# Clustering genetischer Daten auf der Basis eines konsistenten Softwareframeworks zur Datenverwaltung in klinischen Studien

Inaugural-Dissertation zur Erlangung der Doktorwürde der
Philosophischen Fakultät III
(Sprach- und Literaturwissenschaften)
der Universität Regensburg

vorgelegt von

**Christian Rengstl M.A.**

93077 Bad Abbach - Oberndorf
2010

Erstgutachter: Prof. Dr. Christian Wolff
Zweitgutachter: Prof. Dr. Rainer Hammwöhner

# Danksagung

Mein Dank gilt vor allem Prof. Dr. Christian Wolff, Professur für Medieninformatik an der Universität Regensburg, und Prof. Dr. Rainer Hammwöhner, Lehrstuhl für Informationswissenschaften an der Universität Regensburg, für die Betreuung meiner Promotion, sowie Prof. Dr. Christian Hengstenberg an der Klinik und Poliklinik für Innere Medizin II am Universitätsklinikum Regensburg für die fachliche und finanzielle Unterstützung. Desweiteren geht mein Dank an Dr. med Marcus Fischer und Dr. rer. nat. Klaus Stark an der Klinik und Poliklinik für Innere Medizin II am Universitätsklinikum Regensburg für die großartige fachliche Unterstützung, ohne die diese vorliegende Arbeit nicht möglich gewesen wäre. Weiterer Dank geht auch an Fr. Dr. rer. nat. Ute Hubauer ebenfalls an der Klinik und Poliklinik für Innere Medizin II der Universität Regensburg und Fr. Silke Muyrers für ihre Unterstüzung und ihr Feedback sowie an die gesamte Arbeitsgruppe Genetik an der Universitätsklinik Regensburg. Ein besonderer Dank geht ebenfalls an meine Eltern für ihre bedingungslose Unterstützung. Desweiteren möchte ich meiner Frau großen Dank aussprechen für ihre seelische und moralische Unterstützung und ihre aufmunternden Worte, denn ohne sie wäre vermutlich diese ganze Arbeit nicht zustande gekommen.

Für Ariadne und Diana

# Inhaltsverzeichnis

# Kapitel 1.

# Einleitung und Motivation

Im Rahmen epidemiologischer Studien im Bereich der Genetik werden heutzutage vor allem quantitative Analysen von Studiendaten und, unter Umständen darauf aufbauend, weiterführende genetische Untersuchungen von DNA-Proben durchgeführt. Da, basierend auf den so erhobenen Daten, wichtige Schlussfolgerungen gezogen werden, ist es überaus wichtig, dass die Daten in hoher Qualität vorliegen. Hohe Qualität bedeutet in diesem Fall, dass die Daten sowohl bei der Erhebung durch medizinisches Personal richtig ermittelt und hinterlegt werden als auch in einem Format gespeichert werden, das die Stabilität und Qualität der Daten gewährleistet und es ermöglicht die Daten ohne große Programmierkenntnisse wieder zu ermitteln. Der erste Punkt lässt sich nur durch äusserst sorgfältige Datenerhebung und -speicherung gewährleisten zumal die Qualität der Daten oftmals auch von ungenauen Angaben der Studienteilnehmer beeinträchtigt wird. Leider sind die Angaben der Patienten bzgl. eines Krankheitsverlaufs bzw. der Krankengeschichte nicht schlüssig oder korrekt was von Anfang an die Qualität der Studiendaten beeinträchtigen kann. Selbstverständlich können unplausible Daten auch vom Hausarzt des Patienten hinterfragt werden, jedoch muss dabei mit erheblichem zeitlichen und finanziellen Aufwand gerechnet werden. Vor allem die Dateneingabe und -speicherung sowie die Extraktion der Daten können dahingehend positiv beeinflusst werden, dass unplausible und unstimmige Daten vermieden werden. Dies lässt sich durch geeignete Softwareprodukte umsetzen, die die Anwender dahingehend unterstützen und somit die Qualität der Daten steigern. Der zweite Aspekt bei der

Durchführung klinischer Studien - in diesem Falle die Analyse genetischer Daten - stellt ebenfalls eine Herausforderung sowohl an die Qualität der Datengrundlage sowie an die Auswertesoftware bzw -algorithmen nicht zuletzt auf Grund der enormen Datenmengen genetischer Daten dar. Jedoch ist die Analyse genetischer Daten ein langwieriger Prozess, bei dem man erst nach einigen Jahren zuverlässige und beweisbare Ergebnisse erzielen kann. Zwar gibt es heuristische Methoden die Qualität von Untersuchungsergebnissen zu verifizieren, jedoch geben letztendlich nur in-vitro-Versuche oder Tierversuche zuverlässig Auskunft über die Korrektheit von statistischen Analyseergebnissen. Derzeit gibt es eine Vielzahl von Bioinformatikapplikationen, die den Biologen bei der Auswertung und der Analyse genetischer Daten unterstützen sollen. Angefangen bei der Genotypisierung von DNA-Proben über die Visualisierung genetischer Daten bzw. berechneter Werte basierend auf den Daten bis hin zur eigentlichen statistischen Auswertung stehen für alle Bereiche passende Applikationen zur Verfügung. Da allerdings für die Auswertung eine Vielzahl von Algorithmen und Applikationen zur Verfügung steht, ist der Biologe vor die Aufgabe gestellt zum einen die richtigen Algorithmen bzw. Applikationen auszuwählen und zum anderen auch die mögliche Vielzahl der Ergebnisse schlüssig zu interpretieren, da für dieselbe Datengrundlage mehrere Algorithmen bzw. Auswertestatistiken angewandt werden können was zu unterschiedlichen Ergebnissen führen kann. Darüber hinaus sind Applikationen in diesem Bereich oft modellbasiert, was bedeutet, dass die Ergebnisse unter der Annahme eines genetischen Modells, also zum Beispiel ob rezessive oder dominante Allele bzw. Erbgut vorliegt, berechnet wird. Dieser Umstand erschwert daher die zuverlässige und eine Auswertung der erhaltenen Ergebnisse, die einen generellen Überblick ohne der Annahme eines genetischen Modells gibt.

Da die für die Auswertung benötigten genetischen Daten von den Daten abhängen, die während der Durchführung klinischer Studien erhoben wurden, ist nicht nur die tatsächliche Analyse genetischer Daten wichtig, sondern schon die Erhebung. Folglich ist es Ziel dieser Arbeit zum einen die Datenspeicherung, -verwaltung und -abfrage zu vereinheitlichen, was zu einer Steigerung der Qualität der Studienda-

ten und -ergebnisse führt, und zum anderen die Auswertung genetischer Daten dahingehend zu unterstützen, dass ein Softwarewerkzeug entwickelt werden soll, das den Biologen bei der Evaluierung der Ergebnisse unterstützen soll. Da sowohl die Verwaltung klinischer Studien sowie die Auswertung genetischer Daten auf einem homogenen Software- bzw. Datenbanksystem beruhen sollten, war es zuerst ein Teilziel ein einheitliches System zu entwickeln, das es allen Betroffenen ermöglichen sollte, zeitgleich an denselben zentral gespeicherten Daten zu arbeiten. Dieses System soll, wie erwähnt, desweiteren eine Grundlage für die Durchführung genetischer Studien sein, da die Daten klinischer Studien als Ausgangspunkt für eine eventuelle Genotypisierung von DNA und im hier vorliegenden Fall zum Clustering genetischer Daten dienen. Daher ist es von enormer Wichtigkeit, dass die erhobenen Daten hochqualitativ sind, um so die Qualität der DNA-Analyse nicht zu beeinträchtigen. Auf diesem einheitlichen System zur Datenhaltung aufbauend sollte dann ein Softwaretool entwickelt werden, das mit Hilfe eines Clusteringverfahrens auf der Basis des SOM-Algorithmus (siehe Kapitel 4.1.4), die Auswertung von Analysen genetischer Daten unterstützt, indem es Gemeinsamkeiten innerhalb der Datenmenge aufzeigt und so für eine genauere Analyse der verschiedenen Rechenergebnisse anderer Analysetools dienen kann. Auf diese Weise soll der Wissenschaftler unterstützt werden, die Relevanz und die Gemeinsamkeiten von Genen besser beurteilen zu können, was folglich zu einer effizienteren Analyse von genetischen Daten führt.

Als Vergleichsalgorithmus wurde der *k-means*-Algorithmus implementiert, um so Aussagen darüber treffen zu können, welcher Algorithmus für die zu Grunde liegende Aufgabe besser geeignet ist. Darüber hinaus lassen sich auch durch den Vergleich beider Algorithmen Rückschlüsse über die Qualität der Clusteringergebnisse treffen, indem für beide Algorithmen diverse Qualitätsmaße berechnet werden.

Die Analyse und die praktische Umsetzung der oben erwähnten Punkte fand am Universitätsklinikum Regensburg in der Klinik und Poliklinik für Innere Medizin II in der Arbeitsgruppe um Prof. Dr. Christian Hengstenberg statt. Dort wurden vor allem die informationstechnischen Aspekte der HIFAM- und Kora-500K-Studien (kooperative Gesundheitsforschung in der Region Augsburg) analysiert und die dar-

aus gewonnenen Ergebnisse praktisch umgesetzt. Ziel der Herzinfarktfamilienstudie (HIFAM) (siehe Hengstenberg (2006)) ist es die individuelle Früherkennung eines erhöhten Herzinfarktrisikos zu verbessern. Dazu werden von Patienten mit Herzkrankheiten und deren Familienmitgliedern anamnestische Daten und Blutproben erhoben. Das Thema der KORA-Studie ist ebenfalls die Entstehung von Herzkrankheiten sowie von Diabetes mellitus im Zusammenhang mit Risikofaktoren wie Rauchen, Ernährung, Umwelt und auch unter Berücksichtigung genetischer Voraussetzungen.

Die HIFAM-Studie diente dabei vor allem der Analyse des Workflows und des Datenmanagements, wohingegen die Kora-500K-Studie für die Aufbereitung und Clusteranalyse genetischer Daten genutzt wurde. Die Kenntnisse, die bei der Studiendurchführung und der Erhebung bzw. Speicherung der Patientendaten gesammelt wurden, dienten als Ausgangspunkt für das Design und die Implementierung einer einheitlichen Softwareinfrastruktur mit Hilfe derer die Qualität der Studientdaten und -ergebnisse sichergestellt werden soll. Diese Umsetzung erfolgte vor allem im Rahmen der GoKard-Studie des Universitätsklinikums Regensburgs unter der Leitung von Prof. Dr. Christian Hengstenberg, bei der ebenfalls genetische Prädispositionen für Erkrankungen des Herz- Kreislaufsystems untersucht werden.

Im folgenden Kapitel wird auf genetische Grundbegriffe eingegangen, um eine gute Ausgangslage für das Clustering genetischer Daten zu bieten. So werden neben dem Begriff des SNPs auch grundlegende Maße wie das Hardy-Weinberg-Equilibrium oder das Linkage Disequilibrium näher erläutert. Ausserdem werden wichtige Applikationen im Bereich der Bioinformatik sowie die wichtigsten Datenformate bzw. Gendatenbanken näher beschrieben und erklärt. Anschließend erfolgt ein Einblick in informationstechnische Aspekte und Probleme im Kontext klinischer Studien, sowie in den Workflow bei der Durchführung klinischer Studien. Außerdem wird ebenfalls eine Übersicht über Datenbanktheorie gegeben und die Vorteile des Einsatzes von relationalen Datenbanksystemen erläutert. Darauf aufbauend werden ebenfalls hier erstellte Datenmodelle näher beschrieben. Neben diesen theoretischen Aspekten wird aufbauend auf den theoretischen Grundlagen das

Design und die Implementierung eines Softwareframeworks, das speziell auf klinische Studien ausgelegt ist, erläutert. Im Anschluss an dieses Kapitel geht es dann um das eigentliche Clustering genetischer Daten, das auf das im vorangegangenen Kapitel vorgestellten Datenmodell beruht. Es wird eine allgemeine Einführung in die Thematik des Clusterings gegeben und darauf aufbauend der hier umgesetzte Algorithmus und der Grund für die Erstellung eines diversifizierten Gewichtsvektor, der ein zu clusterndes Gen repräsentiert, dargestellt. Im letzten Kapitel wird auf die generelle Problematik der Evaluierung eingegangen und eine grundlegende Evaluierung der Clusteringergebnisse beschrieben sowie ein Ausblick auf mögliche weitere Evaluierungsmöglichkeiten gegeben.

# Kapitel 2.

# Medizinisch bzw. biologische und bioinformatische Grundlagen

Da sich die vorliegende Arbeit sowohl mit der softwaretechnischen Durchführung klinischer Studien als auch mit Genetik bzw. der bioinformatischen Auswertung von Gendaten beschäftigt, dient dieses Kapitel zur Klärung häufig auftauchender biologischer Begriffe und Erklärung biologischer bzw. bioinformatischer Grundlagen. Dazu wird im ersten Teil auf rein biologische bzw. genetische Grundlagen und Termini eingegangen. In den darauf folgenden Abschnitten werden dann sowohl wichtige Datenformate bzw. Datenbanken als auch Applikationen im Bereich der Bioinformatik beschrieben.

Vorab soll allerdings der Begriff der Bioinformatik näher erläutert werden, da es sich hierbei um einen Terminus handelt, der im Gegensatz zu Medizin bzw. Biologie oder Informatik oftmals schwerlich einzuordnen ist, da es sich bei der Bioinformatik um eine interdisziplinäre Disziplin handelt. So bildet die Bioinformatik einen „Schmelztiegel molekularbiologischer, mathematischer und informatischer, aber auch biochemischer und biophysikalischer Sachkompetenz“ (Huett u. Dehnert (2006, 2)). Darüber hinaus ist es eine der Hauptaufgaben der Bioinformatik, Werkzeuge für die Analyse immer größer werdender Datenmengen bereitzustellen. Um diesen „Schmelztiegel“-Charakter zu erfüllen und die Bereitstellung von Analysewerkzeugen zu bewerkstelligen, sind folgende zentrale Aspekte für die Bioinformatik per se wichtig (vgl. Huett u. Dehnert (2006, 4ff)):

- Datenbanken zur Speicherung genetischer Daten, wie z.B. Gensequenzen, bzw. Fachliteratur
- Software etwa zur Konvertierung von Datenformaten, zur Auswertung genetischer Daten, etc.
- Mathematische Modelle zur Analyse biologischer Daten
- Datenorganisation, sprich Definition bzw. Bereitstellung geeigneter Datenformate sowie die Aufbereitung von genetischen Rohdaten

Basierend auf diesen Punkten, stellt die Bioinformatik eine wichtige Schnittstelle zwischen den humanwissenschaftlichen Disziplinen Medizin und Biologie auf der einen Seite und der Informatik auf der anderen Seite dar, die mit zunehmendem Wachstum der Datenmengen in den ersten Bereichen eine immer wichtigere Rolle einnehmen wird.

## 2.1. Allgemeine Grundlagen der Genetik

Nach dieser Erläuterung des Begriffs der Bioinformatik und dessen Einordnung in einen wissenschaftlichen Kontext werden im Folgenden einige grundlegende Termini aus dem Bereich Genetik näher beschrieben, um so eine Verständnisgrundlage für das weitere Vorgehen zu schaffen. So werden wichtige Fachbegriffe und genetische Maße, die für das Projekt wichtig sind, in ihren Grundzügen erklärt. Im Folgenden wird vor allem auf die für dieses Projekt wichtigen *single nucleotide ploymorphisms* (kurz SNPs) eingegangen, da diese die Datengrundlage für das Clustering darstellen. Desweiteren wird das genetische Maß des *linkage disequilibrium* (kurz LD) näher erörtert.

Zuvor soll allerdings ein kurzer Überblick über Genetik selbst gegeben werden. Im Jahr 1953 entdeckten die späteren Nobelpreisträger Francis Crick und James Watson die Struktur einer Doppelhelix der Desoxyribonukleinsäure (DNA). Diese

Doppelhelix ist ein komplexes Biomolekül, das das Erbgut aller lebenden Organismen enthält und sich bei höher entwickelten Organismen (Eukaryoten), also Tieren, Pilzen und Pflanzen, im Zellkern befindet. Dort bildet sie mehrere Chromosomenpaare. Beim Menschen 22 plus im Falle von Frauen einem X-Chromosomenpaar und bei Männern einem X-Y-Chromosomenpaar. Jedes Chromosom wiederum besteht aus Genen, die die eigentliche Erbinformation enthalten und von Generation zu Generation weitergegeben werden. Die DNS wiederum besteht aus Nukleotiden, die ihrerseits aus einem Phosphat, dem Zuckermolekül Desoxyribose und einer der vier organischen Basen Adenin, Thymin, Cytosin oder Guanin. Diese Basen bilden mit Hilfe von Wasserstoffbrücken Kombinationen. So bilden im Normalfall Adenin und Thymin (zwei Wasserstoffbrücken) sowie Cytosin und Guanin (drei Wasserstoffbrücken) Paare. Im Falle von Mutationen wie etwa „single nucleotide polymorphisms“ (siehe 2.1.1) können allerdings auch andere Kombinationen auftreten. Jeweils drei aufeinander folgende Basen legen in codierenden Genabschnitten den Aufbau einer Aminosäure fest, die wiederum Bestandteil eines aus der DNA übersetzten Proteins ist. Ein weiterer häufig auftauchender Begriff in der Genetik ist der Begriff des „Allels“ , worunter man eine mögliche Ausprägung eines Gens versteht. Das bedeutet, dass Allele eine alternative Basensequenz an einem bestimmten Locus eines Gens darstellen, was in einem alternativen Phänotypus resultiert wie zum Beispiel im Falle der Blutgruppe: Jeder Mensch hat eine von vier unterschiedlichen phänotypischen Ausprägungen der Blutgruppe (A, B, AB, 0), die auf sechs unterschiedliche Genotypen zurückzuführen sind. Für A sind das *AA* und *A0*, für B *BB* und *B0*, für AB *AB* und für 0 *00*. Diese sechs unterschiedlichen genotypischen Variationen beruhen dabei auf den drei Allelen, also den genetischen Ausprägungen *A*, *B* und *0*.

### 2.1.1. *Single nucleotide polymorphisms*

Nachdem im vorherigen Kapitel ein Überblick über Genetik bzw. DNA gegeben wurde, wird in diesem Kapitel der Begriff des *single nucleotide polymorphism*, der

eine wichtige Datengrundlage in dieser Arbeit darstellt, erklärt. Unter einem *single nucleotide polymorphism* (SNP) versteht man eine Mutation auf dem menschlichen Genom, die sich auf jeweils nur ein einzelnes Basenpaar beschränkt im Gegensatz zu z.B. sogenannten *tandem repeats*, die sich über mehrere Basenpaare erstrecken. Da diese *tandem repeats* keine praktische Relevanz für dieses Projekt haben, wird auf sie nicht näher eingegangen. Bei dieser Form der Punktmutation, also bei SNPs, wird lediglich eine Base durch eine andere ausgetauscht. Diese Art der Mutation ist mit ca. 90% die am häufigsten auftretende aller Mutationen im menschlichen Genom und tritt vermutlich alle 100 bis 300 Basenpaare auf, was eine große Abdeckung des menschlichen Genoms bedeutet. Desweiteren kann man sagen, dass ca. 85% aller SNPs bei allen Menschen auftreten. Unklar ist dabei allerdings noch zum Teil, ab wann eine Mutation als SNP gelten kann. Während eine lockere Einteilung (siehe Ziegler u. Koenig (2006, 50)) davon ausgeht, dass jede Punktmutation auf einem Basenpaar schon als SNP gewertet werden kann, gibt es im Gegensatz dazu eine striktere Definition, laut derer eine Mutation eine *minor allele frequency* von mindestens 1% innerhalb einer Population aufweisen muss ((Ziegler u. Koenig, 2006, 50)). Diese sogenannte *minor allele frequency* bezieht sich dabei auf das Verhältnis des selteneren zum häufigeren Allel innerhalb einer untersuchten Population.

Der Vorteil bei der genetischen Forschung mittels SNPs ist vor allem die große Stabilität der Mutationen. Das bedeutet, dass SNPs nur äusserst selten wieder Mutationen unterliegen. Konkret mutieren SNPs lediglich alle $2 * 10^{-8}$ Geburten im Gegensatz zu *short tandem repeats*, die ungefähr alle $2 * 10^{-3}$ Geburten mutieren (siehe Ziegler u. Koenig (2006, 58)). Diese Fülle von Daten und deren genetische Stabilität dient wiederum als guter Ausgangspunkt für genetische Analysen, da mit einer solchen Datenmenge die Abdeckung des menschlichen Genoms an Information und damit die Wahrscheinlichkeit steigt, dass ein SNP gefunden wird, der direkt für einen bestimmten Phänotypus verantwortlich ist.

Allerdings hat auch die genetische Analyse an Hand von SNPs Schattenseiten. So hat nämlich ein SNP einen äusserst geringen Informationsgehalt. Da ein SNP

immer nur als biallelischer Marker auftritt, ist die potentielle Information, die er enthält, äusserst gering. Basierend auf bisher verwendeten Informationsmaßen für genetische Marker erreicht ein SNP lediglich ein Informationsmaß von maximal 0.3750 mit dem *polymorphism information content* (PIC) bzw. maximal 0.5 mit dem Maß der *mean heterozygosity* (HET) wohingegen ein *tandem repeat* mit 5 Allelen Werte von 0.768 bzw. 0.8 erreichen kann (siehe Ziegler u. Koenig (2006, 44ff)). Unter der Annahme des *Hardy-Weinberg equilibriums* (siehe 2.1.2) ist die simpelste Form der Gleichung des HET-Maßes in Gleichung 2.1 gegeben, wobei $i$ für das $i$-te Allel, $m$ für die Anzahl der Allele und $p_i$ für die Häufigkeit des Allels $i$ steht.

$$HET = 1 - \sum_{i=1}^{m} p_i^2 \tag{2.1}$$

Die Gleichung für das PIC-Maß der Information eines genetischen Markers ist in 2.2 dargestellt. Wie in Gleichung 2.2 zu sehen ist basiert sie auf der Gleichung für HET (siehe 2.1) unter der Annahme von HWE. Sie wird allerdings noch dahingehend erweitert, dass das zweite Elternteil $j$ mit in die Gleichung integriert wird.

$$PIC = 1 - \sum_{i=1}^{m} p_i^2 - \sum_{i=1}^{m} \sum_{j=1_{i \neq i}}^{m} p_i^2 p_j^2 \tag{2.2}$$

Trotz des relativ niedrigen Informationsgehaltes von SNPs liegt deren Stärke in der Häufigkeit ihres Auftretens, was das niedrige Informationsmaß wieder ausgleicht. Waren im Jahr 1998 lediglich rund 5000 SNPs bekannt, so stieg diese Zahl im Jahr 2002 bereits auf rund 2,2 Millionen SNPs und im Jahr 2004 auf beachtliche 8,8 Millionen SNPs. Derzeit (Stand Aprill 2008) sind rund 55,9 Millionen SNPs in der *Single Nucleotide Polymorphism database* (dbSNP) erfasst. Darüber hinaus haben SNP-Daten eine geringere Fehlerquote im Hinblick auf Genotypisierungsfehler. Im Gegensatz zu *short tandem repeats* mit einer Fehlerquote von 0,4% bis 3% beträgt die Fehlerquote bei SNP-Daten lediglich 0,1% (siehe Ziegler u. Koenig (2006, 57))

### 2.1.2. Hardy-Weinberg Gleichgewicht

Da die oben erwähnten Informationsmaße für genetische Marker HET und PIC beide auf der Annahme des Hardy-Weinberg Gleichgewichts beruhen, wird im folgenden Abschnitt kurz auf dieses Modell aus der Populationsgenetik eingegangen. Auf den von einander unabhängig gemachten Erkenntnissen des englischen Mathematikers G. H. Hardy (1877 - 1947) und des deutschen Arztes Wilhelm Weinberg (1862 - 1937) beruht das sogenannte *Hardy-Weinberg equilibrium*. Dieses Gesetz der quantitativen Genetik aus dem Jahre 1908 besagt, dass die Gen- und die Genotyphäufigkeit innerhalb einer Population, deren Individuen sich mit gleich großer Wahrscheinlichkeit paaren, ohne Einfluss von Änderungen auf die Gene wie Selektion, Mutation, Migration, etc., stabil bleiben. Das Verhältnis im Falle autosomaler Gene, d.h. Gene, die sich nicht auf dem geschlechtsspezifischen Chromosom 23 befinden, und deren Genfrequenz zweier Allele mit p und q bezeichnet wird, ist dabei in der Elterngeneration folgendes:

| Gene der Elterngeneration | | |
|---|---|---|
| | $A_1$ | $A_2$ |
| Häufigkeit | p | q |

Tabelle 2.1.: Genfrequenz von Eltern im Hardy-Weinberg Gleichgewicht

In der Kindergeneration ist das resultierende Verhältnis in folgender Tabelle dargestellt.

| Genotypen der Kindgeneration | | | |
|---|---|---|---|
| | $A_1A_1$ | $A_1A_2$ | $A_2A_2$ |
| Häufigkeit | $p^2$ | $2pq$ | $q^2$ |

Tabelle 2.2.: Genotypfrequenz in der Kindgeneration im Hardy-Weinberg Gleichgewicht

Voraussetzungen für diese Konstellation sind, dass die Genfrequenzen bei Männern und Frauen der Elterngeneration gleich sind und dass sich die Gene ohne Mutationen, Selektionen, Insertionen oder Deletionen während der Gametogamie, d.h. der Erstellung der Geschlechtszellen normal trennen. Basierend auf dem Aufteilunsverhältnis der Gene der Elterngeneration zu den Genotypen der Kindgeneration kann gefolgert werden, dass, für den Fall, dass sich eine Population im Hardy-Weinberg Gleichgewicht befindet, es nicht mehr als 50% von heterozygoten Genotypen geben kann.

### 2.1.3. Linkage Disequilibrium

Das *Linkage Disequilibrium* (kurz LD) bezeichnet im Allgemeinen die Tatsache, dass Allele zweier genetischer Loci nicht zufällig voneinander abhängig sind. Zur Definition des Grades von LD wird üblicherweise die folgende Formel herangezogen (siehe etwa Clayton (2005)):

$$r^2 = \frac{D^2}{p_1 p_2 q_1 q_2} \tag{2.3}$$

Die Variablen der Formel $p_1$, $p_2$, $q_1$ und $q_2$ geben dabei Auskunft über die Allelhäufigkeit der Allele der beiden SNPs und die Variable $D$ wird folgendermaßen berechnet $D = x_{11} - p_1 q_1$. Die Variable $x_{11}$ steht dabei für die Frequenz der Allelkombination $A_1\ B_1$. Die Allelhäufigkeit beruht dabei wiederum auf der Haplotyphäufigkeit an beiden SNPs. Zur Verdeutlichung soll das Beispiel in Tabelle 2.1.3 dienen mit SNP A = AT und SNP B = CT:

Überschreitet das Ergebnis einen Schwellenwert, so kann davon ausgegangen werden, dass sich zwei SNPs in LD zueinander befinden. Obwohl lange Zeit davon ausgegangen wurde, dass das Ergebnis der LD-Berechnung basierend auf oben genannter Gleichung von der genetischen Distanz zweier SNPs zueinander abhängig (siehe Meunier u. Eyre-Walker (2001)), kann man heute davon ausgehen, dass

| Haplotypen | Haplotyphäufigkeit | Allelkombinationen | Allelhäufigkeit |
|---|---|---|---|
| A1B1 | 0 | p1 = A1B1 + A1B2 | 1 |
| A1B2 | 1 | p2 = A2B1 + A2B2 | 1 |
| A2B1 | 1 | q1 = A1B1 + A2B1 | 1 |
| A2B2 | 0 | q2 = A1B2 + A2B2 | 1 |

Tabelle 2.3.: Haplotyp- und Allelhäufigkeit - A1 und B1 beziehen sich dabei auf das Allel an jeweils erster Stelle

es sich bei dieser Annahme um einen Trugschluss handelte (siehe Ziegler u. Koenig (2006, 195)).

Das LD-Maß hat daher vor allem Auswirkungen auf die Größe einer zu untersuchenden Population sowie auf die Qualität von genetischen Markern. So kann LD als Qualitätskriterium zur Auswahl genetischer Marker dienen, da ein hohes LD-Maß ebenfalls ein hohes Maß an Redundanz zwischen genetischen Markern ausdrückt. Das heisst wiederum, dass eine kleine Menge an Markern innerhalb einer genetischen Region ausreicht, um weitere Polymorphismen vorherzusagen vorausgesetzt es liegen hohe LD-Maße in einer genetischen Region vor.

Im Falle dieses Projektes wurden LD-Werte aller zur Verfügung stehender SNPs der KORA-Studie (Kooperative Gesundheitsforschung in der Region Augsburg) vorab mit Hilfe des Programmes PLink berechnet und ausgewertet, um so die Datenmenge, die in den späteren Clusteringprozess einfließt, zu reduzieren. So wurden nur die SNPs in die Berechnung der Cluster einbezogen, die zum einen nicht im LD zueinander waren sowie das Hardy-Weinberg Gleichgewicht erfüllen und eine bestimmte *minor-allele frequency* aufweisen. Besonders die Überprüfung auf HWE ist hierbei besonders hervorzuheben, da in der Phase der Genotypisierung, also der Prozess der Extraktion von SNP-Informationen aus DNA-Proben, Fehler auftreten können, welche wiederum an Hand des HWE überprüft werden können. So kann neben menschlichen Fehlern zum Beispiel die DNA-Konzentration zu gering sein, was in falschen SNP-Daten resultieren kann.

## 2.2. Digitale Formate zur Speicherung genetischer Informationen

Neben der Erkärung genetischer Grundbegriffe steht auch die Beschreibung einiger wichtiger Datenformate zur Speicherung genetischer Informationen im Mittelpunkt dieses Kapitels. Im Folgenden wird daher ein Überblick über die Speicherformate genetischer Daten sowie deren Vor- bzw. Nachteile gegeben.

Derzeit gibt es einige weit verbreitete Datenformate für die Speicherung genetischer Informationen, wie zum Beispiel das PED-Format und das damit in Verbindung stehende MAP-Format. Leider hängen diese Dateiformate stark von verschiedenen Applikationen ab, was dazu führt, dass es derzeit keinen einheitlichen Standard zur Speicherung und zum Austauch genetischer Daten gibt. Dazu kommt erschwerend, dass die genetische Forschung bzw. Bioinformatik auf diesem Gebiet sich ständig weiterentwickelt, weswegen es sehr schwer ist, ein einheitliches Datenformat zu definieren, da Fortschritte in der bioinformatischen Genanalyse neue oder erweiterte Datenformate benötigen. Dies ist vor allem dann zu beobachten, wenn Forscherteams neue Algorithmen entwickeln und die Ergebnisse in einem für sie passenden Dateiformat speichern, was sehr oft zu proprietären Datenformaten führt deren Auswertung andere Forscher vor eine zusätzliche Herausforderung stellt. Im Folgenden wird trotzdem auf die am wichtige und weit verbreitete Datenformate eingegangen, ihr Einsatz erörtert und ihre jeweiligen Unterschiede zueinander aufgezeigt.

