zeitreihe.
5. Resultatfilter:
   - Maximale durchschnittliche Intracluster-Varianz: Grenzwert, der von einer Clusterlösung nicht überschritten werden darf, um als Resultat zu gelten.
   - Minimale Zwischencluster-Varianz: Grenzwert, der überschritten werden muss, um als Resultat zu gelten.
   - Die n besten Resultate: Es werden nur die n besten Resultate angezeigt. Muss sich entweder auf die Intracluster- oder Zwischencluster-Varianz stützen.
   - n Cluster oder mehr: Grenzwert, wie viele Cluster mindestens gefunden werden müssen, um als Resultat zu gelten.

C.6.4 Resultate

Abbildung C.18: Clusteranalyse-Resultate

1. Übersicht über die analysierten Variablen. Durch Klicken auf eine Variable werden die entsprechenden Daten in die Tabellen und Diagramme geladen.
2. Übersicht über die gefundenen Cluster und deren Kennzahlen der angewählten Variable.
3. Falls mehrdimensionale Objekte analysiert werden, sind hier die statistischen Variablen zu finden aus denen das Objekt besteht.
4. Weitere Informationen:
   - Anzahl Objekte: Wie viele Objekte der ausgewählten Variable analysiert wurden.
   - Dimensionen: Die Dimension der Objekte der ausgewählten Variable.
   - Anzahl Cluster: Die Anzahl an gefundenen Clustern.
   - Anzahl Resultate: Gesamtanzahl an analysierten Variablen. Bezieht sich nicht auf die ausgewählte Variable.
5. Wahl des Diagrammtyps.
6. Die Darstellung der Cluster im zeitlichen Verlauf. Farblich beziehen sich die einzelnen Cluster auf Tabelle 2.
7. Auswahl welches Cluster angezeigt werden soll. Bei “Alle Cluster“ werden alle Cluster miteinander angezeigt.

Alternativer Diagrammtyp zur Anzeige der Clusterresultate. Alle Cluster werden einzeln angezeigt, jedoch können bis zu vier Cluster nebeneinander angezeigt werden.

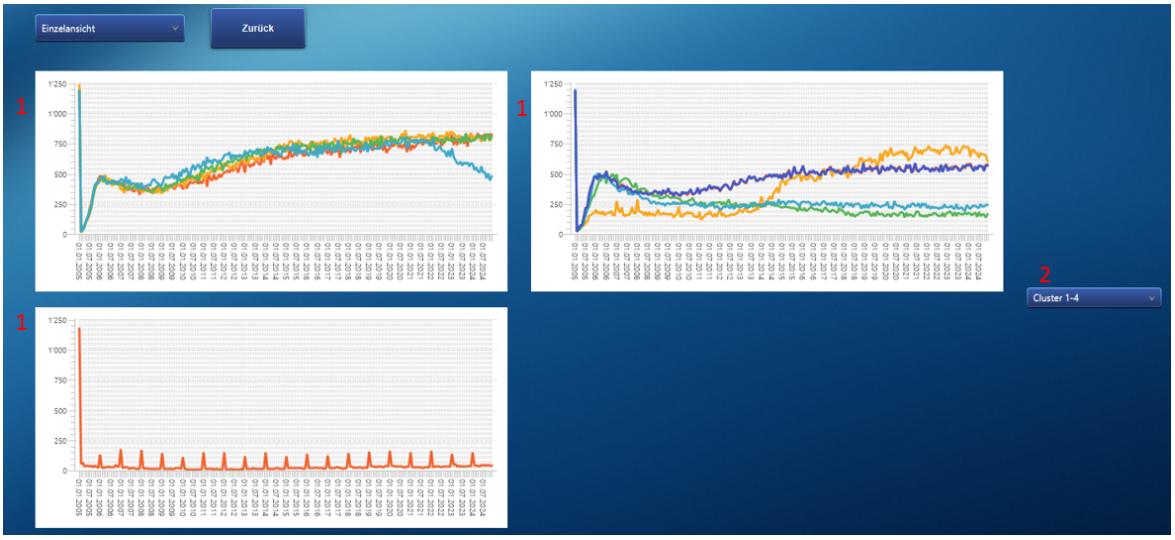


Abbildung C.19: Clusteranalyse-Einzelansicht

1. Die einzelnen Cluster.
2. Bei mehr als vier Clustern, kann hier auf die nächsten vier Cluster geschaltet werden.

Anzeige der Clusterresultate im Steudiagramm.