### 2.2.1. Das PED- und MAP-Format

Ein wichtiges und oft eingesetztes bzw. weit verbreitetes Datenformat zur Speicherung und Übermittlung von genetischen Daten ist das Pedigree-Format oder kurz PED (siehe http://www.broadinstitute.org/mpg/tagger/faq.html). Wie der Name des Formats schon vermuten lässt, dient dieses Dateiformat hauptsächlich zur Speicherung von genetischen Informationen im Zusammenhang mit Familienstammbäu-

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 |
| 1 | 2 | 0 | 0 | 2 |
| 1 | 3 | 0 | 0 | 1 |
| 1 | 4 | 1 | 2 | 2 |
| 1 | 5 | 3 | 4 | 2 |
| 1 | 6 | 3 | 4 | 1 |

Tabelle 2.4.: Grundlegende PED-Datei

men bzw. zur Speicherung von Beziehungen zwischen Individuen. Dieses Datenformat selbst bietet allerdings wiederum verschiedene Möglichkeiten zur Speicherung bzw. welche Daten gespeichert werden. Das heisst, dass selbst ein weit verbreitetes Datenformat wie das PED-Format eigentlich kein wirklich standardisiertes Format ist wie im Folgenden zu sehen sein wird.

Was allerdings alle Möglichkeiten zur Speicherung im PED-Format gemeinsam haben ist, dass die Daten eines jeden Individuums in einer eigenen Zeile gespeichert werden, von denen die jeweils ersten fünf Spalten fest vorgegeben sind. Sprich die Minimalversion des PED-Formats setzt folgende fünf Spalten voraus:

**Familie Individuum Vater Mutter Geschlecht**

Als Beispiel für eine PED-Datei im Minimalformat soll Tabelle 2.4 dienen.

In dieser Tabelle steht wie oben beschrieben die erste Spalte für die Familien-ID, die zweite für die ID des Individuums, dessen Daten in dieser Zeile beschrieben werden, die dritte Spalte für die ID des Vaters und so weiter. Das Geschlecht ist meistens mit 1 für männlich und 2 für weiblich kodiert. Dabei spielt es allerdings keine Rolle wieviele Familien in einer Datei gespeichert werden. Da dieses minimale Datenformat jedoch keine genetischen Daten beinhaltet, sondern lediglich die Zusammenhänge zwischen einzelnen Individuen speichert, kann dieses Format um Gendaten erweitert werden. Daher können folgende Spalten angefügt werden:

**Krankheitsstatus Phänotyp Genotypen**

Diese Erweiterung ist jedoch etwas komplizierter als die Grundform der PED-Datei. Die Möglichkeiten dieser Erweiterung sind nämlich abhängig von der Appli-

kation, die die Daten auswerten soll. So erwartet zum Beispiel das Programm PLink (Purcell u. a. (2007)) in der sechsten Spalte den Phänotyp während Merlin (Abecasis u. a. (2002)) in der sechsten Spalte den Krankheitsstatus und in der siebten Spalte den Phänotyp erwartet. Im Falle von Merlin bezieht sich der Krankheitsstatus auf einen der folgenden Werte:

- U oder 1 für nicht betroffen
- A oder 2 für betroffen
- X oder 0 für unbekannt.

Im Falle von PLink hingegen kann der Krankheitsstatus durch einen der folgenden Werte kodiert werden:

- 1 für nicht betroffen
- 2 betroffen
- -9 oder 0 für unbekannt

Auch wenn die numerischen Werte für die Kodierung des Krankheitsstatuses zum Großteil übereinstimmen, gibt es doch unter Umständen schwerwiegende Differenzen. Zum einen können für Merlin die Statuswerte als Buchstaben kodiert werden und zum anderen interpretiert PLink -9 als einen nicht bekannten Krankheitsstatus.

Die Anzahl der Genotypen hingegen ist bei beiden eben erwähnten Applikationen, bis auf den zur Verfügung stehenden Speicherplatz, unbegrenzt. Die einzige Voraussetzung für die Speicherung der Genotypen in der PED-Datei ist, dass jeweils zwei Allele nur durch ein einzelnes Leerzeichen getrennt werden, sprich paarweise auftreten, wohingegen alle anderen Daten sowei die Allelpaare durch Tabulatoren getrennt werden. Allerdings unterscheiden sich beide Anwendungen wiederum im Hinblick auf die Kodierung von Genotypen bzw. von fehlenden, d.h. nicht ermittelbaren, Genotypen. Während PLink die „0“ bzw. die „-9“ als Zeichen für fehlende Genotypen und fehlende Angaben zum Krankheitsstatus interpretiert, müssen

fehlende Daten in Merlin mit Hilfe von „x“ angegeben werden. Für beide Anwendungen gilt allerdings, dass sobald ein Allel eines Allelpaares unbekannt ist, das zweite auch unbekannt sein muss, das heisst, dass nur komplett gültige Allelpaare zugelassen sind bzw. berechnet werden.

Um die Genotypdaten auswerten und interpretieren zu können benötigen beide Applikationen zusätzlich eine MAP-Datei, die im einfachsten Falle in folgendem Format vorliegen muss:

**Chromosom ID des Markers Position in Basenpaaren**

Die Daten müssen dabei in Spalten vorliegen. Desweiteren entspricht jede Zeile in der MAP-Datei einem genetischen Marker (SNP) in der PED-Datei. Nur falls diese Zuordnung von MAP-Datei zu PED-Datei 100% gelingt, können beide Programme die Daten fehlerfrei auswerten. Allerdings gibt es noch ein erweitertes Format für MAP-Dateien, das folgendermaßen aussieht:

**Chromosom ID des Markers Genetische Distanz in Morgan Position in Basenpaaren**

Diese zweite und um die genetische Distanz erweiterte Version ist allerdings PLink-spezifisch und wird von Merlin nicht unterstützt. Allerdings gibt es auch im Falle von Merlin eine erweiterte Fassung der MAP-Datei, die folgendermaßen aussieht:

**Chromosom ID des Markers Position in Basenpaaren Position bei Frauen Position bei Männern**

Wie man in obiger Darstellung erkennen kann, wurde hier das Standardformat um geschlechterspezifische Positionen der Marker ergänzt.

Zusätzlich zu der PED- und MAP-Datei benötigt Merlin noch eine dritte, eine sogenannte DAT-Datei, zur Interpretation genetischer Daten. Diese Datei hat den folgenden Aufbau:

**A <Krankheit>**
**T <quantitative Charakteristik>**
**M Marker 1**
**M Marker 2**

...

In diesem Dateiformat, das jeweils einen Eintrag pro Zeile erwartet, stehen A für eine mögliche Krankheit, die dann im Anschluss angegeben werden kann, T für quantitative Charakteristika einer Krankheit, C für eine Kovariate und M für einen Marker. Wie oben ersichtlich werden alle Buchstaben (A, C, M oder T) von einer Bezeichnung gefolgt, die keine Leerzeichen enthalten darf. Der Aufbau der DAT-Datei beschreibt die Daten bzw. Spalten der gegebenen PED-Datei, das heisst eine Zeile beschreibt eine Spalte ausser den Spalten mit Familien- oder Individuen-IDs.

Wie man unschwer aus der bisherigen Beschreibung des PED-Formats und den dazugehörigen sekundären Formaten (MAP und DAT) erkennen kann, ist die Auswertung und der Datenaustausch genetischer Daten alles andere als einheitlich. Auch wenn beide Anwedungen, PLink und Merlin, prinzipiell dieselben Dateiformate voraussetzen, so gibt es dennoch etliche unter Umständen schwerwiegende Unterschiede zwischen beiden Auswerteprogrammen:

- Kodierung des Status „unbekannt“ in PED-Dateien
- Aufbau der erweiterten MAP-Datei
- Kodierung des Krankheitsstatuses mittels Buchstaben in Merlin

Diese Unterschiede erschweren den Austausch von Daten erheblich, da vor jeder Analyse die Daten überprüft und falls nötig geändert bzw. umkodiert werden müssen, was nicht nur unnötig Zeit kostet, sondern auch eine enorme Fehlerquelle darstellt. Da PED-Dateien unter Umständen äusserst groß werden können, ist dann eine Fehlersuche bei einer eventuell fehlerhaften bzw. einer auf Grund von Formatfehlern abgebrochenen Analyse, extrem zeitaufwendig und schwierig.

### 2.2.2. BVML - ein XML-basiertes Datenaustauschformat

Da, wie oben beschrieben, der Datenaustausch sogar im Falle eines prinzipiell gleichen Datenformats äusserst kompliziert ist, gab es Ansätze ein plattformunabhän-

giges und universelles Datenmodell zum Austausch genetischer Informationen zu entwickeln. Tyrelle u. King (2003) haben ein XML-basiertes Datenmodell entwickelt, das es erlaubt sowohl die eigentlichen genetischen Daten als auch Metadaten im Bezug auf Gendaten zu speichern und auszutauschen. Dieses Open source Datenmodell mit dem Namen *Biological Variation Markup Language* oder kurz BVML wurde entwickelt, um ein einheitliches Datenmodell zur Verfügung zu stellen, das es gewährleisten soll, Daten ohne Probleme auszutauschen, auszuwerten und zu durchsuchen.

Einer der großen Vorteile von BVML bzw. XML (siehe W3C (2008)) ist es, dass die gespeicherten Daten zum einen strukturiert vorliegen und zum anderen leicht zu lesen/verstehen sind. In ihrem Ansatz haben Tyrelle u. King (2003) nicht nur die tatsächlichen genetischen Daten berücksichtigt, sondern auch Metadaten, Sicherheit, Erweiterbarkeit, Austausch und Auswertung des Datenmodells und der Daten. Dem zu Grunde liegt die Annahme, dass es eine erweiterbare Grundmenge an Elementen gibt, die die Daten ausreichend strukturiert. Der folgende schematische Aufbau des Grundschemas zeigt die Kernelemente von BVML:

```
<bvml>
        <bvmlinfo>
        </bvmlinfo>
        <variants>
        </variants>
        <haplotypes>
        </haplotypes>
        <reports>
        </reports>
</bvml>
```

In diesem Kernschema von BVML ist das Element *bvmlinfo* dazu gedacht, essentielle Metadaten über die zu Grunde liegenden Gendaten zu speichern. Zu den Metadaten gehören hierbei vor allem der Genname, der Name der Datenbank, in der

die Sequenzdaten gefunden werden können, Links zu Referenzen, Datenbanken, etc. Diese sollen bei einer Auswertung dazu dienen, dem Wissenschaftler zusätzliche nützliche Informationen über die vorliegenden Daten zur Verfügung zu stellen, damit weitere Untersuchungen und Analysen geplant werden können. So ist es unter Umständen wichtig zu wissen, in welcher Datenbank die vorliegenden Gendaten gefunden wurden, um dort nach ähnlichen Sequenzen oder in Verbindung mit den vorliegenden Daten in Verbindung stehenden Informationen zu suchen. Das Element *variants* ist zur Speicherung von Effekten von Polymorphismen auf verschiedenen Leveln genetischer Daten gedacht. Zu diesen Leveln gehören mRNA, DNA, Proteine und Gennetzwerke. Zur Beschreibung von Genvariationen können hierbei auch Sequenzen in ein *variants*-Element eingefügt werden. Diese Sequenzen können desweiteren relativ zu anderen angegebenen Sequenzen positioniert werden, sprich die Lage innerhalb des Genoms kann relativ zu anderen Sequenzen angegeben werden. Das Element *reports* dient dazu anzugeben mit welchen Verfahren, Methoden und anderen genetischen Daten die angegebenen Polymorphismen validiert wurden. Jedes *reports*-Element wird dazu mit mindestens einem *variants*-Element verlinkt, um so die Beziehung zwischen Polymorphismus und Analyse herzustellen und so eine spätere Auswertung nicht nur der eigentlichen Gendaten, sondern auch der verwendeten Methoden und Algorithmen zu ermöglichen. Zur Auswertung von möglichen ermittelten Haplotypen dient das Element *haplotypes*, das die Daten eines Haplotyps, das heisst mehrere Allele, deren loci auf dem Genom miteinander in Verbindung stehen, beinhalten kann. Trotz der Vorgabe der oben erwähnten Elemente, ist BVML dahingehend konzipiert, dass es arbiträr erweiterbar ist und um weitere Elemente in eigenen XML-Namespaces ergänzt werden kann.

Ein wichtiger Punkt bei der Speicherung und dem Austausch genetischer Daten, ist die Sensibilität der Daten. Das heisst es kann ohne weiteres vorkommen, dass Patientendaten in einem BVML-Dokument gespeichert werden, die besonders vor dem Zugriff nicht befugter Personen geschützt werden müssen. Auch im Falle von rechtlich restriktierten Daten, wie beispielsweise bei proprietären Daten, spielt deren Sicherheit eine enorme Rolle. Dazu wurde bei der Konzipierung von BVML

eine mögliche Verschlüsselung der Daten an Hand der *XML-Encryption* des W3C (siehe W3C (2002)) berücksichtigt. Diese Erweiterung zu XML sieht vor, dass sowohl einzelne als auch mehrere Elemente eines XML-Dokumentes verschlüsselt werden können. Dies geschieht mittels eines *CypherData*-Elements, das wiederum ein *CypherValue*-Element beinhaltet, das den eigentlichen verschlüsselten Inhalt speichert. Um die kodierten Daten wieder zu entschlüsseln ist die Kenntnis eines Schlüssels notwendig. Desweiteren dient im Falle von BVML der Einsatz von öffentlichen Schlüsseln zur Verifizierung der Datenintegrität nach einem Datenaustausch. So kann ein Dokument mit Hilfe eines Schlüssels „signiert“ werden. Bei der erneuten Bearbeitung des Dokuments kann dann der Schlüssel und damit die Integrität des Dokuments auf Korrektheit überprüft werden, um sicherzustellen, dass nur intakte Daten bearbeitet und weiter analysiert werden.

Für eine bessere und effektivere Auswertung der Metadaten eines BVML-Dokuments kann ein BVML-Dokument ebenfalls RDF-Elemente beinhalten. Das *resource desription framework* (RDF) ist ein XML-basierter Standard des W3C zur Speicherung semantischer Zusammenhänge arbiträrer XML-Elemente und Teil der *semantic web-initiative* (siehe dazu W3C (2004)). Mit Hilfe von RDF ist es möglich, die Zusammenhänge von BVML-Elementen zueinander zu kodieren und so eine bessere Möglichkeit im Hinblick auf Information Retrieval von BVML-Metadaten zur Verfügung zu stellen. Vor allem im Hinblick auf einen möglichen Einsatz von RDF in Gendatenbanken ist dieser Ansatz interessant, da es so möglich wäre globale Metadaten zu Polymorphismen und anderen genetischen Daten abzufragen und so bislang unbekannte Zusammenhänge zwischen Einzeldaten zu erkennen.

Jedoch hat sich allerdings der hier vorgestellte XML-basierte Ansatz zur Speicherung und Übertragung genetischer Daten bisher nicht durchgesetzt. Gründe dafür gibt es sicherlich viele, man darf nämlich nicht unterschätzen, dass ein XML-Dokument im Vergleich zu einem reinen Textdokument ein vielfaches an Speicherplatz benötigt. Dies dürfte sicherlich der Hauptgrund dafür sein, dass BVML keine Relevanz in der bioinformatischen Praxis hat, da ohne weiteres reine Textdateien,

die Gendaten beinhalten, mehrere Gigabyte an Speicherplatz belegen. Wenn man bedenkt, dass zur Kodierung genetischer Information in Form von BVML auch noch Metadaten, Reports und natürlich die eigentlichen Elementtags gespeichert werden, so nehmen derart gespeicherte Gendaten selbstverständlich viel mehr Speicherplatz ein als reine Textdaten oder auch Binärdaten. Auch wenn der Preis für Festplattenspeicher immer weiter sinkt, dürfte dies dennoch einer der wichtigsten Gründe für die derzeitige Irrelevanz von BVML sein. Dazu kommt noch, dass die zeitaufwendige Auswertung von genetischen Massendaten für den Falle einer Auswertung von XML-Dokumenten sicherlich noch langsamer ablaufen dürfte. Eine Möglichkeit dieses Problem zu umgehen wäre der Einsatz binärer XML-Daten wie er vom W3C bereits in Planung ist (http://www.w3.org/XML/EXI/).

## 2.3. Beschreibung relevanter Bioinformatikapplikationen und Datenbanken

Neben speziellen Gen bzw. Bioinformatikdatenformaten spielen auch Gendatenbanken und Applikationen zur Analyse genetischer Daten eine wichtige Rolle. Deshalb soll das folgende Unterkapitel einen Überblick über wichtige Applikationen im Bereich der Bioinformatik und vor allem für die Untersuchung genetischer Daten bieten. Da für das spätere Clustering auf die Bioinformatikapplikation PLink zugegriffen wird, wird vor allem diese Anwendung in einem eigenen Unterkapitel detaillierter beschrieben. Eine weitere Applikation, die vor allem im Zusammenhang mit der Überprüfung von Clusteringergebnissen verwendet wurde, ist Bibliosphere der Firma Genomatix, das ebenfalls in einem eigenen Unterkapitel näher beschrieben wird. Da dieses Projekt in enger Zusammenarbeit mit Wissenschaftlern des Universitätsklinikums Regensburg entstanden ist, wird allerdings nur auf die Applikationen näher eingegangen, die dort standardmäßig zum Einsatz kommen, da im Rahmen dieses Projekts versucht wurde nur bereits vorhandene und bereits erfolgreich eingesetzte Software zu verwenden, um so eine möglichst homogene IT-

Infrastruktur zu schaffen. Insofern werden weitere wichtige Applikationen aus dem Bereich der Bioinformatik wie etwa HelixTree der Firma Golden Helix zur Datenverwaltung und Analyse bzw. Auswertung von genomweiten Assoziationsstudien oder Clustal W, ein Programm zum Sequenzalignment nicht näher beschrieben.

### 2.3.1. PLink

Das von Shaun Purcell entwickelte Toolset PLink (Purcell u. a. (2007)) ist eine Softwareapplikation, die es ermöglicht, Tests und Auswertungen genetischer Daten durchzuführen. Da die Datenmenge für Genanalysen ständig zunimmt und daher, nicht nur wie bisher einzelne Kandidatengene untersucht werden, sondern großangelegte Analysen, die sich über das komplette menschliche Genom erstrecken (*whole-genome association studies*) durchgeführt werden, ist PLink vor allem vor dem Hintergrund dieser Massendaten entwickelt worden. Es bietet daher nicht nur Auswerteverfahren, sondern auch Datenmanagementfunktionen, die es ermöglichen Gendaten, die üblicherweise im Textformat (siehe 2.2.1) vorliegen, in ein eigenes Binärformat zu überführen. Dies hat den Vorteil, dass zum einen die Datenmenge reduziert wird und daher die Daten schneller analysiert werden können. Neben der Analyse von Datensätzen im Hinblick auf genetische Assoziationen und des Datenmanagements, bietet PLink auch noch Funktionalitäten im Hinblick auf die Auswertung der Qualität der vorliegenden Daten und die Untersuchung der Daten auf seltene Genvariationen.

Wie bereits erwähnt wurde für PLink ein eigenes Binärformat entwickelt, das es nicht nur ermöglicht Daten zu „komprimieren“, sondern auch diese zu verwalten. So bietet PLink in dieser Hinsicht die Möglichkeit SNPs oder Individuen basierend auf bestimmten Kriterien, wie etwa der Position eines SNPs auf dem menschlichen Genom, aus den Binärdaten zu extrahieren. Desweiteren besteht auch die Möglichkeit zwei verschiedene Datensätze zu verbinden, vorausgesetzt die Daten liegen im selben Datenformat, dem PED-Format, vor. Dies ist vor allem dann interessant falls aus zwei getrennten Populationen, eine einzige zu untersuchende Population ge-

bildet werden soll, um Gemeinsamkeiten bzw. Unterschiede zwischen beiden Populationen zu ermitteln. Desweiteren bietet diese Funktionalität die Möglichkeit, Ergebnisse zu validieren, da nach der Kombination zweier Populationen, die Werte tatsächlicher informativer Mutationen nicht stark abweichen dürften im Vergleich zu den Ergebnissen bei der Untersuchung von jeweils nur einer Population.

Auch zur Evaluierung der Qualität von genotypisierten Daten bietet PLink viele Funktionen. So können etwa Allelfrequenzen berechnet werden und Tests auf das Hardy-Weinberg Gleichgewicht (siehe 2.1.2) durchgeführt werden. Desweiteren kann untersucht werden, ob das Fehlen von SNPs an einem bestimmten Locus auf Zufall basiert oder es wahrscheinlicher einem Fehler während der Genotypisierungsphase zugeschrieben werden kann. Auch während der Analyse von Gendaten untersucht PLink immer wieder die zu Grunde liegenden Daten auf mögliche Fehler, um das Ergebnis so wenig wie möglich zu verzerren.

Nicht nur die Evaluierung der Qualität der Daten ist eine Funktionalität von PLink, sondern, wie schon erwähnt, auch die Analyse von Populationen im Hinblick auf ihre Genotypen. Dazu bietet PLink drei Möglichkeiten basierend auf der Grundlage des durchschnittlichen Verhältnisses von Allelen, die die gleichen phänotypischen Effekte aufweisen. (*identical-by-state* IBS):

- Clustering
- Multidimensionale Skalierung der Populationsdaten
- Identifikation von Ausreissern innerhalb der Daten

Hierbei ist vor allem die Clusteringanalyse der Populationsdaten von großem Interesse. Als Algorithmus verwendet PLink ein hierarchisches Clusteringverfahren, das zur Ähnlichkeitsberechnung zwischen Objekten das *complete-linkage* Verfahren einsetzt. Dabei wird davon ausgegangen, dass jedes einzelne Individuum zu Beginn des Algorithmus einen eigenen Cluster darstellt. Danach werden iterativ Cluster kombiniert auf der Basis der Entfernung der am weitesten von einander entfernten Objekte zweier Cluster. Die Cluster, die sich am ähnlichsten sind, werden dann

zu einem einzigen Cluser verschmolzen. Dieser Algorithmus wird solange durchgeführt, bis entweder alle Objekte zufriedenstellend klassifiziert wurden oder ein anderes Abbruchkriterium wie zum Beispiel eine maximale Anzahl an Iterationen oder eine minimale Anzahl an Clustern erreicht wurde. Allerdings wird darauf geachtet, dass bestimmte vorab definierte Qualitätskriterien bzgl. der Datengrundlage eingehalten werden. Dazu zählen beispielsweise die Berücksichtigung der Anzahl von Fällen und Kontrollen innerhalb eines Clusters. So kann zum Beispiel angegeben werden, dass in jedem Cluster mit mehr als zwei Individuen mindestens ein Fall und mindestens eine Kontrolle auftritt, um so eine bessere Auswertegrundlage für Fall-Kontroll Studien zu erhalten. Andererseits können allerdings solche Einschränkungen unter Umständen, das Clusteringergebnis stark stratifizieren, da die Einteilung in Cluster nicht mehr nur auf der Ähnlichkeit zwischen Objekten bzw. Clustern beruht, sondern durch andere Faktoren beeinflusst wird.

### 2.3.2. Genomatix

Neben der zuallererst reinen bioinformatischen bzw. statistischen Anwendung PLink gibt es auch Anwendungen aus dem Bereich der Bioinformatik, die einen anderen Ansatz verfolgen wie etwa die Applikation Genomatix der Firma Genomatix Software GmbH. Auch wenn es sich bei Genomatix vorrangig um kein statistisches Analysetool im klassischen Sinne der statistischen Genetik bzw. aus dem Bereich der Bioinformatik handelt, so bietet es dennoch wichtige Funktionen zur Analyse von Gendaten. Genomatix dient vorrangig der Analyse von Regulationen von Genen und den Beziehungen von Genen zueinander. Dazu verwendet es Techniken des Text Minings im Zusammenhang mit Literatur aus den Bereichen Medizin und Biologie. Als Grundlage für das Text Mining dienen rund 15 Millionen wissenschaftliche Abstracts, die in der Pubmed-Datenbank gespeichert sind (Stand 2007 Genomatix (2007)). Jedoch wird im Fall von Genomatix nicht nur reines Text Mining zur Erkenntnisgewinnung eingesetzt, sondern die Ergebnisse des Text Minings werden mit fundierten Ergebnissen aus durchgeführten Experimenten angereichert.

So werden nicht nur Abstracts aus Pubmed mit in die Analyse einbezogen sondern auch unter anderem Geninformationen aus GeneOntology, UniGene und medizinische Termini aus MeSH (*medical subjects headings*), einem Thesaurus kontrollierten medizinischen Vokabulars. Der Grund dafür ist die Datenflut in Form von Publikationen in den Bereichen Medizin, Biologie und Bioinformatik. Laut Scherf u. a. (2005) werden mehrere Tausend Publikationen täglich der Onlineliteraturdatenbank MedLine, die von der *National Library of Medicine* betrieben wird, hinzugefügt. Dies hat zur Folge, dass es unmöglich ist ohne maschinelle Hilfe auf dem neuesten Stand in der biomedizinischen Forschung zu sein. Das ist der Ansatzpunkt von Genomatix, das versucht relevante Informationen von irrelevanten in der biomedizinischen Fachliteratur zu trennen.

Ein besonderes Problem beim Text Mining von biomedizinischer Literatur, vor allem von Fachliteratur aus dem Bereich Genetik, ist das Erkennen von biologischen Eigennamen wie Genbezeichnungen. Auch wenn das Erkennen von Eigennamen ein generelles Problem im Bereich Text Mining ist, so ist die Komplexität im Bereich der Genetik noch um einiges höher, da zum Beispiel äusserst schwer zu sagen ist, ob es sich bei „DREAM“ um das englische Wort für Traum handelt oder um das Gen, das in der gemeinen Fruchtfliege (drosophila melanogaster) auftritt und dort für Zelltod und Proteolyse verantwortlich ist. Dazu kommt noch, dass das Gen „DREAM“ nicht nur ambig ist, sondern auch mehrere Synonyme hat wie zum Beispiel „CG7863“ oder „STRICA“. Auch wenn die Disambiguierung an Hand von einschlägigen Onlinewörterbüchern wie etwa der *HUGO Gene Nomenclature* durchgeführt werden könnte, so ist diesbezüglich jedoch anzumerken, dass erstens nicht alle Terme in einem Wörterbuch auftreten müssen und zweitens entschieden werden muss, ob und falls ja in welchem Wörterbuch nach dem Term gesucht werden soll.

Nach dem Erkennen biologischer Terme und Entitäten in arbiträren und unstrukturierten Texten, ist es notwendig die Relationen zwischen einzelnen Genen zu identifizieren, um so komplexere Genanalysen zu ermöglichen. Dies kann zum einen basierend auf dem gemeinsamen Auftreten mehrere biologischer Terme in derselben

Textpassage und zum anderen basierend auf speziellen Syntaxregeln durchgeführt werden. Obwohl der erste Ansatz eher ungenaue Ergebnisse liefert hat er dennoch den Vorteil gegenüber dem zweiten Ansatz, dass er weniger arbeitsintensiv ist, da die Erstellung und Pflege von Syntaxregeln vor allem von spezialisierten Regeln im Bereich Biomedizin äusserst zeitintensiv ist.

Als letzter Schritt gilt es die Relationen zwischen biologischen Entitäten korrekt an Hand ihres funktionellen Kontexts im menschlichen Körper zu ermitteln. Das Problem hierbei ist, dass sich der funktionelle Kontext der Genrelationen von Objekt zu Objekt ändert, was eine generelle automatische Einteilung unmöglich macht. Hierfür können Referenzdatenbanken, die zum Großteil von Hand gepflegt werden, wie etwa GeneOntology (siehe 2.3.3) oder MeSH mit in den Prozess des Text Mining einbezogen werden. So können Genrelationen bzgl. des Auftretens bzw. der manuell erstellten Klassifizierung der einzelnen Gene mit großer Wahrscheinlichkeit erkannt und definiert werden.

Die so gewonnenen Daten können dann grafisch in einem Clusternetzwerk mittels der Software Bibliosphere der Firma Genomatix dargestellt und untersucht werden.

Wie in Bild 2.1 erkennbar werden Gene und deren Relationen, sprich die Signalpfade zwischen Genen, in Form eines Graphen dargestellt. Optional kann das Netzwerk auch in einer dreidimensional Darstellung (2.2) angezeigt werden, was neben der anderen Darstellungsart auch noch einige zusätzliche Optionen bietet.

So bietet die dreidimensionale Darstellung den Vorteil, dass Gene eines Clusters eigene „Wolken“ in der Darstellung bilden und es so leichter ist, zusammenhängende Gene zu identifizieren. Darüber hinaus werden sobald der Benutzer mit der Maus ein Gen auswählt, alle Gene, die nicht mit dem ausgewählten Gen in Verbindung stehen, ausgeblendet, so dass nur der Cluster angezeigt wird, der für das ausgewählte Gen relevant ist.

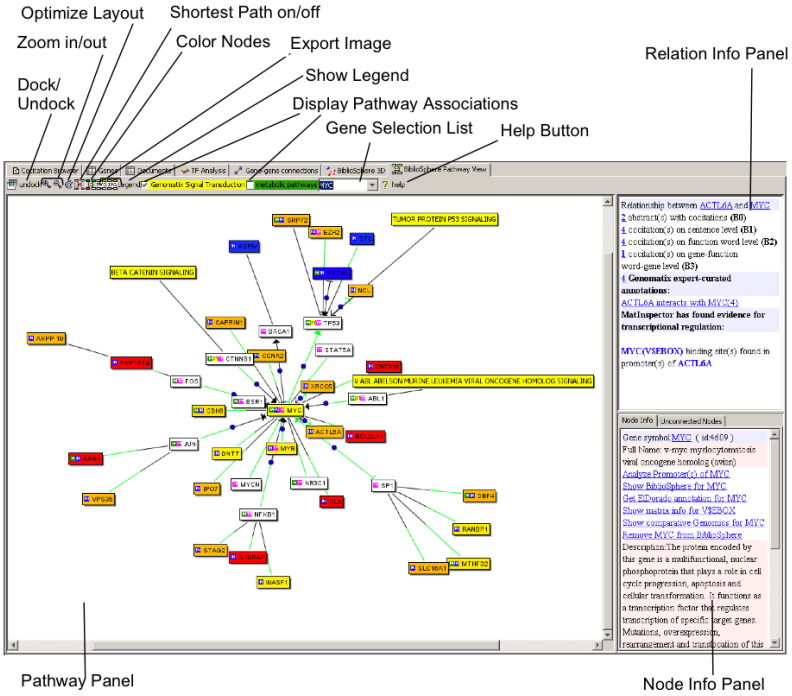



Abbildung 2.1.: Grafische Darstellung eines Gennetzwerks (Genomatix (2007))

### 2.3.3. GeneOntology

Neben Softwareprodukten, die zur Analyse, Bearbeitung, Test, etc. von genetischen Daten dienen, spielt selbstverständlich auch die Datenhaltung genetischer Daten eine große Rolle. Deshalb werden im Folgenden wichtige und oft benutzte Gendatenbanken näher beschrieben.

Da genetische Daten nicht nur zwischen einzelnen Forschergruppen ausgetauscht werden, sondern auch online offen und damit allen Interessenten zur Verfügung stehen, soll an dieser Stelle die bekannte Onlinedatenbank *GeneOntology* (http://www.geneontology.org) beschrieben werden. GeneOntology (GO) ist nicht nur interessant, da es sich um eine große Datenbank mit genetischen Informationen bzw. um die „the most widely used one “ (Backofen u. a. (2004)) handelt, sondern vor allem auch weil die genetischen Daten, wie der Name der Datenbank schon

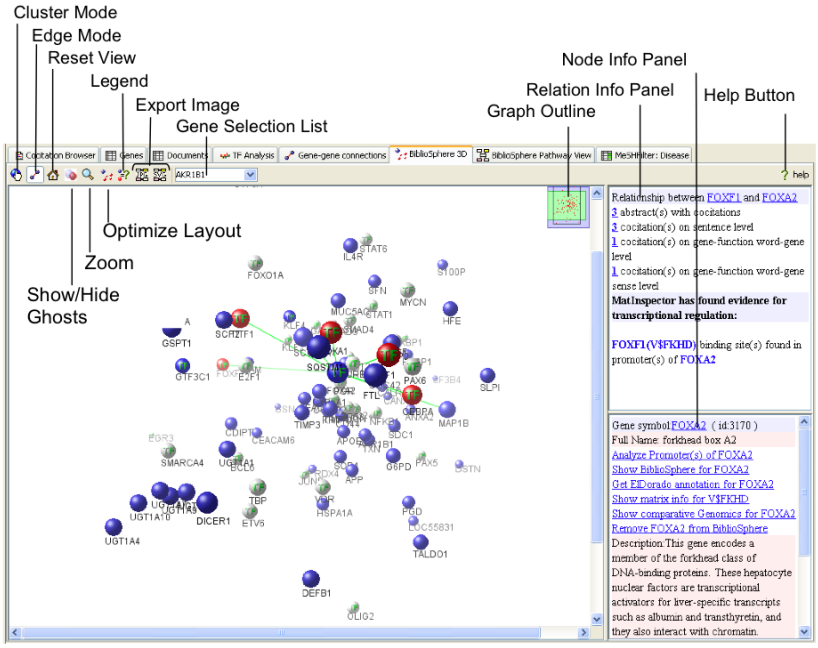



Abbildung 2.2.: Dreidimensionale grafische Darstellung eines Gennetzwerks (Genomatix (2007))

sagt, in Ontologien organisiert sind. Dieses Vorgehen, also das Speichern bzw. Bereitstellung von bioinformatischen Daten ist ein immer weiter verbreiteter Ansatz zur Analyse von Daten im Bereich der Bioinformatik (siehe dazu etwa Hartmann u. a. (2010)) In GO werden mit Hilfe eines kontrollierten Vokabulars Gene, deren Produkte und die Zusammenhänge zwischen Genen beschrieben. Dazu produziert GO „a controlled vocabulary that can be used for dynamic maintenance and interoperability between genome databases.“ (Ashburner u. a. (2000)). Und gerade in dieser „interoperability“ liegt die große Stärke der GO, da sie sich nicht wie andere Genomdatenbanken auf einen Bereich der Genetik beschränkt, sondern viele verschiedene Genetikdatenbank in Zusammenhang bringt.

Die GO wurde 1998 als Gemeinschaftsprojekt von *FlyBase*, *Saccharomyces Genome Database* und *Mouse Genome Database* begonnen und beinhaltet derzeit Da-

ten aus anerkannten Genomdatenbanken wie etwa SwissProt, EMBL, UniProt und andere. Ziel von GO war und ist es durch die Zusammenarbeit mehrer Genomdatenbanken und den Einsatz eines kontrollierten Vokabulars eine allgemein abfragbare Datengrundlage im Bereich Genetik zu schaffen, mit deren Hilfe es möglich sein soll, Daten aus mehreren Datenbanken unkompliziert zu ermitteln.

Grundsätzlich ist GO in folgende drei Teildatenbanken bzw. Teilontologien aufgeteilt:

- Biologische Prozesse
- Zellkomponenten
- Molekulare Funktionen

Hierbei beinhaltet die Teilontologie der biologischen Prozesse Terme, die eine „series of events accomplished by one or more ordered assemblies of molecular functions“ (Ashburner u. a. (2000)) darstellen. In der Teilontologie der molekularen Funktionen werden die Terme gespeichert, die die biochemische Aktivität eines Gens näher beschreiben. Diese Beschreibung beinhaltet allerdings nur die molekularen Funktionen ohne Angaben über Ort bzw. Zeitpunkt der tatsächlichen Funktion. Die letzte Teilontologie der Zellkomponenten beinhaltet Daten über den Ort innerhalb der Zelle, an dem ein bestimmtes Genprodukt aktiv ist.

Alle drei Teildatenbanken werden als azyklische Graphen dargestellt, in denen Terme miteinander mit Hilfe von semantischen Beziehungen hierarchisch verknüpft werden. Dabei gibt es nicht nur die aus dem Bereich des Semantic Web (siehe etwa W3C (2004)) bekannten *is_a* und *part_of*-Beziehungen, sondern die Beziehungsterme wurden um für Genetik und Biologie typische Beziehungen wie *regulates*, *positively_regulates* und *negatively_regulates* erweitert. Die einzelnen Terme, die die Objekte wie Gene, Genprodukte, etc. beschreiben, werden innerhalb der GO als numerische Zeichenfolge bestehend aus 7 Ziffern mit folgendem Muster dargestellt: *GO:nnnnnnn*. Diese Terme können dann hierarchisch in Verbindung gesetzt werden wie das Beispiel in Listing 2.1 zeigt.

Listing 2.1: Beispiel einer is_a-Beziehung (www.geneontology.org/GO.doc.shtml)

```
GO:0043232 : intracellular non-membrane-bound organelle
[i] GO:0005694 : chromosome
---[i] GO:0000228 : nuclear chromosome
```

Der Buchstabe in eckigen Klammern verweist dabei auf die Art der Relation zwischen den einzelnen Termen. In obigem Beispiel steht daher das *[i]* für eine is_a-Beziehung zwischen den Termen GO:0043232, GO:0005694 und GO:0000228. Auf diese Art und Weise lassen sich nicht nur die Daten hierarchisch anordnen sonder auch semantisch zueinander in Beziehung setzen, was wiederum die Auswertemöglichkeiten stark erhöht vor allem da nicht nur die Daten einer einzelnen Datenbank semantisch annotiert wurden, sondern die Daten aus mehreren Datenbanken semantisch aufbereitet wurden und immer noch werden. Als visuelle Darstellung des schematischen Aufbaus soll Bild 2.3 dienen.

Wie in Bild 2.3 zu sehen ist, ist *nucleic acid binding* eine Art molekularer Funktion und wie in Bild 2.4 ersichtlich hat dieser Term (GO:0003676) eine *is_a* - Beziehung zu *binding*. Diese Daten wurden mit Hilfe des AmiGO-Onlinetools der GO (http://amigo.geneontology.org/cgi-bin/amigo/go.cgi) ermittelt. Wie in 2.4 ebenfalls zu sehen ist, bietet GO unmittelbar bei den Suchergebnissen Referenzen auf externe Daten an, um so einen kompletteren Überblick zu bieten als er nur an Hand der GO-Daten möglich wäre.

Um die eigentlichen Inhalte mit anderen Gendatenbanken wie zum Beispiel SwissProt, EMBL, etc. zu verlinken besitzt jedes GO-Objekt einen Link auf eine externe Datenbank. Auf diese Art und Weise ist es möglich, Daten, die sich vor allem im Bereich der Genetik ständig verändern, zu integrieren ohne dabei interne Datenstrukturen zu ändern oder große inhaltliche Änderungen vorzunehmen. Daher ist ein stetes Wachstum ohne komplexe Änderungen an den Daten möglich. Da nicht nur menschliche Gene und deren Produkte und Zusammenwirken in der GO gespeichert werden, sondern auch tierische Daten wie etwa Drosophila-Daten oder

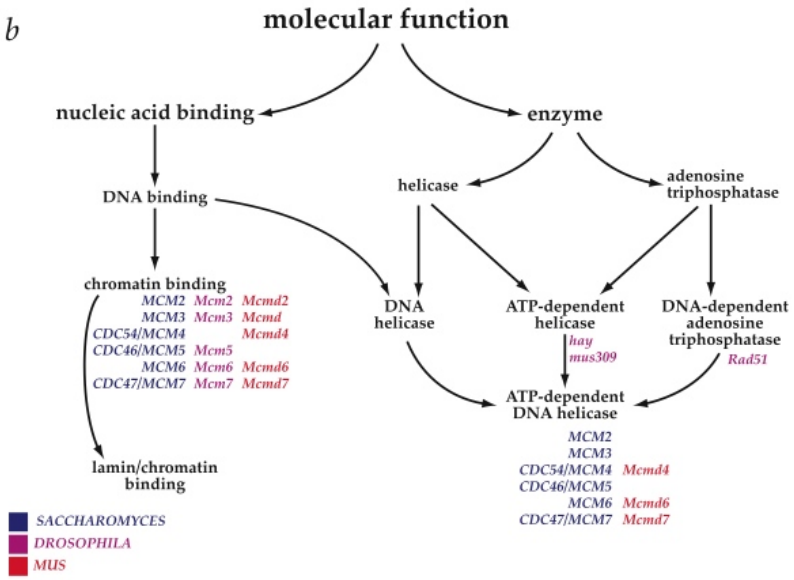



Abbildung 2.3.: Beispiel des schematischen Aufbaus der Ontologie der molekularen Funktionen GO:2000

Gendaten von Hefebakterien ist es möglich spezienübergreifend Analysen zu erstellen, um somit unter Umständen bisher verborgene Zusammenhänge und Gemeinsamkeiten im Rahmen der vergleichenden Genomik zur besseren Entschlüsselung und Annotierung der menschlichen Gene aufzudecken.

### 2.3.4. UniProt

Eine weitere viel benutzte Quelle für genetische Informationen ist die *Universal Protein Resource* (UniProt - http://www.uniprot.org/), die vom European Bioinformatics Institute (EBI), der Protein Information Resource und dem Swiss Institute of Bioinformatics betrieben wird. Als Datengrundlage für alle frei zugänglichen Teildatenbanken (siehe unten) dienen standardisierte Proteindaten, die, genauso wie im Falle der GeneOntology, aus mehreren einzelnen und von einander getrennten Da-

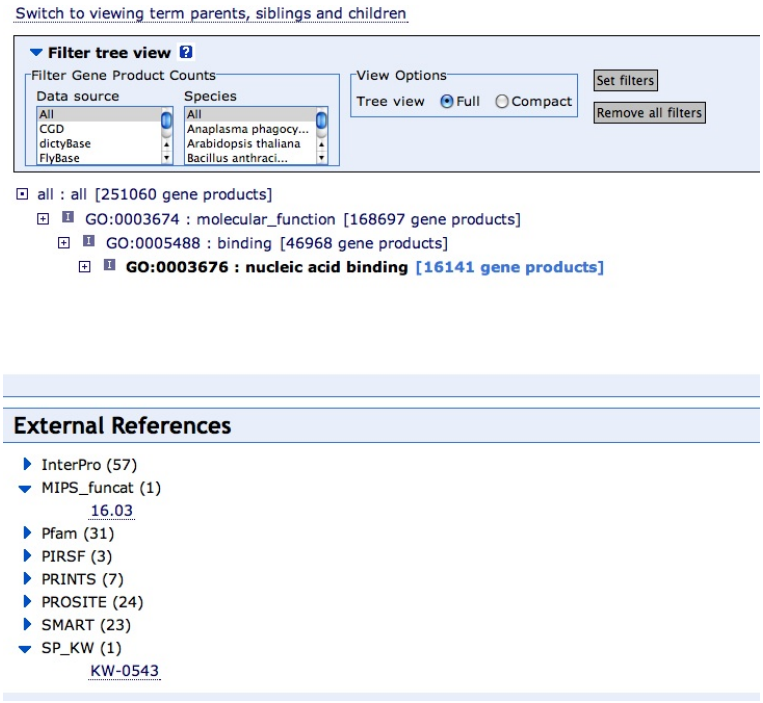



Abbildung 2.4.: Einordnung von *nucleic acid binding* in die Hierarchie der molekularen Funktionen (http://amigo.geneontology.org/cgi-bin/amigo/term-details.cgi?term=GO:0003676session_id=9787amigo1266653942)

tenbanken erstellt bzw. referenziert werden. Ähnlich wie GeneOntology ist UniProt in mehrere Teildatenbanken aufgeteilt:

- UniProt Knowledgebase (UniProtKB)
- UniProt Archive (UniParc)
- UniProt Reference Clusters (UniRef)

Die UniProt Knowledgebase (UniProtKB) selbst besteht wiederum aus zwei Teildatenbanken, nämlich der UniProtKB/Swiss-Prot und der UniProtKB/TrEMBL.

UniProtKB/Swiss-Prot ist eine Wissensquelle, die von Experten an Hand von aktuellen Erkenntnissen aus der Fachliteratur und Ergebnissen von Computeranalysen manuell annotiert wird. Als Annotationen gelten dabei unter anderem Proteinfunktionen und -strukturen, Verbindungen zu Krankheiten, etc. Neben den Annotation stehen dem interessierten Wissenschaftler auch Querverweise zur Verfügung, die Auskunft darüber geben, wo die entsprechenden Informationen gefunden wurden und Verweise auf weitere nützliche Datenbanken.

Ziel der UniProt Reference Clusters (UniRef) ist es zum einen die Suche nach Sequenzähnlichkeiten und zum anderen das Verbinden von Sequenzen zu vereinfachen. Dazu werden ähnliche Gensequenzen zu Clustern zusammengefasst, um so das Problem redundanter Sequenzdaten zu verhindern. Diese Cluster können dann für weiterführende Analysen verwendet werden.

Das UniProt Archive (UniParc) bietet ein nicht redundates Archiv von Proteinequenzen aus verschiedenen öffentlich zugänglichen Datenbanken wie EMBL, Swiss-Prot, WormBase, etc. Dazu werden Verweise auf die Ursprungsdatenbanken erzeugt, damit es später möglich ist, ausgehend von den Proteinsequenzen, weitere Informationen zu erhalten.

### 2.3.5. GenBank

Ähnlich wie das European Bioinformatics Institute (EBI) mit Uniprot eine Gen- und Sequenzdatenbank betreibt, so wird auch von den amerikanischen National Institutes of Health (NIH) eine eigene Genetikdatenbank verwaltet und gepflegt. Diese *GenBank* (http://www.ncbi.nlm.nih.gov/Genbank/) beinhaltet derzeit 106,533,156,756 Basen und 108,431,692 einzelne DNA-Sequenzen (Stand August 2009) und wird alle zwei Monate vom NHI in einer neuen, aktualisierten Version freigegeben. Für Wissenschaftler und Forschungseinrichtigungen besteht darüber hinaus die Möglichkeit entschlüsselte Sequenzen einzureichen.

Die in *GenBank* enthaltenen Sequenzdaten können entweder über ein Webformular oder direkt über Programmierschnittstellen der NCBI e-utilities abgefragt wer-

den. Darüber hinaus besteht noch die Möglichkeit sämtliche in der Datenbank enthaltenen Sequenzdaten in Form von ASCII-Dateien zusammen mit Metainformationen wie Autor, Schlüsselwörtern, Zitierungen, etc. von der Website der *GenBank* mit einem Gesamtumfang von ca. 431 GB herunterzuladen. Es stehen allerdings nicht nur menschliche DNA-Sequenzen zur Verfügung, sondern auch andere Formen von DNA wie etwa von Bakterien, Viren, etc.

Um eine möglichst umfassende Datengrundlage für genetische Forschung zu bieten, besteht eine enge Kooperation zwischen dem amerikanischen NIH, der DNA DataBank of Japan (DDBJ) und dem European Institute of Bioinformatics. Hierbei werden täglich Daten untereinander ausgetauscht, um so die jeweiligen Datenbank auf einem möglichst aktuellen Stand zu halten, was vor allem auf Grund der uneingeschränkten freien Nutzung der Datenbank ein sehr wichtiger und positiver Aspekt ist.

# Kapitel 3.

# Informationstechnologische Aspekte im Zusammenhang mit klinischen Studien

Nach der Einführung in allgemeine Aspekte der Genetik bzw. Bioinformatik und der Beschreibung wichtiger Softwareprodukte, Dateiformate und Datenbanken aus diesem Bereich, soll dieses Kapitel dazu dienen, die Durchführung sowie Datenspeicherung im Zusammenhang mit klinischen Studien zu beschreiben. Wie im Einleitungskapitel bereits erwähnt dienen dazu vor allem Studien, die an der Klinik und Poliklinik für Innere Medizin II bzw. der Arbeitsgruppe um Prof. Dr. Christian Hengstenberg durchgeführt werden wie etwa die HIFAM-Studie (Hengstenberg (2006) oder Broeckel u. a. (2002)). Desweiteren sollen vor allem allgemeine IT-Aspekte wie die Speicherung und Auswertung von in klinischen Studien erhobenen Daten, der Workflow im Zusammenhang mit Datenerhebung und -analyse sowie Aspekte der Benutzerschnittstelle näher erläutert werden. Die Erläuterung dieser allgemeinen theoretischen Punkte dient als Ausgangspunkt für die nähere Ausführung der praktischen Aspekte dieses Vorhabens, nämlich der Entwicklung eines einheitlichen Softwareframeworks zur Datenverwaltung.

In diesem Zusammenhang wird vor allem der Speicherung und Verwaltung der Daten, die im Laufe einer klinischen Studie gesammelt werden, ein hoher Stellenwert eingeräumt, da die Güte einer klinischen Studien von der Qualität der Datenverwaltung abhängt. Hierbei ist wichtig zu erwähnen, dass die Qualität einer Studie von der Art und Weise wie die Daten gespeichert werden abhängt, also der zu Grun-

de liegenden Datenbank bzw. des Datenmodells, da dies sowohl die Datenspeicherung selbst als auch das Suchen nach Daten bzw. das Aktualisieren von Daten, die Integrität der Daten und die Auswertung der Studiendaten direkt beeinflusst.

Ein damit verbundener Aspekt ist der Punkt der Benutzerschnittstellen, da diese einen wichtigen Beitrag zur sicheren und korrekten Datenspeicherung und -auswertung liefern können und sollen. So ist es wichtig, dass die gesamte Datenhaltung und -analyse durch eine intuitive Benutzerschnittstelle unterstützt wird, die dem Benutzer jederzeit wichtiges Feedback gibt und die eigentliche Datenstruktur auf Datenbankebene verbirgt, um so direkte Datenmanipulationen zu vermeiden. Dieser indirekte Zugriff auf die zu Grunde liegenden Daten ist insofern von enormer Wichtigkeit, da es ebenfalls zur Aufgabe einer Benutzerschnittstelle gehört nur Operationen zuzulassen, die die Daten nicht korrumpieren, was bei einem direkten Datenzugriff auf Datenbankebene nicht möglich ist.

In den folgenden Abschnitten wird näher auf die eben erwähnten Aspekte eingegangen, um so eine Grundlage für die praktische Umsetzung eines Softwareframeworks zur Datenverwaltung in klinischen Studien aufzubauen. So werden wichtige Aspekte wie die allgemeinen Ziele medizinischer Dokumentation, die Anforderungen an ein medizinisches Informationsframework sowie die Beschreibung des Workflows in klinischen Studien bzw. des Datenmanagements in klinischen Studien weiter ausgeführt.

## 3.1. Ziele und Probleme medizinischer Dokumentation im Rahmen klinischer Studien

Da die Durchführung und vor allem die Datenspeicherung und -auswertung im Rahmen medizinischer Studien einen wichtigen theoretischen Aspekt für die Entwicklung eines bioinformatischen bzw. medizinischen Softwareframeworks darstellen, soll im nächsten Teil ein Überblick über die allgemeinen Zwecke, Ziele und Voraussetzungen der medizinischen Dokumentation, wie sie auch in klinischen Studi-

en eingesetzt wird, gegeben werden. Per definitionem beschäftigt sich die Dokumentation in der Medizin „mit dem Erfassen, Erschließen, Speichern, Ordnen und Wiedergewinnen von medizinischen Information“ (Zaiss u. a. (2002, 46)). Schon im antiken Ägypten wurden „Patientenakten“ und Behandlungsmethoden in Form von Hieroglyphen dokumentiert, da man schon damals wusste, dass eine effektive und erfolgreiche Behandlung von Patienten und deren Leiden sehr stark von bereits bestehendem Wissen abhängt. Ebenso ist die Schaffung neuen Wissens von der Dokumentation von Krankengeschichten abhängig, weshalb auch antike Mediziner wie Hippokrates die Erstellung von Krankengeschichten forderten. Auch heute noch setzt die medizinische Dokumentation ein sogenanntes *minimum basic data set* (Basisdokumentation) eines Patienten voraus. Dazu zählen Informationen wie etwa die ID eines anonymisierten Patienten, dessen Geschlecht und Alter, Diagnosen, Einsatz medizinischer Maßnahmen etc. Dies spiegelt jedoch eine eher minimale Datengrundlage für den klinischen Alltag wider. Werden allerdings Daten für medizinische Studien erhoben, so wird diese horizontale Dokumentation, d.h. eine Dokumentation weniger Merkmale für viele Patienten, in eine vertikale Dokumentation, sprich die Dokumentation vieler Merkmale für wenige ausgewählte Patienten, transformiert.

Verallgemeinernd kann das Ziel der medizinischen Dokumentation nicht nur in klinischen Studien als die Aufgabe „berechtigten Personen alle relevanten (und nur die relevanten) Informationen zu einem oder mehreren Patienten und ihrer Behandlung bereitzustellen, und zwar zum richtigen Zeitpunkt, am richtigen Ort und in der richtigen Form“ (Leiner u. a., 2003, 4) definiert werden. Diese allgemeine Definition muss allerdings im Zusammenhang mit medizinischen Studien um die leichte Möglichkeit der Dateneingabe sowie der leichten Datenabfrage erweitert werden. Vor allem die Datenbereitstellung für berechtigte Personen ist hierbei ein äusserst wichtiger Punkt, da es sich bei den erhobenen Daten im klinischen Kontext um höchstsensible Patientendaten handelt, deren Schutz vor unberechtigtem Zugriff aus datenschutzrechtlichen und auch ethischen Gründen höchste Priorität genießen muss.

Die Bereitstellung der wichtigen und vor allem richtigen Informationen an einem bestimmten Ort setzt zwei wichtige Aspekte voraus. Erstens muss sichergestellt werden, dass die Daten an einem zentralen, für alle berechtigten Personen erreichbaren Ort gespeichert werden und zweitens dass die Daten ständig auf ihre Plausibilität geprüft werden, um die Qualität der Daten sicherzustellen. Daraus ergeben sich weitere wichtige Punkte:

- Sicherstellen der Zugriffsberechtigungen und dadurch sichern der sensiblen Daten
- Speicherung der Daten, so dass sie für mehrere berechtigte Personen gleichzeitig zugänglich sind
- die Notwendigkeit die Qualität und Plausibilität der Daten schon während der Datenerhebung zu überprüfen; vor allem während der Dateneingabe muss die Qualität der Daten sichergestellt werden
- hohe Genauigkeit der errechneten Daten, da sich aus den eingegebenen Grunddaten Werte wie etwa der *body mass index* ableiten und diese dann analysiert werden
- Einsatz und Befolgung von Standards vor allem im Bereich multizentrischer Studien zum Datenaustausch zwischen einzelnen Forschergruppen

Insbesondere der Gesichtspunkt der Qualitätssicherung stellt hohe Ansprüche sowohl an die Erstellung von Fragebögen zur Datenerhebung, damit möglichst alle Daten plausibel erhoben werden können, als auch an die Anwendung, die benutzt wird, um die Daten zu speichern. Da die Auswertung der Daten nur so gut sein kann wie die Rohdaten es zulassen, sollte die meiste Zeit bei der Planung eines Dokumentationssystems in diese beiden Punkte investiert werden. Vor allem im Bezug auf wissenschaftliche Studien im Bereich der Genetik ist es essentiell, dass die

Patientendaten so genau wie möglich erhoben werden und dann genauestens analysiert werden, da diese Daten später für mögliche Genotypisierung der Patienten-DNS herangezogen werden. Sollten daher die Ausgangsdaten fehlerhaft erhoben oder berechnet worden sein, könnte dies zu mangelhaften Gendaten führen, da unter Umständen nicht alle relevanten Patienten genotypisiert werden oder die DNS von Patienten genotypisiert wird, die keine interessanten Merkmale auf die zu untersuchende Ausgangsfrage aufweisen. Insofern kann hier festgehalten werden, dass das Ergebnis genetischer Studien immanent von der Qualität der Datenerhebung und Rohdatenanalyse abhängt. Desweiteren sollte vor allem im Falle multizentrischer Studien darauf geachtet werden, ein einheitliches, kontrolliertes Vokabular zur Speicherung von Patientendaten zu verwenden. Dafür bieten sich zum Beispiel *logical observations identifiers names and codes*, kurz LOINC (siehe McDonald u. a. (2009)), oder die *Systematized NOmenclature of MEDicine*, kurz SNOMED (siehe IHTSDO (2009)), an. Diese Standards bieten ein kontrolliertes Vokabular zur Beschreibung von medizinischen Sachverhalten und sollten zur Vereinfachung des Datenaustausches zwischen Forschergruppen eingesetzt werden. Auch der tatsächliche Datenaustausch sollte zu Gunsten der Qualität medizinischer Studien auf bereits vorhandenen Standards beruhen. So stellt zum Beispiel das *health level 7* (HL7) (siehe Geßner (2006)) ein Datenformat zum Austausch textueller medizinischer Informationen dar. Desweiteren sollte auch auf standardisierte Skalen zur Angabe ordinaler Attribute zurückgegriffen werden. So bietet etwa die NYHA-Klassifikation, die von der New York Heart Association erstellt wurde, eine vier-stufige Skala zur Definition der Schwere von Herzkrankheiten wie etwa der Herzinsuffizienz. Werden die Attribute mit Hilfe solcher standardisierter Skalen gespeichert, so erleichtert dies nicht nur eine konstante Auswertung, sondern auch den Austausch der Daten mit weiteren Forschungseinrichtungen.

Ein nicht zu unterschätzendes Problem bei der Durchführung medizinischer Studien ist jedoch, dass eine schlechte Dokumentation der Daten die spätere Auswertung unnötig erschwert, da unter Umständen der Sinn und Zweck bzw. die Wertebereiche einzelner Variablen nicht mehr klar sind.Auch für den Fall, dass sich dieser

Problemfall eher banal anhört, so ist es doch wichtig alles bis ins kleinste Detail zu dokumentieren und zu protokollieren, denn sollte wirklich einmal die Zuordnung zwischen Daten und deren Bedeutung nicht mehr 100% gewährleistet sein so sind die Daten nicht mehr für das Endergebnis der Studie auswertbar.

Neben der fehlenden Dokumentation einzelner Variablen ist ein großes Problem vor allem während der Planungsphase medizinischer Studien, schlicht die Definition medizinischer Sachverhalte. So entstehen die ersten Probleme zu dem Zeitpunkt, an dem Krankheitsbilder wie zum Beispiel hoher Blutdruck oder Diabetes allgemein definiert werden sollen. Dabei stellt sich z.B. die Frage, ob ein Patient an Diabetes leidet, sobald er Insulin oder ein anderes Diabetespräparat regelmäßig einnimt, wobei hierbei die Definition von „regelmäßig“ ebenfalls ein kritischer Aspekt ist oder leidet ein Patient an Diabetes, sobald sein Blutzuckerwert einen bestimmten Schwellenwert überschritten hat? Im zweiten Falle, stellt die Definition des Schwellenwertes ebenfalls ein Problem dar. So muss hier neben dem reinen numerischen Wert des Schwellenwertes auch entschieden werden, unter welchen Konditionen der Blutzuckerspiegel gemessen werden soll. Der Blutzuckerspiegel kann dabei bei Patienten zum Beispiel im nüchternen Zustand, das heisst ohne vorherige Nahrungsaufnahme, gemessen werden oder in „normalen“ Alltagssituationen. Wie man hier sieht, ist dieser Aspekt klinischer Studien, zwar ein enorm wichtiger, aber andererseits ein sehr komplexer zumal das Ergebnis der Studien direkt von der Definition der zu untersuchenden medizinischen Faktoren abhängt. Werden daher diese Faktoren zu unscharf definiert, werden konsequenterweise auch unscharfe Patientendaten erhoben, die dann wiederum in das Ergebnis der Studie einfließen und diese verzerren.

Zusammenfassend können folgende essentielle Aspekte festgehalten werden, die für ein hohes Maß an Qualität in medizinischen Studien beherzigt werden müssen (vgl. Zaiss u. a. (2002, 50)):

- Vollzähligkeit der Patientendaten
- Vollständigkeit der erhobenen Merkmale eines Patienten

- Richtigkeit der erhobenen Daten
- Beobachtungsgleichehit durch den Einsatz der gleichen Methodik zur Datenerhebung
- Strukturgleichheit der Daten
- Reproduzierbarkeit der Daten

Um diese Punkte umzusetzen ist ein standardisierter Arbeitsablauf bei der Durchführung klinischer Studien notwendig, was im folgenden Kapitel näher beschrieben wird.

## 3.2. Beschreibung des Workflows in klinischen Studien

Die Erörterung des Workflows in klinischen Studien und im speziellen der HIFAM bzw. GoKard-Studien des Regensburger Universitätsklinikums ist hier insofern ein wichtiger Punkt als sich die praktische Umsetzung dieses Projektes nahtlos in den Gesamtworkflow einer klinischen Studie einfügen sollte. Hierbei ist es allerdings wichtig eine Unterscheidung zwischen dem Gesamtworkflow in klinischen Studien, also angefangen von der Patientenrekrutierung bis hin zur Auswertung der Daten und der Analyse der so erhobenen Ergebnisse, und dem IT-gestützten Subworkflow des Datenmanagements von der Dateneingabe bis zur Analyse zu treffen.

Zur Erklärung des Begriffes Arbeitsabluf bzw. Workflow soll folgende Definition dienen: „Ein Workflow besteht aus mehreren Aktivitäten, die miteinander verbunden sind und von Aufgabenträgern nach festgelegten Regeln ausgeführt werden “ (Vogler (1996, 345)). Die Workflow Management Coalition (WfMC) definiert den Begriff des Workflows folgendermaßen: „The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules “ (Allen

(2001)). Jedoch ist es wichtig zu erwähnen, dass innerhalb eines Gesamtworkflows einzelne Aktivitäten ihrerseits wiederum aus Subworkflows bestehen können. Mit Hilfe von fest definierten Workflows ist es möglich einzelne Prozesse besser zu planen, was einen Gewinn an Zeit und finanziellen Vorteil gegenüber locker definierten Arbeitsabläufen darstellt. Darüber hinaus können so sichtbar gemachte repetitive Aufgaben unter Umständen automatisiert werden, was wiederum finanzielles und zeitliches Einsparpotential bietet. Jedoch stellen zeitliche und damit finanzielle Einsparungen nicht die einzigen Vorteile von fest definierten Arbeitsabläufen dar, denn auch die Qualität der durchgeführten Prozesse wird erhöht, da zu jedem Zeitpunkt bzw. zu jedem Schritt innerhalb eines Arbeitsablaufes genaue Tätigkeiten definiert sind.