Abbildung C.20: Clusteranalyse-Streudiagramm

1. Die einzelnen Objekte eines Clusters als Punkte.
2. Die Legende zeigt auf, welche Punkte zu welchem Cluster gehören.

Literaturverzeichnis

Bacher, J., Pöge, A., & Wenzig, K. (2010). *Clusteranalyse Anwendungsorientierte Einführung in Klassifikationsverfahren.* München: Oldenbourg Wissenschaftsverlag GmbH.

Borg, I. (2000). Explorative Multidimensionale Skalierung. *GESIS-How-to.*

Cornish, R. (2007). Statistics: Cluster Analysis. *Mathematics Learning Support Centre.*

Ester, M., Kriegel, H.P., Sander, J., & Xu, X. (1996). *A density-based algorithm for discovering clusters in large spatial databases with noise.* Abgerufen am 23. April von Association for the Advancement of Artificial Intelligence: http://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

Fodor, I. K. (9. Mai 2002). *A survey of dimension reduction techniques.* Livermore: Center for Applied Scientific Computing, Lawrence Livermore National Laboratory.

Fowler, M. (19. Juli 2004). *Presentation Model.* Abgerufen am 02. April 2014 von Martin Fowler Website: http://martinfowler.com/eaaDev/PresentationModel.html

Fowler, M. (18. Juli 2006). *GUI Architectures.* Abgerufen am 2. April 2014 von Martin Fowler Website: http://martinfowler.com/eaaDev/uiArchs.html

Freeman, E., & Freeman, E. (2004). *Head First Design Patterns.* Sebastopol: O'Reilly Media, Inc.

Gossman, J. (Oktober 2005). *Introduction to Model/View/ViewModel pattern for building WPF apps.* Abgerufen am 03. April 2014 von MSDN Blogs: http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx

Harvey, B., & Wright, M. (1999). *Simply Scheme: Introducing Computer Science.* Cambridge: MIT Press.

Holm, S. (2011). *Design und Implementierung eines agentenbasierten Modells des Schweizer Energieholzmarktes.* Abgerufen am 5. März 2014 von Institut für Informatik, Universität Zürich: http://www.ifi.uzh.ch/isr/research/masisr/2011_07_MA_Holm_Stefan.pdf

Iglesias, F., & Kastner, W. (24. Januar 2013). Analysis of Similarity Measures in Times Series Clustering for the Discovery of Building Energy Patterns. *energies,* S. 579-597.

Jann, B. (2005). *Einführung in die Statistik.* München: Oldenbourg Wissenschaftsverlag GmbH.

Kopp, J., & Lois, D. (Dezember 2009). *Clusteranalyse.* Chemnitz, Sachsen, Deutschland.

Manning, C. D., Raghavan, P., & Schütze, H. (2009). *An Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Moore, D. S., McCabe, G. P., Alwan, L. C., Craig, B. A., & Duckworth, W. M. (2011). *The Practise of Statistics for Business and Economics.* New York: W.H. Freeman and Company.

Oracle. (n.d.). *JavaFX Frequently Asked Questions.* Abgerufen am 4. April 2014 von Oracle: http://www.oracle.com/technetwork/java/javafx/overview/faq-1446554.html#6

Pawlan, M. (April 2013). *What Is JavaFX?.* Abgerufen am 4. April 2014 von Oracle: http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

Pelleg, D., & Moore, A. (2000). X-means: Extending K-means with Efficient Estimation of the Number of Clusters. *Proceedings of the Seventeenth International Conference on Machine Learning*, 727-734.

Potel, M. (1996). *MVP: Model-View-Presenter The Taligent Programming Model For C++ and Java.* Abgerufen am 1. April 2014 von http://www.wildcrest.com/:http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

Pretzer, M. (5. Februar 2003). *Clustering und Klassifikation.* München: Oldenbourg Wissenschaftsverlag GmbH.

Stein, P., & Vollnhals, S. (1. April 2011). *Grundlagen clusteranalytischer Verfahren.* Essen: Institut für Soziologie, Universität Duisburg-Essen.

Schlittgen, R. (2000). *Einführung in die Statistik.* München: Oldenbourg Wissenschaftsverlag GmbH.

Shlens, J. (3. April 2014). *A Tutorial On Principal Components Analysis.* Abgerufen am 14.April 2014 von http://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

Smith, J. (Februar 2009). *WPF Apps With The Model-View-ViewModel Design Pattern.* Abgerufen am 3. April 2014 von MSDN Microsoft: http://msdn.microsoft.com/en-us/magazine/dd419663.aspx

Vos, J. (21. Januar 2014). *JavaFX and Android.* Abgerufen am 4. April 2014 von DZone.com: http://java.dzone.com/articles/javafx-and-android

Wiedenbeck , M., & Züll, C. (2001). Klassifikation mit Clusteranalyse: Grundlegende Techniken hierarchischer und K-means-Verfahren. *GESIS-How-to*.