Im Folgenden wird der Workflow bei der Durchführung medizinischer Studien dargestellt, um so eine theoretische Grundlage für die praktische Entwicklung eines Softwareframeworks zum Einsatz in klinischen Studien zu erstellen. Generell muss man hierbei zwischen zwei Arten von medizinischen Studien unterscheiden. Zum einen gibt es großangelegte multizentrische Studien, die, wie der Name besagt, nicht nur an einem medizinischen Forschungsinstitut, sondern an mehreren Forschungszentren durchgeführt werden. Zum anderen gibt es Studien, an denen nur kleinere Gruppen von Forschern in nur einem Forschungszentrum beteiligt sind.

Zu ersterer Art der Studien gehört sicherlich die Eurogene-Studie. Das Ziel dieser großangelegten Studie ist es, genetische Voraussetzungen für Herzkrankheiten zu finden. Dazu haben sich zehn europäische medizinische Forschungszentren, darunter das Universitätsklinikum Regensburg, zusammengeschlossen, um Patienten zu rekrutieren, deren Krankengeschichte zu evaluieren und daraus Schlüsse auf mögliche genetische Prädispositionen im Hinblick auf Herzkrankheiten zu finden. Um dies zu bewerkstelligen werden in allen teilnehmenden Zentren die Krankengeschichten von Patienten mit Herzkrankheiten und deren Familien ermittelt und im Universitätsklinikum Regensburg zusammengeführt. Das vermutlich größte Problem bei der Durchführung multizentrischer Studien ist die Einheitlichkeit der Datenspeicherung. Da die Datenspeicherung in diesem Fall mit Hilfe von Desktopda-

tenbanken durchgeführt wird, tritt logischerweise das Problem auf, dass nicht alle Zentren gleichzeitig auf dieselbe Datenbank Zugriff haben. Dadurch besteht ein hohes Risiko, dass im Laufe der Zeit mehrere Insellösungen zur Speicherung der in den Zentren erhobenen Daten, die wiederum möglicherweise verschiedene Anwendungen und folglicherweise verschiedene Datenformate für die Speicherung verwenden, entstehen. Dies ist zwar kein großes Problem aus der Sicht der einzelnen Zentren, jedoch wird die Datenzusammenführung und Auswertung der erhobenen Daten enorm erschwert, da ein großer manueller Aufwand betrieben werden muss, die Daten auf eine einheitliche Grundlage zu bringen. So kann es vorkommen, dass sich die einzelnen Zentren nicht nur unterschiedlicher Datenformate bedienen, sondern folglicherweise auch unter Umständen unterschiedlicher Datentypen, was zu enormen Problemen bei der Integration der Daten führt. Desweiteren entstehen auch durch die Entwicklung eigenständiger Insellösungen Versionen der ursprünglichen Datenbank, die nicht mehr alle oder zusätzliche Variablen beinhalten, was wiederum die letztendliche Auswertung der Daten erschwert und das Ergebnis unnötig verzerrt. Zur Lösung dieses Problems bieten sich eine Webanwendung oder eine einheitliche desktopbasierte Datenbankschnittstelle an, die auf ein zentrales relationales Datenbanksystem Zugriff hat. So könnte die Entstehung einzelner Insellösungen vermieden und die Auswertung der Daten erleichtert und somit die Qualität von klinischen Studien verbessert werden.

Zur zweiten Art von Studien gehört zum Beispiel die Regensburger Herzinfarktfamilienstudie, die ebenfalls dazu dient, mögliche hereditäre Prädispositionen für Herzerkrankungen zu ermitteln. Zur Rekrutierung der großen notwendigen Patientenzahl wurden in deutschen Rehakliniken Krankenakten von in Frage kommenden Patienten, das heisst Patienten mit koronaren Herzkrankheiten, in deren Familien ebenfalls Herzkrankheiten gehäuft auftreten, gesichtet und falls diese und ihre Familienangehörigen der Teilnahme zustimmen, in die Studie aufgenommen. Danach werden in regelmäßigen Abschnitten telefonische Interviews durchgeführt und die DNA von Studienteilnehmern, die bestimmte Kriterien erfüllen, extrahiert. In diesen Telefoninterviews werden vor allem mögliche Erkrankungen seit der letzten

telefonischen Befragung erfragt. Unmittelbar während der telefonischen Befragung werden die Ergebnisse in Datenbanken gespeichert, wo die Daten für weitere Expertenanalysen zur Verfügung stehen.Die Ergebnisse der Analysen, die auf den in Telefoninterviews erhobenen Daten beruhen, sind wiederum Ausgangspunkt für weiterführende Analysen wie etwa Genanalysen, in denen die genetische Prädisposition für bestimmte Krankheitsbilder untersucht wird. Daher ist eine korrekte und qualitativ einwandfreie Durchführung der Telefoninterviews bzw. der Datenspeicherung essentiell. Der Arbeitsablauf ist in Grafik 3.1 schematisch dargestellt. Im Falle von multizentrischen Studien müsste diese Grafik noch um einen Schritt der Datenzusammenführung und -konsolidierung erweitert werden.

Was allerdings alle diese Studien gemeinsam haben ist neben einem verbindlich festgeschriebenen Arbeitsablauf, der *standard operating procedure (SOP)*, die notwendige Zusammenarbeit verschiedener Fachbereiche. So sind neben dem medizinischen Personal, wie den Ärzten und Pflegepersonal, auch Studienkoordinatoren, IT-Fachleute und Studienassistenten von großer Wichtigkeit für die erfolgreiche Durchführung medizinischer Studien. All diese Fachbereiche haben ihre eigenen Aufgaben, die sich in mehreren Schnittstellen treffen. So gehört zu den Aufgaben des Studienkoordinators das Lösen praktischer Probleme, der Aufbau der Studie sowie der Studiendurchführung, was in Zusammenarbeit mit dem medizinischen Personal erfolgen sollte, und die Aufsicht und Koordination der Studienassistenten. Das medizinische Personal sollte dem Studienkoordinator beratend zur Seite stehen sowie die medizinischen Daten erfassen und Patienten für die Studien auswählen. Das Aufgabengebiet der IT-Fachleute umfasst die Beschaffung und Wartung von Computerhardware, Programmierung von Applikationen und Schnittstellen sowie unter Umständen das Retrieval von Daten aus Datenbanken bzw. das Durchführen von Analysen mit Hilfe von Datenbanken in Zusammenarbeit mit der Studienkoordination bzw. dem medizinischen Personal.

Eine weitere Gemeinsamkeit aller Studien ist die Planung der Fallzahlen, sprich der Patienten, die in eine Studie aufgenommen werden sollten, um ein positives und stichhaltiges Ergebnis einer Studie zu erhalten. Hierbei spielen allerdings neben der

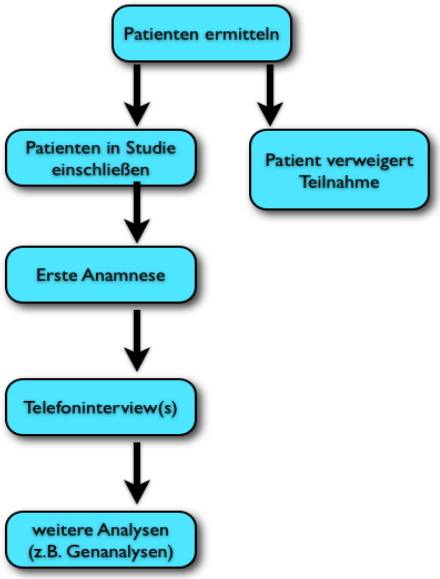



Abbildung 3.1.: Schematischer Workflow in nicht-multizentrischen klinischen Studien

reinen Validität der Studie weitere Faktoren wie etwa die Zeit, die zur Durchführung der Studie zur Verfügung steht sowie finanzielle und personelle Aspekte eine große Rolle. So muss die Fallzahlplanung nicht nur das möglichst positive Ergebnis einer Studie im Auge behalten, sondern ebenfalls die praktische Durchführbarkeit einer Studie im Hinblick auf die zu rekrutierenden und in die Studie einzuschließenden Patientenzahlen. Vor allem bei genetischen Studien spielt der Aspekt der Fallzahlplanung eine große Rolle, da hiervon die statistische Aussagekraft des rein statistischen Ergebnisses abhängt.

## 3.3. Datenmanagement im Hinblick auf das Informationsmanagement in klinischen Studien

Neben der Durchführung klinischer Studien, das heisst dem Workflow bei der Durchführung, ist ein wichtiger Aspekt bei der Durchführung von Studien selbstverständlich das Datenmanagement der erhobenen Patientendaten, auf was im Folgenden näher eingegangen wird. Wie auch in anderen Wisschenschafts- oder Industriebereichen spielt das Management von Informationen also das Bereitstellen, Beschaffen sowie die Speicherung und Kommunikation von Informationen, auch im medizinisch bzw. medizin- und bioinformatischen Bereich eine große Rolle. Vor allem auf Grund der stetig zunehmenden Menge an Informationen insbesondere im Bereich der Genetik, hat das Informationsmanagement eine enorme Bedeutung inne. So ist es ein primäres Ziel des Informationsmanagements, Informationssysteme zu entwickeln und bereitzustellen, die Informationen für alle berechtigten Personen und zum benötigten Zeitpunkt liefern (siehe Kapitel 3.1). Um dies zu bewerkstelligen ist das eigentliche Datenmanagement, also Aspekte der „Datenmodellierung, Datenadministration, Datentechnik, Datensicherheit, Datenkonsistenz, Sicherung von Daten“ (Krcmar (2005, 111)), von großer Bedeutung, da nur mit Hilfe eines effizienten Datenmanagements ein erfolgreiches Informationsmanagement im Allgemeinen sichergestellt werden kann. Um ein zuverlässiges und für alle Beteiligten einheitliches Datenmanagementsystem zur Verfügung zu stellen bieten sich sicherlich web-basierte Lösungen an wie es zum Beispiel Scognamillo u. a. (1999) oder Gillen u. a. (2004) umgesetzt haben. Dieser Ansatz hat vor allem für dezentrale Studien den großen Vorteil der ständigen Verfügbarkeit der gleichen Datenbasis für alle Studienzentren. Darüber hinaus ist eliminiert dieser Ansatz die Fehlerquelle der manuellen oder semimanuellen Datenkonsolidierung im Falle des Datentransfers z. B. mittels FTP. Ebenfalls elektronische Datenmanagementwerkzeuge im Hinblick auf klinische Studien ist das System des „Electronic Data Capture“ , bei dem papierbasierte „Case Report Forms“ (CRF) durch elektronische Formulare ersetzt werden

(siehe etwa Li u. Miller (2007)). Diese Formulare werden vor dem Hintergrund des zu erwartenden Studienberichts erstellt und stellen daher einen eher bottom-up Ansatz dar, bei dem ausgehend vom zu erwartenden Ergebnis aus die Studie und die damit in Verbindung stehenden elektronischen CRFs erstellt werden. Da dieser Ansatz die im Zuge einer Studie erhobenen Berichte als Hauptausgangspunkt für die Planung des Datenmanagements vorsieht, werden die Daten unter Mithilfe verschiedener Rollen (z.B. Datenmanager, Statistiker) von Anfang an so organisiert, dass die Qualität und Zuverlässigkeit der Berichte stets gewährleistet ist. Ein ebenfalls immer weiter verbreiteter Ansatz zum Datenmanagement in klinischen Studien ist das Konzept der „service oriented architecture“ (SOA) (siehe dazu z.B. Calinescu u. a. (2007)). Unter dem Konzept SOA versteht man dabei, dass (Geschäfts-) Prozesse, die auf einer hohen Abstraktionsebene angesiedelt sind in mehrere konkrete Prozess auf einer niedrigeren Stufe aufgeteilt werden. Im Beispiel einer klinischen Studie könnte so der Prozess „Studiendatenerhebung“ auf mehrere konkrete Prozesse wie „Studienteilnehmer anlegen“ oder „Blutdruckdaten erheben“ aufgeteilt werden. So bietet sich die Möglichkeit ein sehr komplexes System bzw. Prozess modular aufzubauen und so das System zum einen leicht wartbar und zum anderen sehr flexibel zu gestalten. Darüber hinaus ist es mit Hilfe des SOA-Ansatzes möglich auch die Kosten klinischer Studien zu minimieren, da das System einer klinischen Studie aus bereits bestehenden kleinen und modularen Services zusammengestellt werden kann.

Diesbezüglich hat das Informationsmanagement im Bereich der biomedizinischen Forschung eine extrem wichtige Rolle inne. Da in diesem Bereich das Informationsmanagement der Entdeckung neuen Wissens und den damit verbundenen Behandlungs- und Diagnosemethoden dient, darf dieser Bereich der Wertschöpfungskette der medizinischen Forschung keinesfalls vernachlässigt werden. Daher ist es wichtig die oben erwähnten Punkte zu beherzigen und ein stabiles, einheitliches Informationsmanagementsystem anstatt proprietärer Insellösungen einzusetzen, um so den größtmöglichen Nutzen aus medizinischen Studien zu ziehen.

Insbesondere die Wichtigkeit der Datenmodellierung darf nicht unterschätzt werden, da dieser Aspekt den Grundstein für das ganze weitere Daten- und Informationsmanagement legt. Hierbei ist darauf zu achten, dass die Daten ganzheitlich und einheitlich modelliert werden, so dass zum einen keine Redundanzen entstehen und zum anderen die Daten und alle damit verbundenen Aspekte komplett in einem Datenmodell dargestellt werden. Für die Entwicklung eines einheitlichen Datenmodells ist es daher ratsam, zuerst alle Variablen, die die einzelnen Fachbereiche benötigen, zu sammeln und diese dann bestenfalls mit Hilfe eines *entity-relationship*-Diagramms zueinander in Beziehung zu setzen. Dabei ist vor allem auf die Art der Beziehungen, sprich 1:1, 1:n oder m:n Beziehung, zu achten, um so von vornherein Unklarheiten und Inkonsistenzen zu vermeiden.

Wurde das Datenmodell erstellt und überprüft, so gilt es ein passendes Speicherformat zu finden. Hierbei hat sich vor allem das Modell einer relationalen Datenbank weitgehend durchgesetzt im Vergleich zu anderen Datenbanksystemen wie etwa objektorientierten Datenbanken. Im Vergleich zur Speicherung in einfachen Dateien oder Desktopdatenbanken wie Microsoft Access, bieten relationale Datenbank(management)systeme neben der problemlosen zeitgleichen Arbeit an den Daten durch verschiedene Benutzer den Vorteil eines Ebenenkonzeptes (siehe Abbildung 3.2). Dieses Konzeptionierungsmodell einer Datenbank bietet für den eigentlichen Benutzer den großen Vorteil, dass sowohl die interne Ebene, also die eigentliche Datenbanksystem-abhängige Speicherung der Daten, sowie die konzeptionelle Ebene, sprich die Beziehungen der Daten zueinander, von der für den Benutzer interessanten Ebene - der externen Ebene - losgelöst sind. So bietet die externe Ebene nur eine Sicht auf die Daten und ist von der Anwendung, mit deren Hilfe Daten gespeichert, angezeigt und gepflegt werden, abhängig. Dies bietet den Vorteil, dass die Datensicht für bestimmte Benutzergruppen angepasst werden kann, so dass nicht jeder Benutzer alle Daten einsehen und pflegen kann, sondern nur die, die für ihn von Bedeutung sind und für den Datenbestand für den er Benutzerrechte besitzt. Außerdem können so die Daten nicht direkt manipuliert werden, was wiederum letztendlich die Datenkonsistenz bzw. -integrität fördert. In der konzeptuellen Ebene hinge-

gen, werden die Daten zueinander, wie bereits erwähnt, in Beziehung gesetzt und die Struktur der Datenspeicherung mit Hilfe der *Data Description Language* (DDL) festgelegt und der Grundstein konsistenter und daher sicherer und qualitativer Daten gelegt, indem nicht nur die Daten in Tabellen strukturiert werden, sondern auch die Tabellen mittels Fremdschlüsselbeziehungen zueinander in Beziehung gesetzt werden. Für diesen Schritt der konzeptuellen Strukturierung der Daten ist vor allem im Bereich der Medizin bzw. Biologie/Genetik eine enge Zusammenarbeit der einzelnen Fachbereiche von großer Wichtigkeit, da zum einen den medizinischen und biologischen Bereichen die notwendige Kenntnis über Datenbanksysteme und zum anderen den informationstechnologischen Bereichen die Kenntnis über genaue medizinisch-biologische Zusammenhänge fehlt. Nur mit einer engen und kommunikativen Zusammenarbeit ist es daher möglich nicht nur alle benötigten Daten und Aspekte beispielsweise einer medizinischen Studie abzubilden, sondern auch deren Zusammenhänge untereinander korrekt zu modellieren.

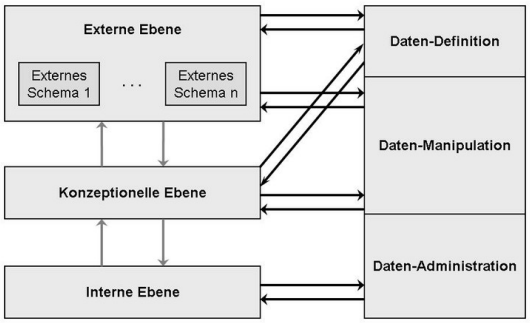



Abbildung 3.2.: Ebenenkonzept eines relationalen Datenbankmanagementsystems (RDBMS) nach ANSI/SPARC (siehe Vossen (2009))

Desweiteren darf der Aspekt der Datensicherheit keinesfalls unterschätzt werden. Vor allem im Bereich der Medizin ist allerdings hierbei nicht nur das Durchfüh-

ren regelmäßiger Backups und die Möglichkeit einer reibungslosen und zeitnahen Wiederherstellung von Daten im Extremfall gemeint, sondern insbesondere auch die Zugriffskontrolle. Da es sich bei Studiendaten, um sehr sensible Daten handelt ist es aus einem ethischen Blickwinkel unumgänglich, dass die Patientendaten nur für einen befugten Kreis von Benutzern zugänglich sind. Dies muss sowohl auf Hardware- als auch auf Softwareebene umgesetzt werden. So bietet sich auf Hardwareebene an, das Datenbanksystem und die damit verbundenen Applikationen in einem eigenen von aussen abgeschotteten Netzwerk innerhalb der Klinik oder der Forschungseinrichtung zu betreiben. Auf Applikations- bzw. Softwareebene muss der Zugriff auf die Daten mit Hilfe von speziellen Zugriffsrechten und Passwortabfragen geregelt werden, um einen sicheren Umgang mit Patientendaten zu gewährleisten.

Neben der Datensicherheit spielt selbstverständlich auch die Konsistenz der Daten, die in klinischen Studien erhoben werden, eine enorme Rolle. Nicht zuletzt da die Analysen und die daraus gezogenen Schlüsse direkt von den erhobenen Daten abhängen, ist es wichtig alle möglichen Probleme, die zu Inkonsistenzen führen können zu vermeiden. Zu diesen zählen vor allem fehlerhaft erstellte Datenmodelle, die nicht den allgemein anerkannten Normalformen folgen (siehe dazu Date (2004, 357ff)). So muss etwa sichergestellt werden, dass jedes Attribut einer relationalen Tabelle nur atomare Werte beinhaltet (erste Normalform). Das bedeutet, dass mehrere Werte pro Spalte in einer Tabelle vermieden werden müssen, da dies zum einen Probleme bei der Aktualisierung der Daten verursacht und zum anderen die Abfrage der Daten verkompliziert. Ebenso ist es wichtig, dass jedes Nichtschlüsselattribut lediglich vom Primärschlüssel voll funktional abhängig ist (zweite Normalform). Das heisst, dass kein Nichtschlüsselattribut von nur einem Teil des Primärschlüssels oder gar von einem Attribut, das nicht Teil des Primärschlüssels ist, abhängen darf. Der Grund für diese Einschränkung ist, dass bei einer Verletzung der zweiten Normalform die Gefahr besteht, dass Datensätze redundant auftreten und dies möglicherweise bei Aktualisierungen dazu führen kann, dass nicht alle gewünschten Datensätze aktualisiert werden. Die Abhängigkeit von Nichtschlüsselat-

tributen spielt ebenfalls bei der dritten Normalform eine Rolle, welche besagt, dass jedes Nichtschlüsselattribut von keinem Primärschlüsselattribut transitiv abhängig sein darf, was wiederum zu unnötigen Redundanzen in den Daten führt. Obwohl es weitere Normalformen in der Datenbanktheorie gibt, soll die Erklärung der ersten drei Normalformen an dieser Stelle genügen, da diese am häufigsten in der Datenbanktheorie auftreten und die weiteren Normalformen sehr spezielle Restriktionen festlegen.

Als konkretes Beispiel für die Studiendurchführung und das damit verbundene Daten- bzw. Informationsmanagement soll hier die Regensburger Herzinfarktfamilienstudie dienen, die am Regensburg Universitätsklinikum konzipiert und durchgeführt wird. Ursprünglich wurden alle Daten, die im Rahmen dieser Studie erhoben wurden in einfachen Tabellenkalkulationsdateien dezentral auf verschiedenen Arbeitsplatzrechnern gespeichert. Diese Art des Datenmanagements stieß aber schon bald an seine Grenzen, da die Auswertung und die Handhabung von großen Datenmengen mit Hilfe von simplen Tabellenkalkulationsprogrammen enorm fehleranfällig und schwer zu bewerkstelligen ist. Daher wurden nach und nach die Daten in das Microsoft Access-Format migriert und pro Studie eine Access-Datenbank erstellt. Diese Datenbanken wurden auf einem zentralen Server gespeichert, um so die Verfügbarkeit für alle an einer Studie beteiligten Personen zu gewährleisten.

Die dabei erstellten Datenbankschemata zeichneten sich vor allem durch ein äusserst horizontales bzw. flaches Design aus. Darunter versteht man, dass es nur wenige Tabellen mit sehr vielen Feldern, in denen die erhobenen Daten gespeichert werden, gibt. Dies ist insofern problematisch, da die Daten, die in klinischen Studien erhoben werden, eher vertikaler Natur sind. Das heisst, dass viele Merkmale und deren Ausprägungen für relativ wenige Patienten im Vergleich zu der Gesamtanzahl der Patienten eines Klinikums erhoben werden. Dies bewirkt folglich, dass vertikale Datenmodelle in ein horizontales Schema gezwängt werden und die gesamte Dokumentation daher schwerer zu warten und auszuwerten ist. Im konkreten Fall bedeutete das, dass für jedes Auftreten eines Merkmals wie etwa der Einnahme eines Medikamentes eine neue Variable in der Tabelle hinzugefügt werden muss. Sollte

es sich von vornherein um eine fixe Anzahl an Merkmalshäufigkeiten handeln, wie etwa einer ersten und zweiten Blutdruckmessung, so wäre kaum etwas gegen dieses Tabellenlayout einzuwenden. Bei unbekannten Merkmalshäufigkeiten führt dies allerdings zu ständigen Änderungen des Tabellenlayouts und der damit verbundenen Abfragen, was wiederum einen hohen Arbeitsaufwand und potentielle Fehlerquellen bedeutet. Ferner hat dies wiederum nicht nur mögliche Integritätsverletzungen zur Folge, sondern erschwert auch das Warten von Datenbankformularen, da sich für jede neue Spalte in der Ursprungstabelle auch die Anzahl der Eingabefelder in den entsprechenden Formularen ändern muss. Zwar kann ein horizontaler Tabellenaufbau die Performanz von Abfragen steigern, allerdings muss man natürlich bedenken, dass die Wartung und Erweiterung von horizontalen Tabellen ebenfalls viel Zeit in Anspruch nehmen. Andererseits haben auch vertikale Tabellen Nachteile vor allem wenn Daten mit Hilfe von Join-Operationen ermittelt werden, können diese Abfragen schnell sehr komplex werden. Jedoch sinkt der Wartungsaufwand dieser Tabellen und der damit in Verbindung stehenden Formulare drastisch, weshalb im Zusammenhang mit Daten, die in klinischen Studien erhoben werden, die vertikale Strukturierung der Tabellen vorzuziehen ist, zumal die erhobenen Daten überschaubar sind. So wurden im Zuge der HIFAM-Studie bisher ca. 7000 Individuen in Tabellen gespeichert, was kaum Performanzverbesserungen in einer horizontalen Tabellenstruktur mit sich bringt. Auch wenn im Bereich der Medizin desöfteren Microsoft Access als Software für die Datenspeicherung zum Einsaz kommt (vgl. Kho u. a. (2007)), so muss doch erwähnt werden, dass diese Software nicht für den Einsatz mehrerer gleichzeitiger Benutzer geeignet ist und ein mögliches manuelles verschmelzen verschiedener Datenquellen, für den Fall, dass die Access-Datenbank an weitere Benutzer geschickt wurde und später zu einer großen Datenbank zusammengefügt werden soll, unnötiges Fehlerpotential birgt.

## 3.4. Vorteile und Einführung eines relationalen Datenbankmanagementsystems

Auf Grund der eben erwähnten Probleme mit Desktopdatenbanken im Zusammenhang vor allem mit der zeitgleichen Datenbearbeitung durch mehrere Benutzer und der Einhaltung von Normalformen ist es, vor allem für größere Datenmengen, notwendig die Datenspeicherung dahingehend zu verändern, dass mehrere Benutzer zeitgleich Daten speichern können ohne dadurch die Daten zu kompromittieren. Deshalb ist der erste Schritt hin zu einem Informationssystem, die Einführung einer relationalen Datenbank. In diesem Falle wurde die Open-Source Datenbank PostgreSQL (http://www.postgresql.org) in der Version 8.2 eingesetzt wie sich auch untere anderem bei Dugas u. a. (2002) zum Einsatz gekommen ist. Dieses relationale Datenbankmanagementsystem (RDBMS) bietet den Vorteil, dass es zum einen kostenfrei erhältlich ist und zum anderen ist es möglich die Datenspeicherung in Transaktionen zu kapseln und die Integritätsprüfungen erst am Ende einer Transaktion durchzuführen. Desweiteren ist es möglich Plausibilitätsprüfungen auf der Seite der Datenbank durchführen zu lassen, indem *stored procedures* erstellt werden, die wiederum durch *trigger* ausgelöst werden. Auf diese Art und Weise ist es möglich mit relativ einfachen Mitteln die Plausibilität der Daten sicherzustellen und eigene, dem Benutzer verständliche, Fehlermeldungen anzuzeigen sobald eine Plausibiliätsregel nicht eingehalten wurde. Tritt dieser Fall ein, so werden auch die Daten nicht gespeichert bis der Benutzer den aufgetretenen Fehler behoben hat.

Ein weiterer großer Vorteil, der mit der Einführung eines RDBMS einhergeht ist sicherlich die Mehrbenutzerfähigkeit im Gegensatz zu MS Access. Durch Locking-Mechanismen und mit Hilfe des ACID-Paradigmas ist es möglich für mehrere Benutzer zeitgleich in der Datenbank zu arbeiten. Das ACID-Paradigma setzt für Transaktionen folgende essentielle Punkte (siehe Date (2004, 452 f.)) voraus:

- *Atomicity*
- *Correctness*

- *Isloation*

- *Durability*

Diese vier Punkte dienen zur Sicherstellung der Datenintegrität beim Einsatz von Transaktionen, was für ein Mehrbenutzersystem unerlässlich ist. Der Punkt *atomicity* besagt dabei dass Transaktionen nach einem „Alles-oder-Nichts Prinzip“ ablaufen sollen. Das heisst, dass Speicher-, Lösch- oder Aktualisierungsoperationen nur dann vollständig umgesetzt werden, falls während der Durchführung kein Fehler auf Grund von z.B. Schlüsselverletzungen auftritt. Diese Voraussetzung ist auch bei der Umsetzung von GUI4DB (siehe Kapitel 3.6) beherzigt worden, da jeder Benutzer die Datenspeicherungen und -Änderungen in einer Transaktion durchführt, das heisst, dass zum Beispiel ein neu anzulegender Patient nur dann gespeichert wird, falls während der Speicherung der Patientendaten in die einzelnen Tabellen kein Fehler auftritt. So wird die Datenintegrität zu 100% sichergestellt, was wiederum die Qualität der Auswertungen der Daten enorm erhöht.

Ähnlich zum Punkt *Atomicity* besagt der Punkt *Correctness*, dass nach dem Ende einer beliebigen Transaktion sich die Datenbank in einem korrekten Zustand befinden muss. Sollte dies nicht der Fall sein, da zum Beispiel Schlüsselverletzungen während einer Transaktion auftraten, so muss die Transaktion mit Hilfe von *ROLLBACK* rückgängig gemacht werden. Allerdings gilt es dabei zu beachten, dass die Korrektheit der Datenbank nur am Ende einer Transaktion gewährleistet sein muss (siehe Date (2004, 452)). Dies ist vor allem dann von Interesse, wenn Schlüsselprüfungen zeitlich verzögert werden, da nur nach dem Speichern aller Daten aber vor dem Ende einer Transaktion die Schlüssel überprüft werden können. In diesem Fall werden zwar die Daten unter Umständen für einen geringen Zeitraum gespeichert ohne dass die Integrität zu 100% gewährleistet ist, allerdings wird mit Hilfe des Prinzip der *Correctness* sichergestellt, dass die Transaktion nur dann abgeschlossen wird, falls die am Ende durchgeführten Überprüfungen auf Integrität erfolgreich waren.

Das Prinzip der *Isolation* besagt, dass jede Transaktion für alle anderen simultanen Transaktionen nicht einsehbar sein soll. Das bedeutet, dass der Benutzer A nur dann die Daten sieht, die der Benutzer B speichert, sobald die Transaktion des Benutzers B abgeschlossen ist. Derzeit gibt es dazu im SQL-Standard vier verschiedene Level von Isolationen (siehe Date (2004, 458)):

- *Read uncommitted*
- *Read committed*
- *Repeatable read*
- *Serializable*

Diese vier Level sollen dazu dienen die folgenden Probleme unsauberer Transaktionen zu verhindern:

- Lesemöglichkeit von Daten einer anderen Transaktion (*dirty read*)
- Wiederholtes Lesen von geänderten Daten innerhalb einer Transaktion (*non-repeatable read*), obwohl während einer Transaktion nur ein Stand der Daten einsehbar sein sollte.
- Abfrage von Daten, die während der Laufzeit der Abfrage von einer anderen Transaktion geändert wurden, was die Ergebnisse der Daten beeinflusst (*phantom read*)

Je nach dem gewünschten Sicherheitsgrad kann eins der vier oben erwähnten Isolationslevel gewählt werden wobei *Serializable* als das strikteste gilt. Das bedeutet, dass im Isolationslevel *Serializable* wirklich nur die Daten zur Verfügung stehen, die nach erfolgreichen Transaktionen gespeichert wurden. So werden alle drei der oben erwähnten Phänomene behoben. Im Gegensatz dazu bietet *Read uncommitted* keine Sicherheit vor dem Lesen nicht gespeicherter Daten, was unter Umständen zu großen Diskrepanzen in den Daten mehrerer Transaktionen führen kann. Auf

Grund der sensiblen Daten vor allem im medizinischen Bereich und den damit verbundenen ethischen und finanziellen Aspekten sollte für alle erstellten Datenbanken das höchste und sicherste Isolationslevel gewählt werden, um die Datenintegrität zu allen Zeiten zu 100% sicherzustellen.

Der letzte Punkt des ACID-Paradigmas, *Durability*, besagt, dass Daten auch nach einer Transaktion noch in der Datenbank gespeichert sein müssen. Was auf den ersten Blick banal und selbstverständlich klingt, besagt jedoch, dass auch für den Fall eines Systemausfalls alle gespeicherten Daten mit Hilfe von Systemwiederherstellungsmechanismen wieder für die Benutzer zugänglich gemacht werden müssen. Um diesen Punkt zu gewährleisten stellen RDBMS-Systeme herstellerspezifische Mechanismen zur Verfügung. Hier soll jedoch nur kurz auf den häufig eingesetzten Mechanismus des *Point-In-Time Recovery* (PITR) eingegangen werden. Unter PITR versteht man dabei, dass nach jeder erfolgreichen Transaktion in einem speziellen Verzeichnis alle durchgeführten Änderungen gespeichert werden, die dann mit Hilfe dieses Mechanismus nach einem Systemcrash zum Zeitpunkt der letzten erfolgreichen Transaktionen wiederhergestellt werde können.

### 3.4.1. Aufbau der Datenbank für die GoKard-Studie

Als Studie, die auf einem relationalen Datenbanksystem anstatt eines Desktopdatenbanksystems beruht, soll an dieser Stelle die GoKard-Studie erwähnt werden. Für diese Studie wurde das frei zugängliche Datenbanksystem PostgreSQL in der Version 8.2 installiert. Innerhalb dieser Datenbank wurde für jede Studie ein eigenes Schema erstellt, um so Namenskonflikte bzgl. der Tabellennamen zu vermeiden. Dies ist insofern sinnvoll, da normalerweise verschiedene Telefoninterviews pro Studie durchgeführt werden, deren Daten in gleichnamigen Tabellen gespeichert werden, die wiederum von einander getrennt sein sollen. Es wäre zwar möglich alle Daten in derselben Tabelle mit einem zusätzlichen Wert *interview* zu speichern, allerdings werden die Daten kaum interviewübergreifend abgefragt und somit kann eine komplexere Abfrage vermieden werden. Darüber hinaus bietet dieser schema-

basierte Ansatz ein höheres Maß an Übersichtlichkeit. In Abbildung 3.3 wird das Datenbankschema speziell für das Erstinterview der GoKard-Studie gezeigt.

Die komplette GoKard-Datenbank beruht auf dem Designansatz eines *Snowflake*-Schemas. Dabei werden typischerweise zwei Arten von Tabellen unterschieden. Zum einen Faktentabellen, in denen die zentralen, zu verwaltenden Fakten gespeichert werden. Davon hängen die Dimensionstabellen ab, in denen die beschreibenden Daten in logisch und semantisch zusammenhängenden Einheiten liegen. Im konkreten Fall bedeutet das, dass es eine zentrale Tabelle gibt, in der die Patienten-IDs und andere wichtige zentrale Patientendaten gespeichert werden. Von dieser zentralen Faktentabelle hängen direkt oder indirekt alle Dimensionstabellen wie zum Beispiel Tabellen zur Speicherung der Krankheitsgeschichte eines Patienten ab. Würden alle Dimensionstabellen direkt von der Faktentabelle abhängen, so würde es sich um ein Sternschema handeln. Da aber im Falle der GoKard-Datenbank nicht alle Dimensionstabellen direkt von der zentralen Faktentabelle abhängen, sondern einige Dimensionstabellen wiederum von Dimensionstabellen abhängen spricht man von einem *Snowflake*-Schema. Doch obwohl das Sternschema „durch seinen klaren und leicht verständlichen Aufbau und auf Grund der niedrigen Anzahl von Tabellenjoins durch eine höhere Performanz besticht“ (siehe Gabriel u. Röhrs (2003, 377)) sind jedoch Schneeflockenschemata besser normalisiert als Sternschemata. Die Entscheidung welcher Designansatz letztendlich zum Einsatz kommt ist daher eine Entscheidung zwischen Übersichtlichkeit und Performanz auf der einen Seite und Datenkonsistenz andererseits. Da diese Datenbank dazu dienen sollte eben Konsistenzprobleme zu beheben, fiel die Entscheidung auf ein Schneeflockenschema.

Das Kernstück, sprich die Faktentabelle, des Datenbankschemas (siehe Bild 3.3) ist die Tabelle *stammdaten*, die die wichtigsten Informationen zu an der Studie teilnehmenden Patienten beinhaltet. Die Spalte *id* stellt den Primärschlüssel dar, denn jeder anonymisierte Patient d.h. jede ID kann jeweils nur einmal pro Studie eingetragen werden. Ausnahmen stellen hierbei Tabellen dar, in denen z.B. Herzinfarktdaten gespeichert werden, da jeder Patient mehrere Herzinfarkte gehabt ha-

ben kann. In diesen Tabellen dient eine generische *SERIAL*-Nummer als Primärschlüssel. Diese *id* aus der Tabelle *stammdaten* dient ebenfalls als referenzierte Spalte in den Fremdschlüsseln anderer Tabellen, um somit die referentielle Integrität der Datenbank sicherzustellen. Eine kleine Ausnahme bilden die Tabellen *klappenstoerung*, *aortenklappe* bzw. *mitralklappe*. Von diesen Tabellen hängt lediglich die Tabelle *klappenstoerung* direkt vom Primärschlüssel *stammdaten.id* ab, die Tabellen *aortenklappe* und *mitralklappe* hängen wiederum von der Tabelle *klappenstoerung* in einer Fremdschlüsselbeziehung ab. Dies bewirkt, dass schon in Form des Datenbankschemas definiert ist, dass ein Patient eine Klappenstörung gehabt haben muss, um überhaupt eine Störung der Aorten- bzw. Mitralklappe gehabt zu haben. Dies wiederum festigt die referentielle Integrität und Normalität der Datenbank und somit die Qualität der Daten schon auf der untersten Ebene, da keine Werte eingetragen werden können, die nicht eingetragen werden dürften.

Um die Plausibilität der eingegebenen Daten zu gewährleisten wurden für die zu überwachenden Tabellen *stored procedures* erstellt, mit deren Hilfe es möglich ist, die Plausibilität der Daten einfach zu überprüfen. Diese *stored procedures* werden mittels *trigger* aufgerufen sobald ein neuer Wert eingegeben oder ein bereits vorhandener Wert aktualisiert wird. Sollte bei der Überprüfung ein Fehler auftreten, so wird eine Ausnahme erzeugt und die komplette Transaktion abgebrochen bzw. zurückgesetzt. So wurde für jede zu überprüfende Tabelle ein *trigger* erstellt (z.B. siehe B.4), der die entsprechende Plausibilitätsprüfung startet. Im Beispiel B.4 wurde ein *trigger* erstellt, der jedes mal für jeden Tupel aufgerufen wird, für den eine *INSERT*- oder *UPDATE*-Operation durchgeführt wird. Für jede Operation zur Speicherung oder Aktualisierung von PTCA-Daten (Perkutane transluminale coronare Angioplastie) wird dann die Funktion *check_ptca()* aufgerufen. In dieser Funktion (siehe Listing B.5) werden die Daten überprüft, die in der Tabelle *gokard_erstiv . ptca* gespeichert werden.

Wie in Listing B.5 ersichtlich werden zuerst die Daten einer anderen Tabelle mit Hilfe einer SELECT-Anweisung ermittelt, von denen die Daten der hier als Beispiel benutzten Tabelle abhängen. Sollte nun der Fall eintreten, dass zwar PTCA-Daten

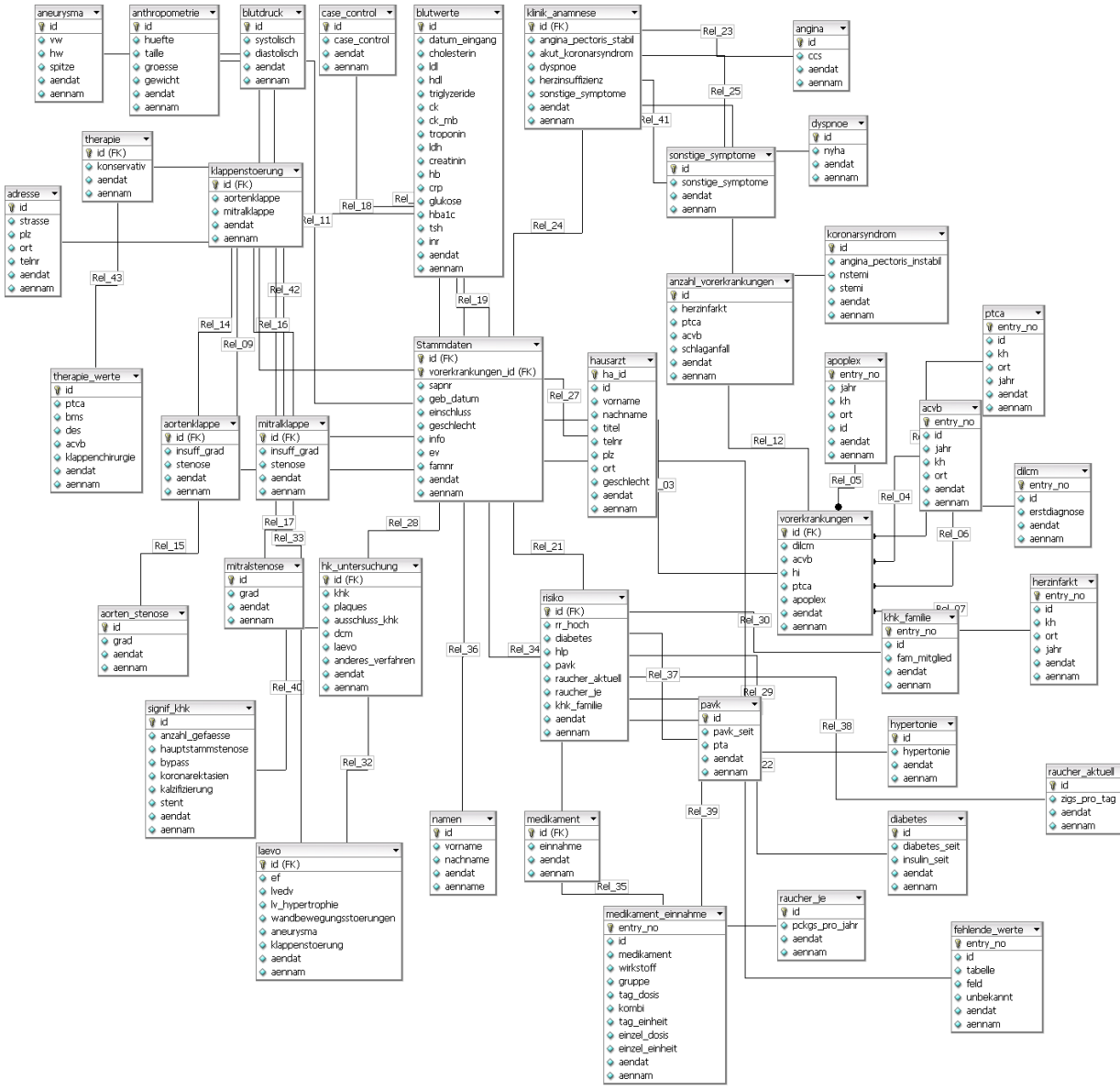



Abbildung 3.3.: GoKard-Datenbankschema

für einen Patienten angelegt werden sollen, der nie eine PTCA hatte, wird eine Ausnahme erzeugt. In diesem Fall werden keine Daten in der Datenbank gespeichert, da die Transaktion unterbrochen und wieder zurückgesetzt wird. Sollten überhaupt keine Daten im Formular in GUI4DB angegeben worden sein, wird die Prüfung abgebrochen und die nächste Tabelle überprüft. Im nächsten Schritt (Zeile 16) wird überprüft, ob die eingegebene Jahreszahl, in der der PTCA aufgetreten ist, valide ist. Die Jahreszahl ist nur dann gültig, falls sie zwischen 1900 und dem jetzigen Jahr liegt. Der Grund dafür, dass Werte bis einschließlich 1900 zugelassen werden, liegt daran, dass 1900 in Fällen, in denen das Jahr nicht 100% ermittelt werden konnte, als „Dummy"-Wert benutzt wird. Sollte die Überprüfung an diesem Punkt scheitern wird in obigem Fall wieder die Transaktion unterbrochen und kein Wert gespeichert. Die Daten, die nicht ermittelt werden konnten, können entweder mit „999" im Falle von numerischen Werten, „1900" im Falle von Jahreszahlen oder „01.01.1900" im Fall von Datumsangaben angegeben werden. Sollte einer dieser Werte in den zu speichernden Daten auftreten, wird der Wert des entsprechenden Feldes auf *NULL* gesetzt und das Fehlen des Wertes in einer separaten Tabelle (*gokard_erstiv . fehlende_werte*) vermerkt. Dabei wird unterschieden, ob der eingegebene Wert einen „Dummy"-Wert beinhaltet oder ob gar nichts für die entsprechende Variable angegeben wurde. Für den ersten Fall wird gespeichert, dass der Wert generell nicht ermittelbar ist und für den zweiten Fall, dass der Wert prinzipiell ermittelbar wäre, aber zum jetzigen Zeitpunkt die entsprechenden Daten nicht vorhanden sind. Dies erleichtert die spätere Auswertung der Daten, da zum einen die „Dummy"-Werte nicht mehr in den eigentlichen Tabellen auftauchen und somit nicht mit Hilfe einer oder mehrerer Bedingungen in einer Abfrage ausgeschlossen werden müssen. Zum anderen bietet dieser Ansatz auch Auskunft darüber welche Werte eventuell noch nachgefragt werden müssen.

### 3.4.2. Aufbau der Genom-Datenbank

Da sich ein wichtiger Bestandteil der vorliegenden Arbeit mit der Auswertung bzw. dem Clustering von genetischen Daten beschäftigt, ist es notwendig ebenfalls auf die Datenbank bzw. das Datenbankschema der Gendatenbank einzugehen. Wie bereits erwähnt wurde als Datenbankserver PostgreSQL in der Version 8.2 eingesetzt, welche auch bei anderen ähnlichen Vorhaben (siehe Wegrzyn u. a. (2008) oder Dugas u. a. (2002)) eingesetzt wurde. In diesem Zusammenhang war insbesondere die enorme Datenmenge von ca. 800 Millionen Tupeln eine große Herausforderung. Da die vorliegenden Daten rein statisch sind und nicht verändert werden müssen, wurde auf den Einsatz eines ausgefeilten Datenbankdesignansatzes wie für die GoKard-Datenbank verzichtet, da bei diesen Daten die Performanz bei der Datenermittlung Vorrang hat. Da die Daten auch nicht manipuliert werden können und nur zu Analysezwecken benötigt werden, wurde darauf geachtet die Anzahl notwendiger JOIN-Operationen zwischen mehreren Tabellen so gering wie möglich bei der Abfrage der Daten zu halten indem so wenig Verknüpfungen zwischen einzelnen Tabellen erstellt wurden wie möglich. Die zu integrierenden Daten lagen in einem reinen textbasierten Format vor (siehe Tabelle 3.1), in dem die jeweiligen Spalten mit Tabulatoren voneinander getrennt waren. Zur Integration der Daten wurde das PostgreSQL-eigene Kommando *COPY* verwendet, das es dem Benutzer ermöglicht Daten aus einer CSV-Datei direkt in eine Tabelle zu übertragen bzw. Daten aus einer Tabelle direkt in eine Text-Datei zu exportieren. Allerdings wurde für den Import eine temporäre Tabelle eingesetzt, in die zuerst alle Daten integriert wurden und von wo sie in die einzelnen Tabellen, basierend auf der Anzahl der Probanden-IDs (PIDs), kopiert wurden, um so sicherzustellen, dass die Anzahl der Tupel in einer Tabelle kein allzu großes Performanzproblem darstellt und alle Daten eines Individuums in einer Tabelle zu finden sind.

Da PostgreSQL die Möglichkeit bietet Daten zur Performanzsteigerung zu partitionieren wurde folgender Aufbau der Datenbank gewählt. In einer Master-Tabelle *snp_allel_master* wird prinzipiell nur die Daten- bzw. Tabellenstruktur aufgebaut.

| Chromosome | Position | PID | SNP-ID | Allel 1 | Allel 2 |
|---|---|---|---|---|---|
| chr_22 | 014433758 | ZZZ000110011 | RS915677 | C | C |
| chr_22 | 014433758 | ZZZ000110022 | RS915677 | A | G |
| chr_22 | 014433758 | ZZZ000110033 | RS915677 | A | T |
| chr_22 | 014433758 | ZZZ000110033 | RS915677 | - | - |

Tabelle 3.1.: Textbasiertes Format der SNP-Daten

| entry_no | pid | snp_id | allele_1 | allele_2 | chromosome |
|---|---|---|---|---|---|
| bigserial | varchar(15) | varchar(13) | varchar(1) | varchar(1) | smallint |

Tabelle 3.2.: Mastertabelle der SNP-Daten

Diese Struktur ist wie in Tabelle 3.2, wobei das Speichern des Chromosoms, auf dem sich die SNP-Daten befinden nur für die tatsächliche Mastertabelle von Belang ist. So können über die Mastertabelle alle Daten chromosomabhängig komplett abgefragt werden.

Von der Mastertabelle (siehe Tabelle 3.2) erben dann chromosomspezifische Mastertabellen. Das heisst, dass jedes Chromosom eine eigene Haupttabelle hat, von der wiederum maximal 23 Untertabellen erben. In diesen bis zu 23 Untertabellen wurden dann die eigentlichen Daten gespeichert und die Elterntabellen dienen lediglich dazu, dass chromosomübergreifend Daten ermittelt werden können.

Da die meisten Abfragen die Patienten-ID (pid) und die SNP-ID bzw. das Chromosom als Kriterien haben, gibt es eine Metatabelle, in der die pids und die Tabellen, in denen sie auftreten gespeichert werden. Nachdem die Daten an Hand der pids in die Tabellen importiert wurden, tritt eine Patienten-ID nur in einer Tabelle pro Chromosom auf. Der Aufbau dieser Metatabelle (*snps_table_overview*) ist wie in Tabelle 3.3 gegeben. Der Tatsache, dass eine PID nur jeweils in einer Tabelle pro Chromosom auftritt, wurde mit dieser Tabelle Rechnung getragen, um so eine Performanzsteigerung bei der Abfrage von SNPs pro PID zu erhalten, da nicht alle bis zu 23 Kindtabellen pro Chromosom abgefragt werden müssen, sondern lediglich eine einzelne Tabelle mit maximal zwei Millionen Tupeln. Mit Hilfe der Metatabel-

| table_name | chromosome | pid | study |
|---|---|---|---|
| VARCHAR(50) | SMALLINT | VARCHAR(15) | VARCHAR(5) |

Tabelle 3.3.: Metatabelle über PIDs und Tabellen

le in Tabelle 3.3 ist es möglich in kurzer Zeit die betreffende Tabelle zu ermitteln, um die Abfrage dann nur auf diese eine Tabelle anzuwenden. Vor allem für den programmatischen Export von SNP-Daten an Hand von PIDs über mehrere Chromosome hinweg bietet sich dieses Verfahren an. So werden für jedes Chromosom die Daten aller PIDs aus den entsprechenden Tabellen ermittelt, was wiederum eine Performanzsteigerung bewirkt, da PostgreSQL einen Cachingmechanismus bei Abfragen verwendet, der die Ergebnisse von WHERE-Klauseln im Arbeitsspeicher behält, um so Abfragen schneller durchführen zu können. Da nun die WHERE-Klauseln bis auf die PID immer gleich sind, sinkt die Dauer einer Abfrage von bis zu 20 Sekunden auf wenige Millisekunden für eine chromosomweite Abfrage über mehrere PIDs. Diese Performanzsteigerung ist vor allem vor dem Hintergrund der Datenmenge ein unverzichtbares Kriterium für eine gelungene und durchführbare Analyse der vorhandenen Gendaten. Auch vor dem Hintergrund des in Kapitel 4 näher beschriebenen Clusterings von SNP-Daten ist es von enormer Wichtigkeit Daten mit hoher Performanz ermitteln zu können.

Eine weitere Metatabelle ist die Tabelle *snps_map*. In dieser Tabelle werden alle SNPs, die auf einem 500K-Chip, dem Speichermedium, das die SNP-Daten beinhaltet, vorhanden waren an Hand ihrer Position auf dem menschlichen Genom sowie des Gens, auf dem sie gefunden wurden, näher beschrieben. Der Aufbau der Tabelle ist in 3.4 gegeben. Diese Tabelle beinhaltet Positionsdaten von *single nucleotide polymorphisms* auf dem menschlichen Genom, sowie Daten mit welcher Probenid die SNPs während der Genotypisierungsphase ermittelt wurden. Diese Tabelle dient hauptsächlich dazu, die Reihenfolge beim Exportieren vor allem in das PED-Format (siehe Kapitel 2.2.1) zu gewährleisten. So werden beim Export die Daten an Hand der Position der SNPs auf dem menschlichen Genom aufstei-

| probe_id | pos | chromosome | cytoband | snp_id | gene |
|---|---|---|---|---|---|
| CHAR(13) | INTEGER | CHAR(2) | CHAR(6) | CHAR(13) | CHAR(14) |

Tabelle 3.4.: Tabelle über SNP-Position auf dem menschlichen Genom

gend sortiert ermittelt und ebenso in den entsprechenden Dateien gespeichert. Dies ist bei der Erstellung von MAP-Dateien und PED-Dateien wichtig, da die Reihenfolge der SNP-IDs in der MAP-Datei zu 100% der Reihenfolge der Allele in der PED-Datei entsprechen muss. Ist dies nicht der Fall so entstehen verzerrte Analyseergebnisse, da die SNPs nicht mehr den Allelen zugeordnet werden können. Dieses Vorgehen erhöht zwar einerseits die Laufzeit einer Abfrage, stellt aber andererseits sicher, dass die Daten in der gleichen Reihenfolge gelesen und analysiert werden. Es wäre natürlich möglich gewesen, die Positionsdaten zusammen mit den SNPs in den eigentlichen Alleltabellen zu speichern, allerdings würde das den Speicherplatz unnötig stark erhöhen, da mit der Speicherung der Position zusammen mit jedem Eintrag ein hohes Maß an Redundanz entstünde.

Der prinzipielle Aufbau einer Abfrage zur Ermittlung der Allele eines Patienten auf einem Chromosom ist in Listing B.2 gegeben. Wie in Listing B.2 ersichtlich erfolgt die Abfrage zur Ermittlung der Allele bzw. SNPs eines Patienten mit Hilfe eines JOINs über die Tabelle, in der die Position der SNPs auf dem menschlichen Genom gespeichert sind. Ein weiteres Kriterium ist dabei das Chromosom, dessen SNPs pro Patient ermittelt werden sollen und natürlich die ID des Patienten. Als JOIN-Kriterium wird dabei die ID des SNPs verwendet, da diese in beiden Tabellen auftritt und auch in beiden Tabellen angeglichen wurde. Der Angleich der SNP-IDs war insofern notwendig, da in der ursprünglichen SNP-Map alle IDs in Kleinbuchstaben angegeben waren und in allen anderen Tabellen in Großbuchstaben was einen JOIN-Vergleich an Hand der IDs nicht zuließ. Ebenfalls wichtig bei der obigen Abfrage ist die ORDER BY-Anweisung. So wird sichergestellt, dass bei allen Abfragen, die auf die SNP-Daten Bezug nehmen die Reihenfolge übereinstimmt, was vor allem für den Export in PED- und MAP-Dateien von ernormer Wichtigkeit

| gene | func | go_id | kind |
|---|---|---|---|
| VARCHAR(14) | VARCHAR(220) | VARCHAR(10) | VARCHAR(1) |

Tabelle 3.5.: Tabelle zur Speicherung der Genfunktionen

ist, da Auswertung dieser Dateien nur 100% stimmig sind wenn die Reihenfolge der Allele in der PED-Datei mit der Reihenfolge der SNP-IDs in der MAP-Datei übereinstimmt.

Um die nicht zu unterschätzende Komplexität des Exports und des Erstellens von Auswertungen genetischer Daten für die Benutzer so gering wie möglich zu halten, wurde für die Analyse- und Datenbeschaffungsprozesse ein Plugin für GUI4DB (siehe Kapitel 3.6) erstellt, das es ermöglicht die Daten sowohl zu exportieren als auch die in Kapitel 2.3.1 beschriebene Anwendung PLink mit den gefundenen Daten für Analysezwecke aufzurufen.

Für die Clusteringanalysen und die dafür benötigten Metadaten über Gene wurden Daten aus der GeneOntology (siehe Kapitel 2.3.3) integriert. So wurden in entsprechenden Tabellen die Genfunktionen, Synonyme für Gene und die Level der Gene innerhalb der GO-Taxonomie gespeichert, um so jederzeit und ohne Zeitverlust, wie etwa bei der Übertragung der Daten via Internet, Zugriff auf die notwendigen Daten zu haben. Zu diesem Zweck wurden drei Tabellen erstellt.

Die Tabelle *gene_functions* beinhaltet, wie der Name vermuten lässt, die funktionellen Beschreibungen von Genen. Der genaue Aufbau der Tabelle ist in Tabelle 3.5 gegeben. Der Name des Gens ist an Hand der HUGO-Nomenklatur gegeben und die genaue funktionelle Beschreibung des Gens, wie sie in der GeneOntology hinterlegt bzw. referenziert ist, in der Spalte *func*. Die ID des Gens, die von der GeneOntology zugewiesen wird, wird in der Spalte *go_id* gespeichert. Das Feld *kind* beinhaltet Daten darüber zu welcher Teilontologie der GeneOntology (siehe Kapitel 2.3.3) das Gen gehört. So steht „P“ für biologische Prozesse, „C“ für Zellkomponenten und „F“ für molekulare Funktionen. Als Beispiel sollen die Daten in Tabelle 3.6 dienen.

| gene | func | go_id | kind |
|---|---|---|---|
| A4GALT | membrane fraction | GO:0005624 | C |
| A4GALT | glycosphingolipid biosynthetic process | GO:0006688 | P |
| A4GALT | plasma membrane organization and biogenesis | GO:0007009 | P |
| A4GALT | integral to Golgi membrane | GO:0030173 | C |
| A4GALT | galactosyltransferase activity | GO:0008378 | F |

Tabelle 3.6.: Beispiele von Genfunktionen an Hand des Gens *A4GALT*

Wie in der Tabelle 3.6 ersichtlich wurden dem Gen *A4GALT* fünf „Funktionen“ zugewiesen, darunter zwei Funktionen innerhalb biologischer Prozesse, zwei Funktionen als zelluläre Komponente und eine molekulare Funktion. Die Speicherung der Funktion ist insofern wichtig, da die Genfunktion einen Ausgangspunkt für die spätere Gewichtung und Klassifizierung von Genen im Rahmen des Clusterings genetischer Daten darstellt. Die GO-ID dient dazu, die Gene in den anderen genbezogenen Tabellen leicht referenzieren zu können.

Zur Speicherung von möglichen Synonymen und folglich zur Disambiguierung von Gennamen wurde die Tabelle *gene_synonyms* erstellt, deren Aufbau in Tabelle 3.7 gegeben ist. Sinn und Zweck dieser Tabelle ist es, für das Clustering genetischer Daten eine Grundlage zu liefern um Gene und deren Benamungen zu analysieren. Als Beispiel für die Daten, die in Tabelle 3.7 gespeichert werden soll die Übersicht in 3.8 dienen. Wie in obigem Beispiel wurde das Gen *A1GALT* gewählt. Wie in der Beispieltabelle ersichtlich ist, wurde dem Gen *A4GALT* lediglich ein Synonym zugewiesen, nämlich *A14GALT*. Es kann allerdings auch vorkommen, dass einem Gen mehr als ein Synonym zugewiesen wurde, wie zum Beispiel dem Gen *MOCS2*, dem zwei Synonyme (*MCBPE* und *MOCO1*) zugeordnet sind. Dieser Fall tritt allerdings nur selten ein. Zum derzeitigen Stand der Daten (Januar 2008) wurden lediglich 4 Genen mehr als ein Synonym zugewiesen.

Die letzte Datenbanktabelle zur Speicherung von Metainformationen über Gene ist *goid_level* , in der die Tiefe von Genen innerhalb der Taxonomie der GeneOntology gespeichert wird. Diese Daten werden ebenfalls für den Gewichtsvektor von

| gene | synonym |
|---|---|
| VARCHAR(14) | VARCHAR(30) |

Tabelle 3.7.: Tabelle zur Speicherung der Synonyme von Genen

| gene | synonym |
|---|---|
| A4GALT | A14GALT |

Tabelle 3.8.: Beispiel für Synonyme von Genen an Hand des Gens *A4GALT*

Genen und damit für das Clustering benötigt (siehe dazu Kapitel 4.8). Der Aufbau der Relation *goid_level* ist in Tabelle 3.9 schematisch dargestellt. Als Beispiel der gespeicherten Daten soll Tabelle 3.10 dienen, in der die GO-IDs des Gens „A4GALT“ zusammen mit dem jeweiligen Level innerhalb der GeneOntology aufgeführt werden. Die Daten bzgl. der Level eines Genes wurden mit Hilfe des Perl-Pakets

Das komplette Datenbankschema der Gendatenbank ist in Grafik 3.4 ersichtlich. Auf Grund der Einbindung von anerkannten Daten aus der GeneOntology entsteht zwar einerseits in heterogenes Datenmodell, andererseits kann dieses Datenmodell als Ausgangspunkt für die Entwicklung eines Systems für bioinformatisches Information Retrieval genutzt werden. Einen ähnlichen Ansatz mit heterogenen Datenquellen zur Entdeckung neuen medizinischen Wissens verfolgen etwa auch Wehbe u. a. (2009), die wissenschaftliche Aufsätze, Datenmengen, statistische Modelle und Algorithmen und deren Ergebnisse in einem System für Information Retrieval bündeln. Auch Lim u. a. (2008) verfolgen das Ziel einer umfassenden Datenquelle für genetische Informationen, die von unterschiedlichen Datenquellen gespeist wird.

| go_id | level | kind |
|---|---|---|
| VARCHAR(10) | SMALLINT | VARCHAR(1) |

Tabelle 3.9.: Tabelle zur Speicherung der Tiefe von Genen in der GO-Taxonomie

| go_id | level | kind |
|---|---|---|
| GO:0005624 | 2 | C |
| GO:0030173 | 121 | C |
| GO:0008378 | 1 | F |
| GO:0006688 | 26 | P |
| GO:0007009 | 1 | P |

Tabelle 3.10.: Beispiel der Tiefe von Genen in der GO-Taxonomie des Gens A4GALT

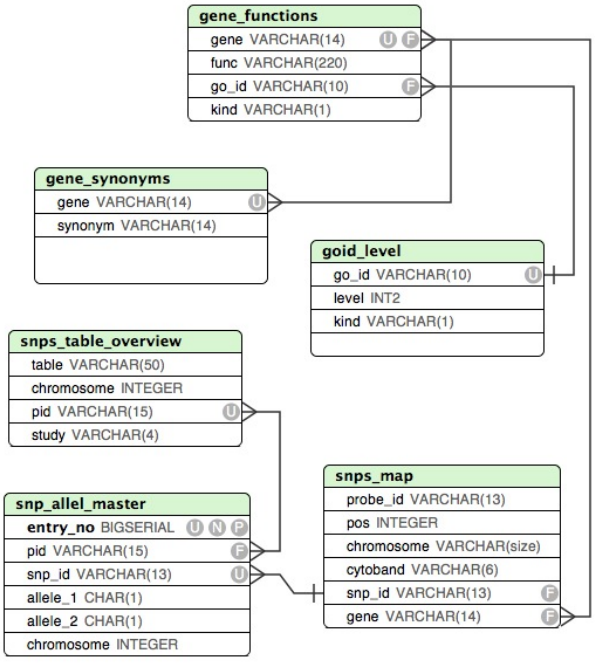



Abbildung 3.4.: Datenbankschema der Gendatenbank

Nach der Beschreibung der entworfenen Datenmodelle wird in den folgenden Abschnitten die darauf aufbauende Benutzerschnittstelle näher erläutert. Dieser Erläuterung geht allerdings eine allgemeine Einführung in die Thematik der *User-Interface Management Systeme* (UIMS) voraus. Hierbei wird vor allem der Sinn und Zweck, die Vorteile sowie die Voraussetzungen von UIMS erörtert.

## 3.5. User-Interface Management Systeme (UIMS)

Da in den folgenden Kapiteln das Design und die Implementierung einer abstrakten und zur Laufzeit generierten Benutzerschnittstelle für Datenbankanwendungen beschrieben wird, soll im folgenden Abschnitt der Sinn und Zweck sowie Voraussetzungen und derzeit existierende Frameworks zur Generierung abstrakter Benutzerschnittstellen beschreiben werden.

UIMS sind als Architekturmodelle in der Softwareentwicklung zu verstehen, bei der die eigentliche Anwendungsoberfläche von der darunterliegenden Anwendungssemantik getrennt ist. Dies hat vor allem zum Ziel, dass die Entwicklung von Benutzerschnittstellen beschleunigt wird, da nicht für jede Änderung oder Neuentwicklung einer Anwendungsoberfläche die Applikationssemantik neu programmiert werden muss, sondern lediglich die Beschreibung der Benutzerschnittstelle angepasst werden muss, was einen enormen Zugewinn an Flexibilität im Vergleich zu herkömmlichen GUI-Toolkits darstellt. Ein weiterer Vorteil von UIMS ist, die große Wiederverwendbarkeit bzw. Portierbarkeit und die Personalisierbarkeit, also die Anpassung an die Bedürfnisse eines Benutzers bzw. von Benutzergruppen, einer Benutzerschnittstelle. Desweiteren ermöglichen UIMS, dass die entsprechende Anwendung mit verschiedenen Benutzerschnittstellen kombiniert bzw. erweitert wird ohne dass dabei die eigentliche Anwendungslogik geändert werden muss.

Grundlegend muss gesagt werden, dass damit es sich um eine wirklich abstrakte Benutzerschnittstelle handelt, deren Beschreibung nicht auf bestimmte Plattformen und Eingabemöglichkeiten beschränkt sein darf (siehe etwa Richter (2007, 53)).

Das heisst, dass dieselbe Beschreibung sowohl auf herkömmlichen Desktoprechnern als auch auf mobilen Endgeräten und mit Geräten mit anderweitigen Eingabeformen wie etwa Touchscreens einsatzbar sein muss. Im Falle von GUI4DB (siehe Kapitel 3.6) stellt dies kein großes Problem dar, da Java eine der weit verbreitetsten plattformunabhängigen Programmiersprachen ist und auch für mobile Plattformen zur Verfügung steht.

Als Voraussetzungen können folgende Punkte ergänzend zur eben erwähnten Plattformunabhängigkeit in Betracht gezogen werden (siehe Trewin u. a. (2003)):

- Personalisierbarkeit
- Flexibilität
- Erweiterbarkeit
- Einfachheit

Obwohl schon seit längerer Zeit immer wieder Ansätze für leicht wartbare *user interface management systems* (UIMS) gemacht wurden, die Benutzerschnittstellen zum einen auf der Basis von ER-Diagrammen erstellen (vgl. Janssen u. a. (1993)) und zum anderen auf der Basis von XML (Puerta u. Eisenstein (2002)), reichte keiner dieser Ansätze aus, um auch die für dieses Projekt geplanten Regeln und Überprüfungen mit Bezug zur Datenbank mit in die Benutzerschnittstellenbeschreibung einzubinden.

Die oben erwähnten Punkte wurden im Falle von GUI4DB (siehe Kapitel 3.6) umgesetzt indem zum einen die Benutzerschnittstelle von den Benutzern mit Hilfe einer simplen Drag-and-Drop-Anwendung erstellt und somit auf deren Bedürfnisse angepasst werden kann, zum anderen ist das Softwareframework ohne großen Aufwand erweiterbar. Desweiteren ist das Framework insofern flexibel, als es mit Hilfe des Drag-and-Drop Werkzeugs zur Benutzerschnittstellendefinition auch an alle Benutzerschichten angepasst werden kann. Ferner zeichnet es sich durch ein hohes Maß an Einfachheit aus, da keine schwer bereitzustellenden Ressourcen benötigt werden und die zu Grunde liegende Technik (Java und XML) zum einen weit

verbreitet ist und zum anderen vor allem im Falle von XML auch leicht verständlich ist.

Derzeit gibt es natürlich mehrere demenstprechende ähnliche Frameworks zur Beschreibung und Generierung von Benutzerschnittstellen, auf die im Folgenden näher eingegangen wird.

#### User Interface Markup Language (UIML)

UIML ist ein Standard auf Basis von XML zur Beschreibung von Benutzerschnittstellen, der von der *organization for the advancement of structured information standards* (OASIS) betreut wird. UIML stellt dabei ein „puzzle piece to be used in conjunction with other technologies“ (OASIS (2008)) dar, das auf ein abstraktes XML-Vokabular zur Beschreibung von Benutzerschnittstellen setzt, das unabhängig von Programmiersprachen eingesetzt werden kann. Allgemein kann zu UIML gesagt werden, dass eine Oberfläche einer Applikation an Hand von Elementen (structure), Layout- und Designattributen (style) und Interaktionen (behavior) beschrieben wird. Die Darstellung selbst ist dabei allerdings Aufgabe der eigentlich Applikation, die dafür verantwortlich ist, die GUI-Beschreibung in eine konkrete Benutzeroberfläche umzuwandeln.

Als kleines Minimalbeispiel soll dazu Listing 3.1 dienen.

Listing 3.1: UIML-Schema zur Benutzerschnittstellenbeschreibung

```
1 <?xml version="1.0"?>
2 <!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0 Draft//EN"
    "http://uiml.org/dtds/UIML3_0a.dtd">
3
4 <uiml>
5   <interface>
6     <structure>
7       <part id="TopHello">
8         <part id="hello" class="helloC"/>
```

```
 9          </part>
10        </structure>
11      </interface>
12  </uiml>
```

Dieses Beispiel stellt insofern nur ein Minimalbeispiel dar, da lediglich „Hello World“ auf einer Benutzerschnittstelle erscheint. UIML bietet allerdings noch etliche weitere Elemente, um vorgefertigte UI-Beschreibungen einzubinden, das Aussehen bzw. Layout und das Verhalten von GUIs zu beschreiben, etc. Darüber hinaus können auch Regeln in UIML beschrieben werden. Dies dient dazu Aktionen festzulegen, die durchgeführt werden sollen, sobald ein bestimmter Zustand eintritt.

#### Extensible Interface Markup Language (XIML)

XIML ist ebenfalls ein auf XML basierendes Beschreibungsformat für abstrakte Benutzerschnittstellen. XIML besteht dabei aus folgenden fünf Komponenten (vgl. Puerta u. Eisenstein (2002) bzw. Trewin u. a. (2003)):

- *Tasks*: Benutzerprozesse und -aufgaben, die die Zielplattform unterstützen.
- *Domain*: Eine Hierarchie von Datenobjekten, die die Benutzer entweder nur sehen oder sehen und manipulieren können.
- *User*: Eine Hierarchie von User-Elementen, zur Speicherung von userbezogenen Informationen, die für die Benutzerschnittstelle relevant sind.
- *Presentation*: Eine Hierarchie von konkreten UI-Elementen (z.B. Buttons, Textlabels, etc.).
- *Dialog*: Eine Beschreibung der möglichen Interaktionen.

Wie aus obiger Auflistung hervorgeht, dient XIML nicht nur zur reinen Beschreibung von Benutzerschnittstellen, sondern auch von Operationen, die zur Laufzeit

durchgeführt werden können. Genauso wie bei UIML auch, ist jedoch der Programmierer wieder für die Umwandlung von der XIML-Beschreibung hin zu einer funktionsfähigen Applikation verantwortlich.

#### XForms

XForms ist eine auf XML basierende Spezifikation zur Beschreibung von Formularen auf Webseiten, die vom W3C Konsortium entwickelt wurde (siehe W3C (2009)), um eine einheitliche Beschreibungsgrundlage für Webformulare zu bieten. Ein XForms-Dokument besteht aus folgenden drei Komponenten, wie sie auch in modernen objektorientierten Programmiersprachen zum Einsatz kommen:

- Model: Beschreibungen der zu Grunde liegenden Daten.
- View: Die konkrete Darstellung der im Model beschriebenen Daten.
- Controller: Eine Zwischenschicht zwischen Model und View, die die Datenbearbeitungen, die im Model durchgeführt und in der View einleitet werden, anstößt und überwacht.

Im Gegesatz zu dem in diesem Projekt entwickelten GUI-Framework GUI4DB (siehe Kapitel 3.6) bieten alle hier erwähnten Beschreibungsformate keine direkte Möglichkeit auf Datenbanken zuzugreifen. Desweiteren ist es nicht möglich einzelne Elemente 1 zu 1 mit Feldern oder Tabellen in Datenbanken in Beziehung zu setzen und auch keine Datenbankregeln bzw. -aktionen wie etwa das Ausführen von Abfragen angegeben werden können, was eine Grundvoraussetzung für die Implementierung von GUI4DB war. Darüber hinaus besitzen alle XML-Beschreibungen ein sehr komplexes Vokabular, was es einem Benutzer ohne fundierte Kenntnisse von Benutzerschnittstellendesign und XML schwer macht ein leistungsfähiges Formular zu erstellen.

## 3.6. GUI4DB - Generic User Interface for Databases

Wie in Kapitel 3.5 erwähnt dienen *User-Interface Management Systeme* zur Erstellung von hoch flexiblen, personalisierbaren und erweiterbaren Benutzerschnittstellen. Daher wurde für das hier vorgestellte Softwareframework zur Datenerhebung in klinischen Studien ebenfalls dieser Architekturansatz gewählt. Im folgenden wird daher das Design bzw. die Implementierung sowie der Aufbau des Softwareframeworks näher beschrieben. Es wird ebenfalls auf die ergonomische Aspekte und auf Vorteile gegenüber herkömmlichen Desktopdatenbankanwendungen eingegangen.

### 3.6.1. Motivation

Der Grund für die Erstellung eines generischen Datenbankinterfaces bestand darin, dass Formulare zur Dateneingabe in Datenbanken entweder nur proprietäre Lösungen sind und auf eine bestimmte Datenbank angewiesen sind, diese bzw. die Formularerstellung äusserst komplex oder die Anwendungen an eine bestimmte Betriebssystemplattform angewiesen sind. Daher war ein Ziel dieser Anwendung dem Benutzer eine datenbank- und plattformunabhängige Applikation zur Verfügung zu stellen, mit Hilfe derer er ohne großen Aufwand und ohne Programmierkenntnisse Formulare erstellen kann, die wiederum eine stabile und sichere, die Datenintegrität wahrende, Speicherung von Daten ermöglichen.

Desweiteren war es ein Anliegen viele der in Kapitel 3.3 erwähnten Probleme zu beheben ohne den Administrations- oder Programmieraufwand seitens der Benutzer zu erhöhen. So sollte vor allem die Integrität der Daten gewährleistet sein, das heisst, dass zur Speicherung der Daten zum einen die Daten in Transaktionen gekapselt gespeichert werden und zum anderen Plausibilitätsprüfungen durchgeführt werden.

Natürlich war es auch, wie bereits erwähnt, ein Ziel eine zentrale Datenspeicherung und eine Dateneingabe, die es mehreren Benutzer ermöglicht, zeitgleich Daten einzugeben, einzuführen. Dies hatte den Zweck, dass Inkonsistenzverletzungen auf

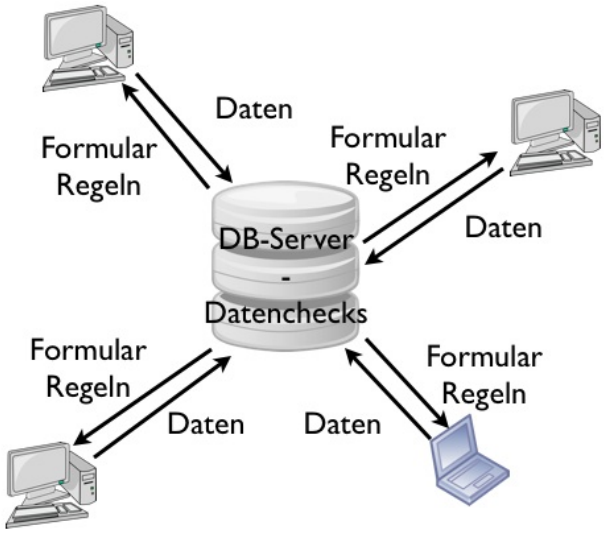



Abbildung 3.5.: Schematischer Aufbau der zentralen Datenspeicherung mit GUI4DB

Grund nebenläufiger Datenmanipulationen zu vermeiden sowie ein allgemeingültiges Datenmodell, das für alle Benutzer jederzeit zugänglich ist, zu bieten, um so Insellösungen zu vermeiden (siehe dazu Kapitel 3.3). Schematisch sind diese eben erwähnten Punkte in der Grafik 3.5 dargestellt. Dabei ist eine zentrale Datenspeicherung ersichtlich, sowie das Laden der Formulare inklusive von Plausibilitätsregeln in Form von XML (W3C (2008)) auf alle zugangsberechtigten Rechner, welche dann wiederum plausible und datentechnisch einwandfreie Daten an einen zentralen Datenbankserver schicken.

Da eine Datenbank ohne die Möglichkeit der Datenabfrage ihren Sinn verfehlt hätte, sollte auch die Möglichkeit bestehen, Daten aus der Datenbank zu holen und diese weiterzuverarbeiten. Daher war es notwendig die Möglichkeit zu bieten sowohl Abfragen durchführen zu können als auch die erhaltenen Daten in bestimmte gängige Dateiformate exportieren zu können, um sie später auszutauschen bzw. ge-

gebenenfalls mit anderen Softwareprodukten zu analysieren. Derzeit besteht daher die Möglichkeit Daten in folgende Formate zu exportieren:

- CSV (comma-separated values)
- Excel
- Attribute-Relation File Format (ARFF) zur Verwendung mit Data-Mining Applikationen

Um Zugriff auf die Daten zu erlangen, besteht zum einen die Möglichkeit vorgefertigte Abfragen, sogenannte *views*, grafisch zu definieren und in der Datenbank zu speichern und zum anderen bereits gespeicherte *views* aufzurufen. Die so ermittelten Daten werden dann in Form einer Tabelle angezeigt. Der Benutzer hat darüber hinaus noch die Möglichkeit in diesen Tabellen Daten zu bearbeiten, neue Daten anzulegen und in den angezeigten Daten zu suchen sowie sie zu exportieren.

### 3.6.2. Softwareentwicklungsprozess

Im folgenden wird auf den generellen Softwareentwicklungsprozess eingegangen und wie dieser hier praktisch umgesetzt wurde. Ein wichtiger Aspekt des Softwareentwicklungsprozesses ist das Einbinden der späteren Benutzer bzw. der „Anforderungsbeitragenden“ (Oestereich (2001, 90)). Dabei ist es essentiell, dass alle Benutzer identifiziert werden, denn andernfalls „steigt die Wahrscheinlichkeit, daß [...] eine Reihe von Anforderungen an das System nicht oder nicht rechtzeitig“ (Oestereich (2001, 90)) erkannt werden. Beim benutzerzentrierten Design und der Implementierung vor allem komplexer Softwaresysteme sind folgende Aspekte bzw. Schritte für eine erfolgreiche Umsetzung und den erfolgreichen Einsatz der Software wichtig (siehe Richter (2007, 140)):

- Anforderungsanalyse: Analyse der Benutzeraufgaben, eingesetzten Plattform, technischen Anforderungen

- Konzeptuelles Design der Software
- Entwicklung und Testen der Software
- Installation und Feedback der Benutzer

Ziel des ersten Schrittes ist es, eine Spezifikation des zu entwickelnden Softwaresystems zu erstellen, die sowohl die Aufgaben und Eigenschaften der Benutzer sowie technische Aspekte der zu Grunde liegenden Systemarchitektur beinhalten soll. Hierbei ist vor allem ein intensiver Kontakt zu den Benutzern essentiell, das heisst, dass der Benutzer und dessen Anforderungen im Zentrum des Softwareentwicklungsprozesses stehen sollten, um so ein möglichst benutzerfreundliches Produkt zu entwickeln. Im zweiten Schritt des Entwicklungsprozesses werden die Anforderungen an die Gesamtsoftware in implementier- und leicht wartbare Softwaremodule unterteilt und erste Designstudien der Anwendung erstellt. Diese Designstudien sollten zusammen mit den Anwendern iterativ verfeinert werden, die dann im nächsten Schritt implementiert werden. Während der Entwicklung ist es beim benutzerzentrierten Entwicklungsprozess wichtig ebenfalls die Benutzer so stark wie möglich in die Entwicklung einzubinden, was durch Tests und Vorführen implementierter Softwaremodule erreicht werden kann. Der letzte Punkt ist vor allem für den langfristig erfolgreichen Einsatz der Software wichtig. In diesem Schritt wird die komplette Anwendung installiert was von intensiven Tests seitens der Benutzer gefolgt werden sollte. Das Feedback der Benutzer dient dann wiederum dazu, die Software weiterzuentwickeln bzw. besser an die Wünsche der Anwender anzupassen.

Daher wurde während der Planung und Entwicklung in enger Zusammenarbeit mit den beteiligten Personen, Schritt für Schritt eine Liste an Anforderungen erstellt und diese dementsprechend umgesetzt. Damit sichergestellt wurde, dass die Anforderungen korrekt umgesetzt wurden und um eventuell dabei aufgetretene Probleme zeitnah zu beheben, wurden die verantwortlichen User ebenfalls in den Design- und Implementierungsprozess der Erstellung von GUI4DB eingebunden. Das heisst,

dass die User von Anfang an die Probleme, die sie beim Einsatz der bis dahin benutzten Software sahen, aufzeigen konnten und gleichzeitig Vorschläge zur Behebung der Probleme einbringen konnten. Ebenfalls wurden generelle Probleme der bis dahin eingesetzten IT-Infrastruktur, wie etwa der dezentralen Datenspeicherung, erörtert, um diese im Falle von GUI4DB zu vermeiden und Lösungen, die besser für den Einsatz in klinischen Studien geeignet sind, zu erstellen, denn „the success of a software system depends on how well it fits the needs of its users and its environment“ (Cheng u. Atlee (2007)). So wurde sichergestellt, dass die Nachteile einer Massensoftware vermieden wurden und diese durch speziell an die Bedürfnisse der User angepasste Software ersetzt wurden.

Neben den eben erwähnten Punkte ist auch eine genaue Analyse der Einsatzszenarien von großer Bedeutung, da die Software auf den genauen Einsatz angepasst werden sollte, um zum einen eine mögliche Umstellung des Softwaresystems und zum anderen einen reibungslosen Betrieb der Software zu ermöglichen. So ist für diese Software ein netzwerkbasierter Einsatz vorgesehen, in dem von mehreren Studienrechnern auf eine zentrale Datenbnk zugegriffen wird. Im konkreten Einsatz der Software sollen Daten, die entweder bereits auf medizinischen Anamnesebögen vorhanden sind oder in Telefoninterviews erhoben werden, eingegeben werden. Dabei stellt vor allem die Eingabe von Daten während Telefoninterviews eine besondere Herausforderung dar, da die Anwendung besonders stabil und performant arbeiten muss. Daher ist es wichtig, dass nicht der komplette Datensatz eines Patienten eingegeben werden muss bis kritische Plausibilitätsprüfungen durchgeführt werden, sondern diese sofort während der Eingabe durchgeführt und Probleme angezeigt werden.

Desweiteren ist eine ausführliche Systemanalyse notwendig, damit die Software ohne Probleme in ein bestehendes System integriert werden kann. In diesem Fall musste die Software in ein klinikinternes Netz integriert werden. Da kein spezielles Datenbanksystem vorgegeben war, fiel die Wahl dabei auf das freie RDBMS PostgreSQL 8.2. Als Programmiersprache bzw. als Laufzeitumgebung wurde das Java Development Kit 6 der Firma Sun Microsystems gewählt, da dies einen plattform-

übergreifenden Einsatz von Software und eine unkomplizierte Datenbankanbindung mit Hilfe von *Java Database Connectivity* (JDBC) ermöglicht. So stellen alle namhaften Datenbankhersteller speziell für ihre Datenbankserver programmierte Java-Schnittstellen zur Verfügung, die alle auf den im Java-Paket java.sql beruhen. Das hat den Vorteil, dass für alle Datenbanktreiber dieselben Funktionen gültig sind, weswegen nicht für jede Datenbank eine eigene Anbindung programmiert werden muss. Neben der Plattformunabhängigkeit und der einfachen Datenbankanbindung war ein weiterer Grund für die Wahl des JDK 6, dass es mit Java einfach ist, XML-Daten zu lesen und zu bearbeiten, was für die Erstellung der eigentlichen Benutzerschnittstelle essentiell ist. Hierzu finden sich wichtige Schnittstellen im Java-Paket org.w3c.dom bzw. javax.xml, die es ermöglichen komplette XML-Dokumenten in ein *Document Object Model* zu lesen und daraus die notwendigen Daten zu extrahieren. Darüber hinaus bietet das JDK6 wichtige Funktionen im Paket javax.swing, um Benutzerschnittstellen zu erzeugen bzw. zu konfigurieren und in ein flexibles jedoch konsistentes UI-Layout zu bringen.

### 3.6.3. Beschreibung von GUI4DB

Nach der allgemeinen Beschreibung des Softwareentwicklungsprozesses und des eingesetzten Systems, wird im folgenden Abschnitt die eigentliche Software näher beschrieben. Um die Eingabe der im Rahmen einer Studie erhobenen Patientendaten so einfach wie möglich zu gestalten, wurde ein Java-basiertes Framework entwickelt, das es zum einen ermöglicht, Formulare zur Dateneingabe ohne tiefe Datenbankkenntnisse und ohne Programmiererfahrung zu erstellen, und zum anderen diese Formulare unabhängig von den jeweiligen Arbeitsplätzen zentral in einer Datenbank zu speichern und zur Laufzeit dynamisch zu generieren. Desweiteren ist es möglich über sogenannte *views* Daten abzufragen und visuell in Tabellen anzuzeigen. Da dieses Framework in der plattformunabhängigen Programmiersprache Java implementiert wurde, kann es auf allen gängigen Betriebssystemenen und mit allen Datenbanken, die einen JDBC (Java database connectivity) Treiber liefern ein-

gesetzt werden. Die einzige Voraussetzung zur Benutzung der Applikation ist eine Datenbank, die Transaktionen und verzögerte Integritätsprüfungen, d.h. Integritätsprüfungen nur am Ende einer Transaktion, unterstützt, da die Daten in Form von Transaktionen gespeichert werden, um so die Integrität der Daten aufrecht zu erhalten. Ein weiterer Vorteil ist, dass bei der Dateneingabe mittels Formularen die Datenbank per se nur noch als rein abstrakter Datenspeicher verwendet wird (siehe dazu Grafik 3.2), was für den Benutzer die Komplexität der Anwendung versteckt und sie somit bedienbarer macht.

Darüber hinaus ist es möglich schon in das Formular selbst Plausibilitätsprüfungen einzubauen. So können zum Beispiel Minimum- und Maximumwerte spezifiziert werden, die vor der eigentlichen Datenspeicherung überprüft werden. Desweiteren besteht die Möglichkeit datenbankbasierte Plausibilitätsprüfungen von Werten bzw. Abfragen, deren Ergebnisse angezeigt werden, durchzuführen, sobald ein Wert in ein Feld eingegeben wurde. Dies macht vor allem dann Sinn falls numerische IDs bzgl. ihrer Korrektheit überprüft werden sollen oder festgestellt werden soll, ob eindeutige Werte unter Umständen schon in der Datenbank vorhanden sind. Dies erhöht schon bei der Eingabe der Daten die Korrektheit und spart vor allem auch Arbeitszeit, da nicht notwendigerweise alle Daten eingegeben werden müssen bis eine Integritäts- bzw. Plausibilitätsprüfung durchgeführt werden kann.

Die Anwendung GUI4DB besteht aus den folgenden drei Kernkomponenten:

- Datenbankoperationen
- Berechnung und Generierung der Benutzerschnittstelle und
- XML-Operationen

Die wichtigste der eben erwähnten drei Kernkomponenten stellt sicherlich die Komponente der Datenbankoperationen dar. So werden in der Klasse *DBHandler* die wichtigsten und elementarsten Datenbankoperationen gebündelt. Mit ihrer Hilfe werden zum einen Transaktionen gestartet und abgeschlossen sowie alle gängigen (Meta-)Daten aus der Datenbank abgefragt. Dies hat den großen Vorteil, dass alle

Datenbankoperationen einheitlich durchgeführt werden können ohne eigene Implementationen von datenbankbasierten Operationen erstellen zu müssen. Auf diese Art und Weise ist es möglich, ohne großen Aufwand die Benutzerschnittstelle von den eigentlich Datenbankoperationen zu trennen auch wenn später die Anwendung erweitert werden sollte, da es genügt die vorgefertigten Methoden auszuführen. Durch die einheitliche Bündelung der Datenbankoperationen lassen sich desweiteren mögliche Probleme in Form von Inkonsistenzen in der Datenbank vermeiden, da vor allem durch den obligatorischen Einsatz von Transaktionen zur Durchführung von SQL-Befehlen das Problem inkompletter und damit inkonsistenter Daten vermieden werden kann. So werden zum Beispiel alle Speicheroperationen in Transaktionen gekapselt, um nur komplette Datensätze in der Datenbank zu speichern.

Da die Benutzerschnittstelle zur Datenbank dynamisch aus XML erstellt wird, was ein hohes Maß an Flexibilität bietet, stellen die XML-Operationen eine ebenfalls wichtige Kernkomponente dar. Um die Benutzerschnittstelle dynamisch zu generieren ist eine Beschreibung in Form von XML notwendig, die dem Schema B.7 unterliegen muss. Wie in Listing B.7 ersichtlich ist, ist das zentrale Element des XML-Schemas das Element *panel*. Dieses Element entspricht einem Reiter in der tatsächlichen Benutzerschnittstelle und muss mindestens einmal vorkommen ohne dass eine maximale Anzahl von *panel* definiert wird. Ein *panel*-Element bzw. ein Reiter beinaltet wiederum alle UI-Elemente zur Dateneingabe wie Textfeld, Dropdown-Liste oder einen Button zum Starten eines Plugins. Ein Reiter muss allerdings mindestens eines dieser Elemente enthalten.

Um nun die Benutzerschnittstelle zu generieren, wird zuerst die GUI-Beschreibung entweder aus der Datenbank oder einer XML-Datei gelesen und auf ihre Validität hin überprüft. Im nächsten Schritt werden alle UI-Elemente wie Reiter, Textfelder, Dropdown-Listen und Plugins/Buttons extrahiert um daraus die entsprechenden Elemente in der Benutzerschnittstelle zu generieren. Dabei werden alle notwendigen Eigenschaften (siehe XML-Schema B.7) in Unterklassen der in Java im Paket *javax.swing* vorhandenen UI-Elemente gespeichert, um so leicht Zugang auf sie

zu haben sobald die Attribute benötigt werden. So ist es möglich auf einfache Art und Weise Datenbankinformationen in der Benutzerschnittstelle zu speichern und somit die Benutzerschnittstelle mit den entsprechenden Tabellen und Feldern in der Datenbank zu verbinden. Sobald alle UI-Elemente aus der XML-Beschreibung extrahiert und erstellt wurden, wird das eigentliche Formular in einem *JInternalFrame* innerhalb eines *JDesktopPane* angezeigt.

Die Datenbankinformationen wie etwa die Zieltabelle bzw. das Zielfeld werden in speziellen Unterklassen gespeichert, die die Standard-Javaklassen erweitern. So wurden für diesen Zweck folgende Klassen implementiert:

- *XTextField*
- *XComboBox*
- *XButton*

Da die Vererbungslinien für Textfelder und Comboboxen unterschiedlich sind und in Java Mehrfachvererbung nicht möglich ist, besitzen leider die beiden Klassen *XTextField* und *XComboBox* zu einem großen Teil die selben Attribute. So besitzen beide Klassen folgende Eigenschaften:

- ***protected*** *XObject.TYPE _type* - Speichert den Datentyp der einzugebenden Werte
- ***protected*** *String _preparedQuery* - Speichert eine Abfrage, die ausgeführt werden soll, sobald ein Wert eingegeben wurde.
- ***protected*** *TreeMap<String, String> _tableFields* - Speichert die Tabellen und die entsprechenden Felder, in die der Wert geschrieben werden soll.
- ***protected boolean*** *_isUnique* - Gibt an, ob der Wert, der eingetragen wurde auf Eindeutigkeit überprüft werden soll.

Über diese Eigenschaften hinaus, besitzt die Klasse *XTextfield* noch Attribute, um optionale Minimal- und Maximalwerte zu speichern, sowie anzugeben ob eine Wertangabe zwingend erforderlich ist oder nicht. Alle diese Attribute werden bei den beiden eben erwähnten Klasse während der Erstellung der Benutzerschnittstelle gesetzt. Zusätzlich werden die Elemente, die in der Combobox erscheinen sollen, ebenfalls aus der XML-Beschreibung gelesen und gespeichert. Zum Laden von externen Plugins besitzt die Klasse *XButton* folgende notwendige Attribute:

- ***protected*** *String _pluginEntryPoint* - Gibt den voll qualifizierenden Namen der Klasse an, die das *Plugin*-Interface implementiert und die zum eigentlichen Starten des Plugins dient.
- ***protected*** *TreeMap<String, String>* *_parameters* - Eine Datenstruktur zum Speichern von optionalen Parametern, die an das Plugin übergeben werden können. Dabei dient der Parametername als Schlüssel und der eigentliche Parameter als Wert.

Jedes dieser Elemente wird nach der Initialisierung und dem Setzen der Attributwerte zu einem *JPanel*-Objekt hinzugefügt. Das Layout wird danach dynamisch aus der Anzahl der vorhandenen Elemente berechnet, die wiederum so angeordnet werden, dass jeweils nur 3 Elemente inklusive deren Beschriftungen eine Reihe auf dem entsprechenden Panel bilden (siehe Abbildung 3.6), was die Übersichtlichkeit und damit die Benutzerfreundlichkeit und Effektivität der Applikation erhöhen soll.

Da das manuelle Erstellen einer GUI-Beschreibung in XML ein mühsamer und aufwendiger Prozess ist für den XML-Kenntnisse zwingend notwendig sind, gibt es zur Vereinfachung der Erstellung neuer Benutzerschnittstellen bzw. Formulare einen visuellen Editor (siehe Abbildung 3.7). Diese Erweiterung ermöglicht es dem Benutzer mittels einfacher „Drag and Drop“-Operationen Benutzerschnittstellen neu zu erstellen. Zur Angabe der benötigten Eigenschaften der einzelnen Elemente dienen Dialoge bzw. Eingabefelder an der rechten unteren Seite des Fensters.

Neben der Eingabe von Studienwerten und der Erstellung von Formularen, bietet GUI4DB auch die Möglichkeit Abfragen durchzuführen bzw. zu erstellen. Ähnlich

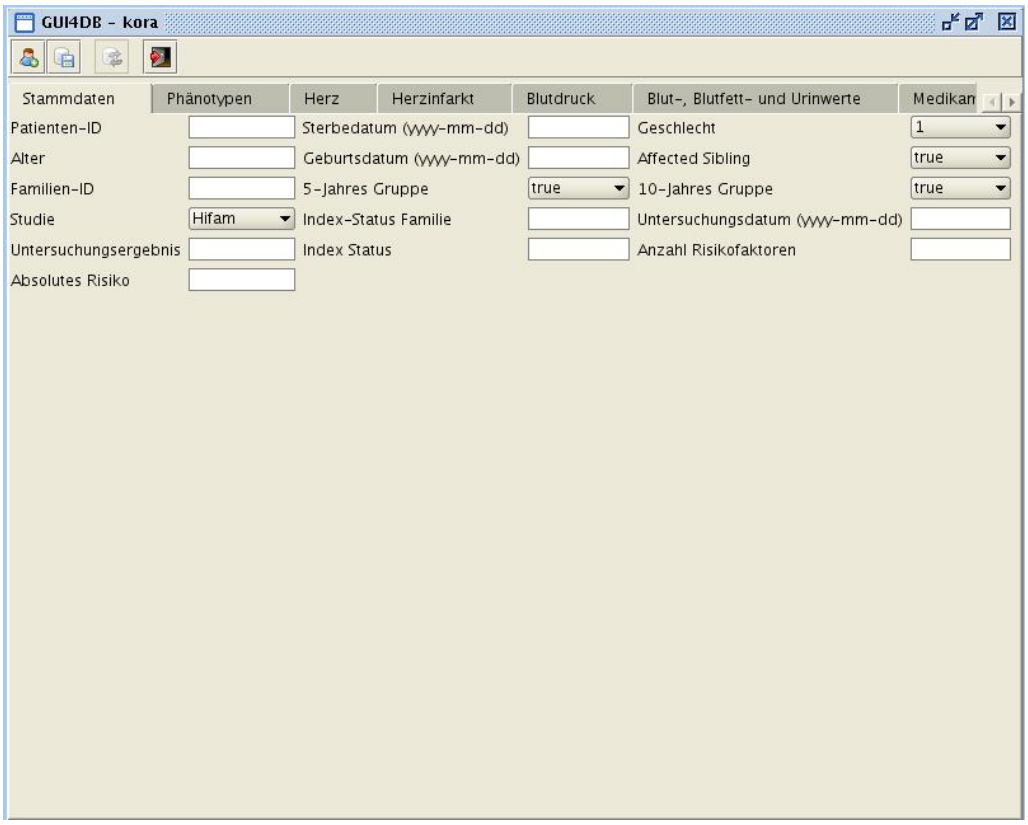



Abbildung 3.6.: Hauptfenster zur Dateneingabe in GUI4DB

wie bei Microsoft Access gibt es ein Plugin, das es dem Benutzer ermöglicht mit Hilfe von „Drag and Drop“ ohne große SQL-Kenntnisse Abfragen zu erstellen und diese auszuführen. Sollte der Benutzer diese Abfrage öfter ausführen wollen, so besteht die Möglichkeit die Abfrage in Form einer „View“ direkt in der Datenbank abzuspeichern. Unter „View“ versteht man dabei, dass eine Abfrage in Form einer virtuellen Tabelle in der Datenbank gespeichert wird und diese jedesmal neu mit den Werten der Abfrage gefüllt wird. Sollte diese „View“ updatebar sein, so besteht auch die Möglichkeit die Daten in der Tabellenansicht (siehe Abbildung 3.8) zu verändern oder den gesamten Datensatz in das Formular zu laden für eine bes-

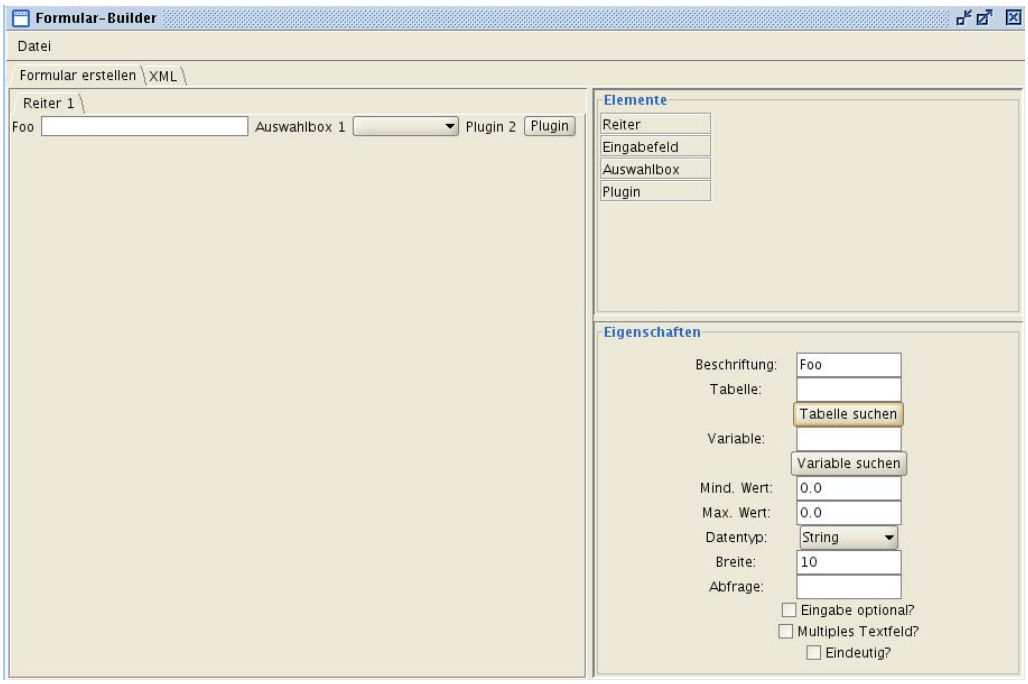
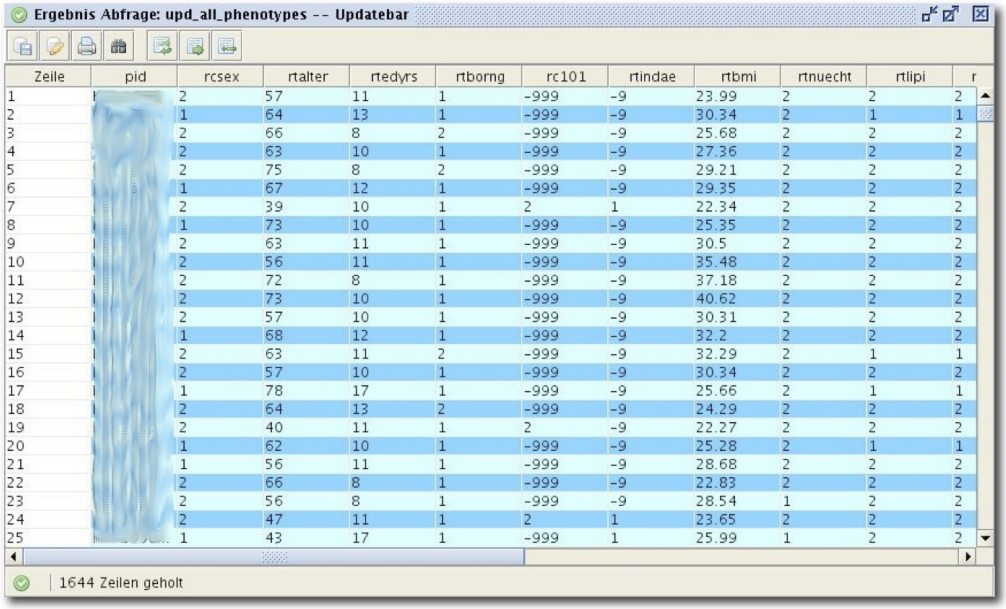



Abbildung 3.7.: Erstellung von Benutzerschnittstellen in GUI4DB

sere Übersicht über die Daten. Die geänderten Daten, sei es in der Tabelle oder im Formular, werden dann innerhalb einer Transaktion in der Datenbank gespeichert. Wie bereits in Kapitel 3.6.1 erwähnt besteht die Möglichkeit die Daten in gängige Dateiformate (CSV, Excel, ARFF) zu exportieren, um so Daten auszutauschen oder anderweitig zu analysieren.

### 3.6.4. Ergonomie- und Benutzerbarkeitsaspekte der Benutzerschnittstelle

Ein weiterer wichtiger Punkt neben der automatischen GUI-Generierung und der Einbindung von Regeln ist selbstverständlich die Usability sprich die Benutzerfreundlichkeit bzw. Benutzbarkeit der Applikation. Darunter versteht man „the ca-




| Zeile | pid | rcsex | rtalter | rtedyrs | rtborng | rc101 | rtindae | rtbmi | rtnuecht | rtlipi | r |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 2 | 57 | 11 | 1 | -999 | -9 | 23.99 | 2 | 2 | 2 |
| 2 | | 1 | 64 | 13 | 1 | -999 | -9 | 30.34 | 2 | 1 | 1 |
| 3 | | 2 | 66 | 8 | 2 | -999 | -9 | 25.68 | 2 | 2 | 2 |
| 4 | | 2 | 63 | 10 | 1 | -999 | -9 | 27.36 | 2 | 2 | 2 |
| 5 | | 2 | 75 | 8 | 2 | -999 | -9 | 29.21 | 2 | 2 | 2 |
| 6 | | 1 | 67 | 12 | 1 | -999 | -9 | 29.35 | 2 | 2 | 2 |
| 7 | | 2 | 39 | 10 | 1 | 2 | 1 | 22.34 | 2 | 2 | 2 |
| 8 | | 1 | 73 | 10 | 1 | -999 | -9 | 25.35 | 2 | 2 | 2 |
| 9 | | 2 | 63 | 11 | 1 | -999 | -9 | 30.5 | 2 | 2 | 2 |
| 10 | | 2 | 56 | 11 | 1 | -999 | -9 | 35.48 | 2 | 2 | 2 |
| 11 | | 2 | 72 | 8 | 1 | -999 | -9 | 37.18 | 2 | 2 | 2 |
| 12 | | 2 | 73 | 10 | 1 | -999 | -9 | 40.62 | 2 | 2 | 2 |
| 13 | | 2 | 57 | 10 | 1 | -999 | -9 | 30.31 | 2 | 2 | 2 |
| 14 | | 1 | 68 | 12 | 1 | -999 | -9 | 32.2 | 2 | 2 | 2 |
| 15 | | 2 | 63 | 11 | 2 | -999 | -9 | 32.29 | 2 | 1 | 1 |
| 16 | | 2 | 57 | 10 | 1 | -999 | -9 | 30.34 | 2 | 2 | 2 |
| 17 | | 1 | 78 | 17 | 1 | -999 | -9 | 25.66 | 2 | 1 | 1 |
| 18 | | 2 | 64 | 13 | 2 | -999 | -9 | 24.29 | 2 | 2 | 2 |
| 19 | | 2 | 40 | 11 | 1 | 2 | -9 | 22.27 | 2 | 2 | 2 |
| 20 | | 1 | 62 | 10 | 1 | -999 | -9 | 25.28 | 2 | 1 | 1 |
| 21 | | 1 | 56 | 11 | 1 | -999 | -9 | 28.68 | 2 | 2 | 2 |
| 22 | | 2 | 66 | 8 | 1 | -999 | -9 | 22.83 | 2 | 2 | 2 |
| 23 | | 2 | 56 | 8 | 1 | -999 | -9 | 28.54 | 1 | 2 | 2 |
| 24 | | 2 | 47 | 11 | 1 | 2 | 1 | 23.65 | 2 | 2 | 2 |
| 25 | | 1 | 43 | 17 | 1 | -999 | 1 | 25.99 | 1 | 2 | 2 |

Abbildung 3.8.: Darstellen der Ergebnisse einer Datenbankabfrage

pability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.“ (Bevan (2001)). Um ein hohes Maß an Usability zu erreichen sind dabei folgende Punkte von hoher Wichtigkeit (vgl. Bevan (2001):

- Funktionssicherheit
- Genauigkeit
- einfache Bedienbarkeit
- Effizienz
- Stabilität
- leicht verständlich

Um alle diese Punkte zu gewährleisten erfolgte die gesamte Phase des *requirements engineering*, also der Phase der Anforderungserhebung und -managements, sowie die Design- und Implementierungsphase in enger Zusammenarbeit mit den späteren Benutzern. Desweiteren wurden die Prototypen der Anwendung in regelmäßigen Abständen den Benutzern vorgeführt und mit ihnen besprochen, um so wichtiges Feedback für die weitere Entwicklung der Gesamtapplikation zu erhalten. Was in diesem Fall aber erschwerend hinzukam ist, dass GUI4DB eine generische Applikation ist, die aus mehreren einzelnen Komponenten bzw. Modulen besteht. Dahingehend war es eine Herausforderung die einzelnen Zusammenhänge der einzelnen Module untereinander und die Konsequenzen von bestimmten Anforderungen im Gesamtkontext zu vermitteln. Nichtsdestotrotz war die enge Zusammenarbeit und das direkte Einbeziehen der Benutzer ein wichtiger Faktor für die Entwicklung einer benutzerfreundlichen Applikation.

Im konkreten Fall wurden zur Verbesserung der Usability die oben erwähnten Punkte mit leicht verständlichen Fehlermeldungen umgesetzt, die sofort erscheinen sobald der Benutzer eine falsche Eingabe gemacht hat. Dies ist insofern wichtig, da eine fehlerhafte Eingabe in einem UI-Element die Speicherung der gesamten Daten verhindert. So kann der Benutzer zuerst die fehlerhafte Eingabe falls möglich korrigieren und danach fortfahren die restlichen Daten einzugeben. Darüber hinaus wurde bei der automatischen Generierung der Eingabeformulare darauf Wert gelegt, dass die Formulare soweit wie möglich übersichtlich bleiben. Selbstverständlich liegt dieser Aspekt vordergründig in der Hand des Benutzers, der das Formular erstellt. Jedoch werden die einzelnen Elemente in einem möglichst übersichtlichen GUI-Layout, das automatisch berechnet wird, angeordnet, um den Benutzern ein effektives Arbeiten zu ermöglichen und die Wahrscheinlichkeit, dass die Benutzer die Übersicht verlieren, zu minimieren. Um den Benutzer davor zu bewahren falsche Daten hinsichtlich des Datentyps bzw. des möglichen Wertebereichs einzugeben, erscheinen Tooltips sobald die Maus über ein Eingabelement bewegt wird, in denen der Datentyp bzw. der Wertebereich angezeigt werden. Um Plugins einzubinden ohne die Anwendung bzw. Formulare unnötig zu überladen, was eine Steigerung

der Übersichtlichkeit zur Folge hat, können diese entweder direkt in das Formular als Button eingebunden werden oder in die Menüleiste der Applikation aufgenommen werden. Dieser Aspekt ist jedoch wieder dem Benutzer überlassen, der das Formular bzw. das Plugin erstellt. Allerdings ist es wichtig, dem Benutzer beide Möglichkeiten zu bieten, um die Übersichtlichkeit der Anwendung nicht negativ zu beeinträchtigen. Da der Hintergrund für die Entwicklung von GUI4DB klinische Studien sind, kann der Benutzer bei der Formularerstellung ebenfalls Tooltips anlegen, um so möglicherweise schwer verständliche medizinische Termini näher zu erklären, was zum einen die allgemeine Bedienbarkeit bzw. Verständlichkeit des Formulars erhöht und zum anderen die Stabilität bzw. Datenqualität positiv beeinflusst.

Ein weiterer wichtiger Aspekt vor allem bei der Umstellung von einem Softwareprodukt auf ein anderes ist die „Wissenskontinuität“ (Richter (2007, 45)). Darunter versteht man, dass das Wissen eines Benutzers im Hinblick auf ein bestimmtes Softwareprodukt auch bei der Umstellung auf eine ähnliche Software erhalten bleiben sollte. Im konkreten Fall bedeutet dies, dass die Software prinzipiell ähnlich zu geläufigen Desktopdatenbanken wie MS Access aufgebaut ist, da die Benutzer vor dem Einsatz von GUI4DB ausschließlich mit dieser Software gearbeitet haben. Daher wurde zum einen die Desktopmetapher gewählt, das heisst, dass innerhalb der Anwendung mehrere Formulare in internen Fenstern angezeigt werden und zum anderen kann der Benutzer mit Hilfe der Benutzerschnittstelle sowohl Formulare als auch Datenbankabfragen erstellen.

## 3.7. Datenintegrität als Ausgangspunkt zur Durchführung medizinischer Auswertungen

Klinische Studien mit dem „Ziel [...], die Erfahrungen aus der Versorgung einzelner Patienten zu verallgemeinern, Regelhaftes in ihnen zu finden und zu beschreiben“ (siehe Leiner u. a. (2003, 6)) und deren Erkenntnisse beruhen allesamt auf den

Daten, die während der Phase der Patientenbefragungen bzw. Anamnese erhoben werden. So gilt es zu beachten, dass die Angaben, die die Patienten machen und die im Rahmen einer Studie gespeichert werden, sorgfältig geprüft werden und auch im Nachhinein, das heisst während der Auswertung der Studiendaten, noch nachvollziehbar sein müssen, um so die Korrektheit und Glaubwürdigkeit der Studie nicht negativ zu beeinträchtigen.

Desweiteren sollen die Patientendaten, sprich die medizinische Dokumentation der Patienten, dazu dienen, die Auswahl der Patienten zu ermöglichen, die in die Studie miteinbezogen werden. Daher ist es wichtig, ein einheitliches Informationssystem zu erstellen, das sowohl die Datenintegrität gewährleisten als auch die Auswertung der Daten ermöglichen soll. Ebenso muss sichergestellt werden, dass eingegebene Werte per se plausibel sind, um so von vornherein möglicherweise verzerrte Ergebnisse auszuschließen (vgl. Leiner u. a. (2003, 143)). Denn „the validity of the findings generated by a study clearly is an important dimension of quality“ (siehe Jüni u. a. (2001)). Dabei muss allerdings zwischen einer internen und externen Validität (vgl. Jüni u. a. (2001)) unterschieden werden, die beide in diesem Zusammenhang gleich wichtig sind.

Die interne Validität auf der einen Seite gibt Auskunft über den „extent to which systematic error (bias) is minimized in clinical trials“ (Jüni u. a. (2001)), sprich je weniger Verzerrungen bzw. systematische Fehler während der Erhebung der Patientendaten auftreten, desto höher ist die daraus resultierende interne Qualität der Studie. Diese möglichen Verzerrungen entstehen etwa durch falsches Vorgehen während der Patientenrekrutierung oder der inkorrekten Eingabe der Daten und wirken sich folglich negativ auf die komplette Studie aus.

Die externe Validität andererseits bezieht sich auf den Grad, zu dem es möglich ist, die aus einer Studie gewonnenen Ergebnisse weiterzuverwenden. Das bedeutet, dass die Resultate einer Studie nicht nur für eben jene Studie Gültigkeit haben dürfen, sondern auch als valide Basis für weiterführende Analysen verwendet werden können sollen. Insofern beruht die externe auf der internen Validität und somit ist auch die valide Durchführbarkeit von Clusteringanalysen, deren Daten auf der

Grundlage von Patientenerhebungen und den Auswertungen von Patientendaten erhoben werden, unbedingt von der Gültigkeit der Datenerhebung abhängig.

Mit Hilfe des eben beschriebenen Systems zur Dateneingabe ist es möglich Daten verlustfrei mit intuitiven Benutzerschnittstellen in der Datenbank abzuspeichern, was ein hohes Maß an Datenintegrität gewährleistet, welche wiederum Ausgangspunkt für die mögliche Auswertung der Daten ist, da die Qualität und Gültigkeit der Auswerteergebnisse direkt von der Qualität der zu Grunde liegenden Daten darstellt. Das Softwareframework hat daher direkten Einfluss auf die interne Validität der Studie, was sich wiederum in einer erhöhten externen Validität, also der allgemeinen Gültigkeit, der Studienergebnisse bemerkbar macht.

Insofern beruht auch die Qualität des später vorgestellten Clusteringverfahrens auf der Integrität und Güte der in Studien erhobenen und gespeicherten Daten, da die Blutproben, die zur Genextraktion herangezogen werden, an Hand der Patientendaten, die während der Dokumentationsphase erhoben werden, ausgewählt werden. Für den Fall von fehlerhaften oder unstimmigen Patientendaten hätte das wiederum direkte Auswirkungen auf die erhobenen Gendaten und folglich auch auf die Qualität der Ergebnisse der durchgeführten Clusteranalysen.

Konkret heisst das, dass mit Hilfe der hier vorgestellten Software und vor allem des entwickelten Datenbankmodells eine qualitativ hochwertige Auswertung von Studien- und Gendaten erreicht werden soll. Mit Hilfe eines konsistenten Datenmodells, das für alle Benutzer einheitlich ist und es ermöglicht ohne Inkonsistenzen Daten zu speichern, zu verwalten und zu bearbeiten, wird daher sichergestellt, dass die Daten, die auf Grund der Studie in eine DNA-Analyse einfließen eine hohe Qualität besitzen und daher auch zuverlässige Ergebnisse liefern.

Neben den hier vorgestellten Punkten wie Softwareentwicklung und -ergonomie sowie Workflow und Datenmanagement in klinischen Studien, ist ein weiterer Forschungsschwerpunkt dieses Projekts das Clustering genetischer Daten. Basierend auf den hier entwickelten Grundlagen, wurde daher das Clustering genetischer Daten im allgemeinen näher untersucht sowie eine Software zur Clusteranalyse von

Gendaten entwickelt. Auf diese Punkte wird in den folgenden Kapiteln detaillierter eingegangen.

# Kapitel 4.

# Clustering von Gendaten

Nach der Erläuterung wichtiger grundlegender Aspekte und Datenbanken im Bereich Bioinformatik bzw. Medizin und einer Einführung in den Workflow und das Datenmanagement in klinischen Studien sowie in User-Interface Management Systeme (UIMS) soll dieses Kapitel zuerst einen Überblick über wichtige Clusteringverfahren geben, um dann darauf aufbauend das in diesem Projekt entwickelte Clustering von genetischen Daten darzustellen. Einleitend kann zur Zielsetzung aller Clusteringverfahren festgehalten werden, dass mit ihrer Hilfe eine Menge $X$ an unstrukturierten und a priori unbekannten Objekten in möglichst homogene Gruppen zusammengefasst werden soll, um somit eine strukturierte Analyse der vorliegenden Daten gewährleisten zu können. Heutzutage ist der Einsatz von Clusteringverfahren zur Datenanalyse daher nicht mehr auf einzelne wissenschaftliche Domänen beschränkt, sondern findet in nahezu allen Bereichen der Wissenschaft und der Informationstechnologie Einsatz. Vor allem im Bereich der Genforschung mit ihren Massendaten bietet sich der Einsatz von Clusteranalysen bestens an, denn „the next great task of biology will be to convert its quickly accumulating supply of data into a broad understanding of the way organisms work“ (siehe etu). Obwohl Clustering in der Medizin und Biologie genauso weit verbreitet sind wie in anderen Wissenschaften, richtet sich das Clustering von genetischen Daten jedoch fast ausschließlich an die Analyse von Genexpressionsdaten. Das vorrangige Ziel solcher Clusteranalysen auf Basis von Genexpressionen ist es Zusammenhänge zwischen unbekannten genetischen Daten aus einer Menge an Genexpressionsdaten zu extrahieren, um so

Rückschlüsse auf die Funktion(en) von Genen und deren Zusammenhänge wie etwa in Signalpfaden zu zulassen. Der Ausgangspunkt für dieses Projekt ist allerdings ein anderer. Auf Grund der Fülle von bioinformatischen Algorithmen und Applikationen (siehe Kapitel 2.3) zur Berechnung der Wahrscheinlichkeit, dass ein Gen bzw. mehrere Gene für ein Krankheitsbild verantwortlich sind, entstehen mehrere unterschiedliche Ergebnisse, die jeweils von dem gewählten Algorithmus bzw. der gewählten Applikation oder dem für die statistische Berechnung zu Grunde liegendem genetischen Modell abhängen. Allerdings ist es bisher mühsam diese Ergebnisse zu validieren, um so Schlüsse ziehen zu können, welche Gene denn nun wirklich relevant für weitere Untersuchungen sind und welches Ergebnis am schlüssigsten und plausibelsten ist. Desweiteren werden diese Ergebnisse unter der Annahme genetischer Modelle, zum Beispiel die Annahme, dass ein Phänotyp dominant oder rezessiv verrerbt wird, errechnet, was bei der Genanalyse allerdings wiederum eine Grundkenntnis der zu Grunde liegenden Daten voraussetzt. Daher sollte das hier eingesetzte Clusteringverfahren vor allem zur Überprüfung der Qualität und Relevanz anderer Ergebnisse auf Basis von genetischen Daten dienen. Es war daher wichtig ein modellfreies Clusteringverfahren zu entwickeln, das Rückschlüsse auf die Relevanz von Gendaten zulässt und folglich auch auf die Qualität anderer statistischer Maße. Der modellfreie Ansatz war hierbei insofern wichtig, als dadurch die erhaltenen Ergebnisse einen generelleren Überblick und Rückschlüsse auf die zu Grunde liegenden Daten zulassen. Das gewählte Analyseverfahren hierbei war das Clustering von Genen ausgehend von SNP-Daten wie sie in Kapitel 2.1.1 näher beschrieben wurden. Dabei sollten allerdings nicht alleine die SNP-Daten, sondern vor allem die Gene selbst, auf denen sich die SNPs befinden, geclustert werden. Das Ziel der Clusterapplikation war dabei informative von uninformativen Daten zu unterscheiden, denn „from a set of hundreds of thousands of tests, many highly significant results are expected by chance alone, making it hard to distinguish signal from noise“ Purcell u. a. (2007). Vor diesem Hintergrund sollte ein Clusteringalgorithmus entwickelt werden, der nicht nur Gene in Cluster unterteilt, sondern auch

Rückschlüsse auf die Informativität und die Qualität der Daten sowie Ergebnisse anderer bioinformatischer Applikationen zulässt.

Dabei sollen Clusteringverfahren als „main tool used for explanatory data analysis, when one is dealing with data about whose internal structure little or no prior information is available“ Levine u. Domany (2001) dienen. Denn vor allem genetische Massendaten mit, wie im Falle der hier vorliegenden Daten, 820.000.000 Tupeln in einer Datenbank, stellen eine große Herausforderung dar, wenn es darum geht, Strukturen und Muster in unbekannten Daten zu erkennen und diese zu analysieren.

Im Folgenden wird zunächst eine Übersicht über häufig eingesetzte und am weit verbreitete Clusteringverfahren gegeben, um danach das hier eingesetzte Verfahren zu erklären und einzuordnen.

## 4.1. Clusteringverfahren

Generell können Clusteringverfahren in folgende Kategorien unterteilt werden (siehe etwa Steinbach (2000)):

- Partitionierendes Clustering
- Hierarchisches Clustering
- Clusterbildung mit Hilfe neuronaler Netze bzw. maschinellen Lernens

Obwohl es viele unterschiedliche Clusteringalgorithmen gibt, die auf spezielle Anforderungen zugeschnitten sind, so wird im folgenden jedoch nur auf die im Bereich Genetik/Bioinformatik am häufigsten eingesetzten Algorithmen eingegangen. Es muss allerdings erwähnt werden, dass vor allem im Bereich der Genanalyse keine vielfältigen Clusteringverfahren eingesetzt werden. In der Literatur treten in diesem Zusammenhang häufig nur die am weitesten verbreiteten Algorithmen wie k-means oder hierarchisches Clustering auf (siehe z.B. Bar-Yossef u. a. (2002), Jiang u. Zhang (2002), Ben-Dor u. a. (1999)). Vor allem im Bereich des Clusterings

von Gendaten auf der Basis von informationstheoretischen Parametern gibt es keine größeren Studien oder Ansätze. Neben partitionierendem und hierarchischem Clustering, die ohne weiteres als Standardverfahren des Clusterings bezeichnet werden können (vgl. Kaufman u. Rousseeuw (1990)), spielt auch das dichtebasierte Clusteringverfahren (DBSCAN) eine nicht zu unterschätzende Rolle im Bereich der Bioinformatik (vgl. Erciyeş (2010) oder Kriegel u. Kröger (2004)), weswegen auch auf dieses Clusteringverfahren eingegangen wird. Eines der sicherlich am meisten verwendeten Verfahren nicht nur im Bereich Genetik oder Bioinformatik ist hierbei das sogenannte *k-means* Clustering (siehe etwa Potamias (2006)), auf das im folgenden Abschnitt näher eingegangen wird.

### 4.1.1. Partitionierendes Clustering

Obwohl es unterschiedliche partitionierende Clusteringverfahren gibt, die auf bestimmte Daten abgestimmt sind, soll hier im Folgenden auf das k-means Clustering als Beispiel für ein partitionierendes Verfahren, eingegangen werden, um so den Ablauf bzw. den Hintergedanken von partitionierenden Clusteringalgorithmen zu verdeutlichen. Der k-means Algorithmus mit einer Rechenkomplexität von $O(nkt)$, mit $n$ Objekten, $k$ Clustern und $t$ Iterationen zielt genauso wie andere Clusteringverfahren darauf ab eine vorgegebene Datenmenge so umzustrukturieren, dass möglichst homogene Cluster entstehen, die eine optimale Einteilung der Inpudaten darstellen. Unter „optimaler Klassifikation“ versteht man dabei, dass die klassifzierten Objekte innerhalb der Cluster eine sehr hohe Ähnlichkeit aufweisen und zwischen den Clustern sehr unterschiedlich sind.

Das von Lloyd im Jahr 1957 (Lloyd (1982) veröffentlich erst 1982) entwickelte unbeaufsichtigte Verfahren zur Clusterbildung sieht vor, dass eine Anfangspartitionierung hinsichtlich einer vorgegebenen Anzahl von $k$ Clustern optimal im Bezug auf Homogenität und Heterogenität umstruktiert wird. Die vorgegebene Anfangspartitionierung sowie die Anzahl der Cluster kann dabei zufällig gewählt werden. Neben diesen Aspekten hängt das Ergebnis, wie in anderen Clusteringverfah-

ren auch, ebenfalls von der gewählt Abstandsmetrik zur Berechnung der Distanz zwischen Objekten und den Zentroiden der Cluster ab. Der Algorithmus des k-Means Verfahren ist im folgenden schematisch dargestellt:

1. Auswahl von Startwerten für die Zentroide
2. Verschieben des ersten Objektes in einen Cluster dessen Distanz am geringsten zu einem Clusterzentroiden ist
3. Neuberechnung der Zentroiden
4. Durchführung des Schritts Nummer 2 bis die maximale Anzahl der Iterationen erreicht wurde oder bis in $z$ Iterationen kein Objekt mehr verschoben worden ist.

Da nach jeder einzelnen Verschiebung von Objekten in andere Cluster die Zentroide der Cluster neu berechnet werden, ist es möglich, dass der Algorithmus sehr schnell konvergiert. Problematisch dabei ist allerdings, dass die Partitionierung von der Reihenfolge der untersuchten Objekte abhängt. So werden Klassen umpartitioniert und die Zentroiden dementsprechend neu berechnet je nachdem welches Objekt verschoben wurde, was wiederum von Reihenfolge der Inputobjekte abhängig ist. Die Abhängigkeit von der Reihenfolge der berechneten Objekte resultiert daher in einer geringeren Stabilität des Clusterings im Vergleich zu etwa semi-automatischen Clusteringverfahren, die ein Training für das eigentliche Clustering voraussetzen. Allerdings bietet das k-Means Verfahren den großen Vorteil, dass eine Implementierung sehr schnell und leicht durchgeführt werden kann, was die oben genannten Nachteile unter Umständen weniger negativ erscheinen lässt.

Ein weiteres Problem, das nicht nur auf das k-means Verfahren, sondern auf alle partitionierenden Clusteringverfahren, zutrifft ist, dass die Anzahl $k$ der Partitionen a priori festgelegt werden muss (sieh dazu etwa Jiang u. Zhang (2002)). Dies setzt zum einen eine genaue Kenntnis der zu partitionierenden Daten voraus, was sicherlich in den meisten Fällen unrealistisch ist, zum anderen kann dies dazu führen,

dass die Klassen zu ungenau partitioniert werden, da sie entweder in zu viele oder zu wenige Cluster aufgeteilt werden. Obwohl die Anzahl *k* mit Hilfe von Clustervalidierungen angepasst und optimiert werden kann, ist dieser Punkt einer der problematischsten im Bezug auf das k-means Verfahren. Ausserdem ist es notwendig den Wertebereich der Inputobjekte zu kennen, um die *k* Partitionen anfangs mit Zufallszahlen initiieren zu können. Werden nämlich die *k* Anfangspartitionen mit Werten initiiert, die nicht oder nur zum Teil im zu erwartenden Wertebereich liegen, kann es sein, dass die Objekte zu ungenau, sprich mit geringer Intraclusterhomogenität und hoher Interclusterheterogenität aufgeteilt werden. Jedoch reduziert die optimale Initialisierung der Zentroiden den Rechenaufwand, da weniger Anpassungsdurchläufe benötigt werden, um ein optimales Clusteringergebnis zu erhalten. Diese Punkte müssen allerdings von Fall zu Fall entschieden werden und sind stark von den Daten und der geplanten Anwendung abhängig. Ferner kommt noch erschwerend hinzu, dass es theoretisch vorkommen kann, dass in einer Iteration ein Cluster *x* geleert wird und dieser dann nicht mehr neu gefüllt werden kann, da logischerweise kein Zentroid für *x* mehr berechnet werden kann.

Wie in Bacher (1994, 311) gezeigt wird, wird das k-means Verfahren vor allem für große Datenmengen (mit n>= 1000) stabiler im Vergleich zu kleinen Datenmengen, was bei dem hier umgesetzten Verfahren problematisch ist, da kaum mehr als 300 Gene pro Analyse geclustert werden sollen. Auch wenn in diesem Fall in Bacher (1994) von überlappenden Datenwerten, sprich von Werten aus einem einzigen Werteintervall, die darüber hinaus auch noch in mehr als einem Cluster auftreten können, ausgegangen wird, so kann man dennoch festhalten, dass die Stabilität des Ergebnisses von der zu Grunde liegenden Datenmenge abhängt. Werden nur Daten geclustert, deren Werte sich nicht überlappen, zum Beispiel Werte aus dem Intervall [0, 1] und [-1, 0], „so werden bereits ab einem Stichprobenumfang von 50 Objekten stabile Ergebnisse erzielt“ (Bacher (1994, 312)). Dies ist allerdings eine eher illusorische Annahme, da sich wohl keine Datenmengen zu 100% voneinander abgrenzen lassen, was die Intervalle ihrer Werte angeht.

Neben dem k-means Verfahren ist auch der k-medoids Algorithmus Kaufman u. Rousseeuw (1990) ein verbreiteter Algorithmus im Bereich des partitionierenden Clusterings. Im Gegensatz zu k-means werden allerdings nicht die Mittelwerte aller Objekte eines Clusters zur Distanzberechnung herangezogen, sondern der Wert eines „Medoids“ . Ein Medoid ist dabei als das Objekt definiert, dessen Distanz zu allen Objekten eines Clusters am geringsten ist. Da bei der Interclusterdistanzberechnung nicht, wie im Falle von k-means, das durchschnittliche Gewicht eines kompletten Clusters herangezogen wird, sondern die Distanz zwischen Medoiden und anderen Clusterobjekten auf einer paarweisen Distanzberechnung beruht, ist dieses Verfahren weniger anfällig für Ausreisser innerhalb eines Clusters.

Prinzipiell ist der Ablauf des k-medoids Ansatzes der gleiche wie beim k-means Algorithmus nur dass wie bereits erwähnt die Ähnlichkeit bzw. Distanz der Cluster zueinander auf Basis von, bei der ersten Iteration, zufällig initialisierten Medoiden berechnet wird. Nun werden die Objekte so verteilt, dass die Distanz zu den jeweils ausgewählten Medoiden optimal ist. Findet sich nun ein Objekt eines Clusters $C$, das besser als Medoid geeignet ist basierend auf folgender Fehlersumme mit $m(i)$ als Medoid und $i$ als Objekt innerhalb eines Clusters:

$$E = \sum_{i=1}^{n} d(i, m(i)) \tag{4.1}$$

Sollte basierend auf dieser Gleichung kein Medoid gefunden werden, der die Fehlersumme $E$ minimiert, so ist der optimale Zustand des Clusterings gefunden. Andernfalls werden ausgehend auf den neu definierten Medoiden die Objekte wiederum verschoben. Auf Grund der Distanzberechnung basierend auf Medoiden ist dieser Algorithmus zwar robuster im Hinblick auf Ausreisser allerdings ist er auch rechenintensiver als der k-means Algorithmus, da bei jeder Iteration ein weiterer Medoid gefunden werden kann, für den wiederum die Distanzen zu anderen Clusterobjekten neu berechnet werden muss. Genau wie der k-means Algorithmus hat

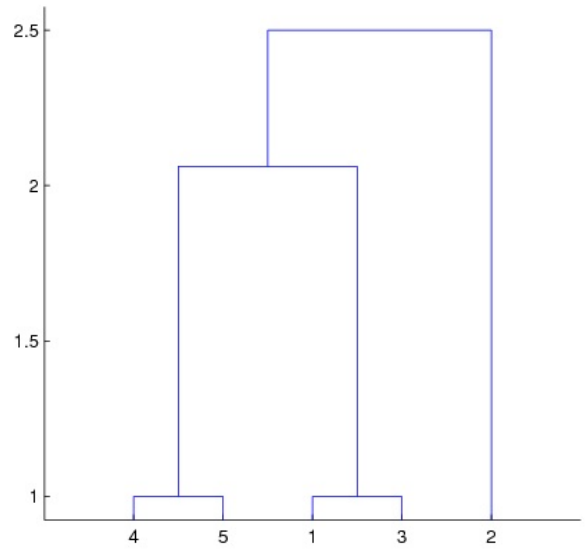



Abbildung 4.1.: Schematische Visualisierung des hierarchisches Clusterings in Form eines Dendrogramms (Quelle: Mathworks)

der k-medoids Algorithmus auch das Problem, dass $k$, sprich die Anzahl der Cluster, a priori definiert werden muss.

### 4.1.2. Hierarchisches Clustering

Im Gegensatz zu partitionierenden Clusteringverfahren, wie sie im vorhergegangenen Kapitel beschrieben wurden, bilden hierarchische Verfahren Cluster in Form einer Baumstruktur bzw. eines Dendogramms und eignen sich auf Grund der hohen Rechenkomplexität mit $O(n^2)$ vor allem für kleinere Datenmengen. Zur Veranschaulichung soll das Dendrogramm in 4.1 dienen, das eine typische Baumstruktur eines hierarchischen Clusterings darstellt.

Hierarchische Clusteringverfahren lassen sich grundsätzlich wiederum in zwei Gruppen aufteilen. Zum einen gibt es agglomerative Verfahren, in denen die Anfangscluster aus jeweils nur einem Objekt bestehen und danach Schritt für Schritt zu größeren Clustern kombiniert werden. Zum anderen gibt es divisive Verfahren, in

deren Fall es nur eine Anfangsklasse gibt, die im Verlauf des Clusterings aufgeteilt wird.

#### Divisive Verfahren

Das divisive Clustering von Objekten stellt einen typischen *top-down*-Ansatz dar, da sich zuerst alle Objekte in einer einzelnen Top-Level Klasse befinden. Im ersten Schritt dieses Verfahrens, wird zuerst das Objekt mit dem größten Abstand zum Zentroiden gesucht, das wiederum einen neuen Cluster bildet. Für die Ermittlung des am weiten entferntesten Objekts kann das sogenannte *complete linkage* Verfahren, das allerdings ebenfalls für das agglomerative Clustering eingesetzt werden kann, angewandt werden:

$$d_{max}(k_{r1}, k_{r2}) = min(max_{x_{i1} \in k_{r1}, x_{i2} \in k_{r2}} d(x_{i1}, x_{i2})) \tag{4.2}$$

Danach werden alle Objekte deren Distanz zum Zentroiden des neuen Clusters geringer ist als die Distanz zum Ausgangszentroiden in den neuen Cluster verschoben. So bildet sich die für hierarchische Clusteringverfahren typische Struktur eines Binärbaumes heraus.

#### Agglomerative Verfahren

Im Gegensatz zu divisiven Verfahren stellen agglomerative Clusteringverfahren einen klassischen *bottom-up*-Ansatz dar. Dabei werden zuerst alle zu clusternden Objekte in eigene Klassen aufgeteilt. Das heisst, dass anfangs jedes Objekt in einen eigenen Cluster geschoben wird. Danach werden die zwei ähnlichsten Klassen, d.h. die Klassen mit dem geringsten Abstand zueinander, zusammengelegt. Die Berechnung der ähnlichsten Objekte im agglomerativen Verfahren kann dabei mit Hilfe zum Beispiel des *complete linkage* Verfahrens durchgeführt werden. Neben dem complete linkage gibt es noch zahreiche andere Distanzberechnungen, die vor allem in hierarchischen Clusteringansätzen zum Einsatz kommen wie:

- single linkage
- average linkage
- centroid linkage

Für eine detailliertere Beschreibung der oben erwähnten Distanzmaße siehe Kapitel 4.1.5.

### 4.1.3. Dichtebasiertes Clustering

Neben partitionierden und hierarchischen Clusteringverfahren spielen auch dichtebasierte Verfahren wie DBSCAN (siehe Ester u. a. (1996)) eine wichtige Rolle bei der Clusteranalyse unbekannter Objekte. Dabei wird im Falle des DBSCAN Algorithmus davon ausgegangen, dass die Dichte innerhalb eines Clusters größer ist als außerhalb des Clusters. Die Dichte kann dabei als die Anzahl von Punkten um ein Objekt in einem bestimmten Radius $\varepsilon$ definiert werden. Das Problem hierbei ist allerdings die Definition des Radius $\varepsilon$. Ist dieser nämlich zu groß so haben alle Punkte im schlimmsten Fall dieselbe Dichte. Ist $\varepsilon$ zu gering definiert so können als anderes Extrem alle Punkte eine Dichte von 1 aufweisen.

Der DBSCAN Algorithmus weist generell drei unterschiedliche Arten von Punkten auf. Zum einen gibt es Kernpunkte, Grenzpunkte und Rauschpunkte (vgl. Tan u. a. (2005), 527 ff). Die Punkte sind dabei folgendermaßen definiert (vgl. Tan u. a. (2005), 527 ff bzw. Ester u. a. (1996)):

- Kernpunkte: Diese Punkte bilden das Zentrum eines Clusters und für welche gilt, dass die Anzahl anderer Punkte in einem bestimmten Radius $\varepsilon$ einen vom Benutzer festgelegten Grenzwert $MinPts$ überschreitet .
- Grenzpunkte: Diese Punkte sind selbst keine Kernpunkte, befinden sich jedoch in der Nachbarschaft eines oder mehrerer Kernpunkte.
- Rauschpunkte: Diese Punkte sind weder Kern- noch Grenzpunkte.

Um nun Punkte zu clustern, müssen zuerst alle Punkte als entweder Kern-, Grenz- oder Rauschpunkte kategorisiert werden. Im folgenden Schritt werden alle Rauschpunkte eliminiert und folglich nur noch die Kern- und Grenzpunkte geclustert. Befinden sich Kernpunkte in einem vorab definierten Radius $\varepsilon$ zueinander, so gilt, dass diese in einem Cluster vereinigt werden. Ebenfalls werden die Grenzpunkte mit dem am nächsten liegenden Kernpunkt in einem Cluster verlegt.

Das Problem des DBSCAN Algorithmus ist allerdings die Definition der beiden Parameter $\varepsilon$ und $MinPts$. Hierzu schlagen Ester u. a. (1996) eine Heuristik vor, in der die Distanz eines Punktes zu $k$ nähesten Punkt ermittelt wird auch $k-dist$ genannte (siehe Ester u. a. (1996)). Werden nun so die Distanzwerte zwischen Punkten und den Nachbarn ermittelt, können die Werte sortiert werden und in einem Diagramm dargestellt werden. Auf Grund der Distanz zwischen Punkten und Rauschpunkten tritt in dem entstehenden Diagramm ein Bruch auf, der als Wert für $\varepsilon$ herangezogen werden kann. Der Wert für $k$ kann dabei als Parameter $MinPts$ definiert werden.

Im Gegensatz zu partitionierenden Verfahren wie k-means ist DBSCAN kaum anfällig für Ausreisser in der Menge der zu clusternden Objekte. Allerdings ist eine große Varianz der Dichtewerte ein Problem des DBSCAN (siehe Tan u. a. (2005), 532). Ferner ist der Algorithmus unter Umständen sehr rechenintensiv falls, da die Distanz paarweise für alle benachbarten Punkte berechnet werden muss, was vor allem für mehrdimensionale Punkte von großem Nachteil ist.

### 4.1.4. Self-organizing Maps

Da im Zuge dieses Projektes der Self-organizing Maps (SOM) Algorithmus implementiert wurde, um genetische Daten zu clustern, wird im Folgenden näher auf diesen Algorithmus eingegangen. Die Wahl des SOM-Algorithmus beruht darauf vor allem auf den Vorteilen im Vergleich zum k-means Algorithmus (siehe unten). Auch wenn im Bereich der Genetik auch andere Algorithmen entwickelt wurden (siehe Beschreibung des CLICK-Algorithmus in Kapitel 4.2) so dient die hier vor-

liegende Auswahl und Implementierung des SOM-Algorithmus als Vergleichsimplementierung zu anderen Clustering-Projekten in der Genetik, um die Performanz und Qualität des SOM-Algorithmus für mehrdimensionale genetische Daten zu untersuchen.

Der SOM-Algorithmus wurde in den 1980-er Jahren von Teuvo Kohonen (Kohonen (1997)) entwickelt. Dieser Algorithmus bildet ein künstliches neuronales Netz ab, um durch „Reizübertragung“ Objekte den passendsten Clustern zuzuweisen. Ursprünglich wurde der Algorithmus entwickelt um mehr-dimensionale Gewichtsvektoren auf einer zwei-dimensionalen Ebene abzubilden und zu visualisieren bzw. zu clustern. Der Grundgedanke bei der Entwicklung des Algorithmus war die Abbildung von neuronalen Verbindungen und Reizübertragungen im menschlichen Gehirn auf informationstechnische Clusteringprobleme. Dieser Gedanke ist insofern sinnvoll, da im menschlichen Gehirn das gespeicherte Wissen in Form von Neuronen, die sich in einer bestimmten Form anordnen, d.h. miteinander verbunden sind, enthalten ist. Trifft nun ein äusserer Stimulus z.B. in Form eines visuellen Reizes auf das Gehirn, so wird dort mittels Reizübertragung in einer bestimmten Region mit Hilfe von Merkmalskarten entschieden, um was es sich bei diesem Objekt handelt. Bei dieser Entscheidung in der Phase der sogenannten „präattentiven Entscheidung“ werden die einzelnen Merkmale des zu klassifizierenden Objekts mit im Gehirn gespeicherten Merkmalen verglichen und dem passenden Merkmalsort zugeordnet. Der passende Merkmalsort wird indentifiziert indem das Inputmerkmal der Merkmalskarte präsentiert wird und dort den „richtigen“ Dektektor aktiviert. Genauso verhält es sich im Falle des SOM-Algorithmus. Hierbei wird die Struktur des menschlichen Gehirns in Form einer sogenannten Kohonenkarte mit einer Anzahl $x$ an Neuronen/Knoten, die miteinander verbunden sind, abgebildet.

Zuerst wird in einer unüberwachten Trainingsphase eine Merkmalskarte aufgebaut, der dann im eigentlichen Clusteringprozess Inputvektoren zugewiesen werden. Diese Merkmalskarte besteht aus $x$ Neuronen, die während der ersten Phase des Algorithmus trainiert werden, um die Struktur des menschlichen Gehirns, genauer gesagt, die Struktur der Reizklassifizierung, bestmöglich abzubilden. Die Inputvek-

toren werden daraufhin nach der Trainingsphase den Knoten auf der Merkmalskarte zugewiesen zu denen sie am besten passen, sprich zu denen sie die geringste Distanz aufweisen (siehe Kapitel zu Ähnlichkeitsmetriken: 4.1.5).

Auch wenn Jiang u. Zhang (2002) festhalten, dass der SOM-Algorithmus „does not overcome the problems of K-means such as cluster number determination“, so bietet der SOM-Algorithmus in diesem Zusammenhang dennoch den Vorteil, dass, obwohl sicherlich eine initiale Anzahl an Knoten der Kohonenkarte definiert werden muss, die Karte dennoch so trainiert wird, dass diese Karte auf eine optimale Anzahl an relevanten Knoten schrumpft. Denn anders als beim K-means Algorithmus besteht daher nicht die Möglichkeit, dass zu klassifizierende Objekte im Laufe des Algorithmus in alle möglichen Klassen verschoben werden, sondern nur in die die während der Trainingsphase optimal auf die zu erwartenden Inputdaten trainiert wurden. Das heisst, dass nur die Knoten um das Zentrum der Kohonenkarte für die Zuweisung von Inputdaten relevant sind. Für alle anderen Knoten besteht nur eine äusserst geringe Wahrscheinlichkeit, dass ihnen Daten zugewiesen werden, weswegen sie prinzipiell vernachlässigt werden können. Insofern spielt die initiale Anzahl der Knoten innerhalb der Kohonenkarte eine wesentlich geringere Rolle als zum Beispiel beim K-means Algorithmus.

Ebenso ist die Behauptung von Jiang u. Zhang (2002), dass der SOM-Algorithmus, genauso wie K-means oder hierarchische Clusteringalgorithmen, nur ein lokal optimales Ergebnis liefern, da die Objekte „are grouped based on local decisions, with no guarantee of global optimization“ nur bedingt gültig. Dadurch, dass ein SOM-Netzwerk anfangs zufällig initialisiert wird und diese Anfangskarte Schritt für Schritt von einer globalen Karten auf eine eher „lokale“ Karte reduziert wird, unterschtreicht auf den ersten Blick Jiangs Behauptung. Allerdings darf nicht vergessen werden, dass die ursprünglich globale Aufteilung der SOM-Karte insofern reduziert wird, da nur die Knoten der Karte angepasst und trainiert werden, die im Hinblick auf die zu erwartenden Inputdaten relevant erscheinen. Insofern beruht die Zuweisung zu einem Cluster nicht auf einer *local decision*, sondern eher auf einer optimierten globalen Entscheidung.

Ein weiterer Vorteil des SOM-Algorithmus gegenüber anderen Clusteringalgorithmen wie etwa k-means ist, dass nach der Trainingsphase lediglich ein Durchlauf ausreicht um Daten zu clustern. Sollten sich daher die Wertebereiche der Daten von Anaylse zu Analyse nicht stark ändern, so ist es möglich die gleiche trainierte Kohonenkarte für Clusteranalyse heranzuziehen, was einen erheblichen Performanzgewinn darstellt. Neben dem Performanzgewinn bei der Wiederverwendung von bereits trainierten Kohonenkarten stellt auch die dadurch erzielte Stabilität des Algorithmus einen enormen Vorteil gegenüber Algorithmen wie k-means oder hierarchisches Clustering dar. Denn anders als bei beispielsweise k-means, bei der jeder Clusteringdurchlauf andere Ergebnisse mit derselben Datengrundlage liefern kann, werden beim SOM-Algorithmus die Daten immer gleich auf Basis von bereits trainierten Netzwerken in Cluster eingeteilt.

In den nachfolgenden Abschnitten werden die einzelnen Phasen des SOM-Algorithmus, sprich die Trainings- und die Clusteringphase im Detail beschrieben.

#### Trainingsphase

Während der unbeaufsichtigten Trainingsphase (siehe Abbildung 4.2) wird ein Netzwerk mit x Knoten initialisiert und trainiert, um später während des eigentlichen Clusteringalgorithmus die Inputobjekte den passenden Knoten zuzuweisen. Dazu werden zuerst alle Knoten mit Zufallswerten initialisiert. Um nun die einzelnen Knoten zu trainieren werden dem Netzwerk zufällig Inputvektoren zugeführt. Der Knoten, dessen Distanz zum Inputvektor $i$ am geringsten ist - die sogenannte *best-matching unit* (BMU) - und dessen benachbarte Knoten, die in einem Radius $r$ um die BMU liegen, werden dabei von der BMU „lernen“. Das heisst, dass der Gewichtsvektor der Knoten in einem bestimmten Radius $r$ um die BMU so angepasst werden, dass sie dem Inputvektor ähnlicher werden. Mit einer steigenden Anzahl an Iterationen führt das immer weiter zu einer Optimalisierung bzw. eines „glättenden

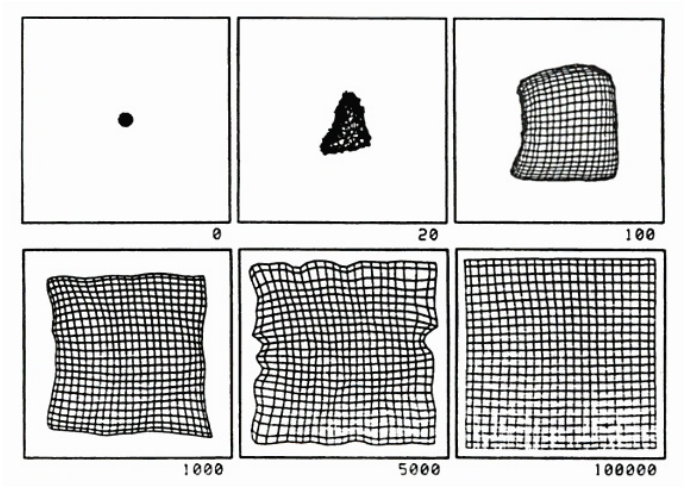



Abbildung 4.2.: Eine Kohonenkarte während des Lernprozesses. Die Zahlen geben die Anzahl der Iterationen an. (Kohonen (1997, 114))

Effekts“ („smooting effect“ Kohonen (1997, 87)) des neuronalen Netzes. Um die Gewichtsvektoren der einzelnen Knoten anzupassen dient Gleichung 4.3.

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{4.3}$$

In obiger zeitabhängiger Gleichung verweist die Variable $t$ auf die Iteration, $m$ auf den Knoten/Gewichtsvektor in der Kohonenkarte und $i$ auf den Inputvektor. Als besonders wichtig für die Lernphase ist dabei die Variable $h_{ci}$ anzusehen. Diese sogenannte Nachbarschaftsfunktion $h_{ci}$ dient dem Annähern der einzelnen Gewichtsvektoren zueinander. Damit sich die Werte anpassen und ihrem endgültigen idealen Wert annähern gilt $h_{ci} \to 0$ für $t \to \infty$. Nun lässt sich diese Vorgabe einfach erreichen, indem ein ein festgelegter Ausgangswert immer mit einem Faktor f ($0 < f < 1$) multipliziert wird. Eine andere von Kohonen selbst vorgeschlagene einfache Lösung wäre es $h_{ci}(t) = \alpha(t)$ zu setzen für den Fall, dass $i \in N_c$ mit $N_c$ als einer Menge um den Knoten $c$. Für den Fall dass $i \notin N_c$ gilt $h_{ci}(t) = 0$. Der Wert von $\alpha(t)$ ist dabei ein Faktor, der streng monoton fällt mit steigenden $t$, und für den folgendes gilt:

$0 < \alpha(t) < 1$. Eine andere Nachbarschaftsfunktion, die von Kohonen favorisiert und als *smoother* bezeichnet wird, beinhaltet die Gaußsche Exponentialfunktion und ist in 4.4 gegeben.

$$h_{ci} = \alpha(t) * exp(-\frac{\| r_c - r_i \|^2}{2\sigma^2(t)}) \tag{4.4}$$

Der Ausdruck $r_c - r_i$ verweist auf die Distanz zwischen der BMU und dem Knoten $c$. Wie bereits erwähnt ist $\alpha(t)$ ein streng monoton fallender Faktor. Dasselbe gilt für die Variable $\sigma(t)$, die den Radius um die BMU darstellt und der oben erwähnten Variablen $N_c$ entspricht. Daher gibt $\sigma(t)$ an welche Knoten um die BMU angepasst werden sollen. Dieser Radius sinkt ebenfalls für steigende $t$, um somit die Anzahl der anzupassenden Knoten im Verlauf der Trainingsphase zu verringern. Der Radius ist allerdings ein sehr wichtiger, wenn nicht sogar der wichtigste, Parameter, da für einen sehr kleinen Anfangsradius gar keine oder zu wenige Knoten verbessert werden und für einen zu großen Anfangsradius zu viele Knoten angepasst werden. Daher ist es wichtig einen Mittelweg zu finden, um den Radius zu initialisieren und um so die topografische Ordnung der Kohonenkarte zu optimieren. Kohonen schlägt einen Radius größer der Hälfte des Durchmessers der Kohonenkarte als Initialwert vor:

> the initial radius of $N_c$ can even be more than half the diameter of the network!

(Kohonen (1997, S. 88)). Desweiteren ist es wichtig eine passende Funktion zu benutzen, damit der Radius nicht zu stark oder zu schwach mit steigendem $t$ abfällt.

Nicht nur das Anpassen des Radius um die *best-matching unit* BMU ist von großer Wichtigkeit, sondern auch das Anpassen des Faktors $\alpha(t)$. Im Falle von $\alpha$ spielt vor allem $t$ eine große Rolle. Laut Kohonen sollte vor allem für $t$ kleiner 1000 gelten, dass $\alpha \approx 1$ und erst ab ca. $t >= 1000$ sollte $\alpha(t)$ streng monoton fallen. Diese Entscheidung bewirkt, dass während der ersten 1000 Iterationen die Knoten innerhalb der Karte särker geordnet werden, wohingegen ab $t$ größer 1000 die Kno-

ten nur noch feinjustiert werden. Ebenso wie für $\sigma(t)$ ist auch für $\alpha(t)$ wichtig, eine Funktion zu finden, die den Wert nicht zu stark und nicht zu schwach sinken lässt. Dazu gibt es in der Literatur etliche Vorschläge. Für Details der eingesetzten Funktionen siehe Kapitel 4.9.1.

Ferner spielt auch die Anzahl der Iterationen $t$, um die Karte während der Trainingsphase optimal zu ordnen, eine große Rolle. Diese Anzahl hängt dabei wiederum vom Einsatzgebiet des Algorithmus ab. Für den Fall, dass der Algorithmus in zeitkritischen Programmen, wie zum Beispiel der Spracherkennung, eingesetzt wird, sollte die Anzahl der Iterationen - abhängig von der Hardwarkonfiguration - nicht zu groß gewählt werden. Im Falle von Spracherkennung schlägt Kohonen als $t$ „10000 steps and even less“ (Kohonen (1997, S. 88)) vor. Ist die Rechenzeit allerdings kein kritischer Faktor so sollte $t = 500 * x$ als Minimum gesetzt werden, wobei $x$ die Anzahl der Netzwerkknoten angibt. Nur bei dem oben angegebenem Wert oder mehr Iterationen, kann man davon ausgehen, dass die Anordnung des Netzwerkes sich an ein Optimum annähert.

#### Klassenbildung mit Hilfe des SOM-Algorithmus

Nach der Trainingsphase werden die zu clusternden Objekte an Hand des trainierten Netzwerks von Knoten klassifiziert. Dazu werden dem trainierten Netzwerk die zu klassifizierenden Objekte als Input übergeben. Für jedes Inputobjekt wird daraufhin für jeden Knoten innerhalb des Netzwerks die Distanz zum Inputvektor berechnet und nach den Berechnungen das Objekt in den Knoten verschoben für den gilt:

$$min(d(x_i, x_j)) \tag{4.5}$$

wobei $x_i$ den Gewichtsvektor des Inputs und $x_j$ den Gewichtsvektors des Knotens im Netzwerk referenziert. Anders als wie zum Beispiel beim k-means-Verfahren, durchläuft der Clusteringprozess nur $i * j$ Berechnungen was zu einer enormen Performanzsteigerung führt. Für die Qualität des Clusteringergebnisses ist dabei die

Qualität der Trainingsphase ausschlaggebend. Wurden zu wenige Trainingsschritte durchgeführt, führt dies möglicherweise dazu, dass die Objekte nur unzureichend geclustert werden bzw. zu viele oder zu wenige Cluster entstehen und die Objekte nicht optimal aufgeteilt werden. Sollte es allerdings vorkommen, dass die Trainingsdaten zu spärlich oder zu ungenau sind, d.h. die Trainingsdaten spiegeln die tasächlichen Daten nur unzureichend wider, kann es zu einem sogenannten *over-fitting* kommen (siehe Weijters u. a. (1997)). Unter *over-fitting* versteht man dabei den Zustand, dass „the generalisation performance of the trained model is much worse than its performance on the training material (i.e., its ability to reproduce the training material)“ (Weijters u. a. (1997)). Da bei den vorliegenden genetischen Daten keine nennenswerten Unterschiede bzgl. der Datenqualität zwischen zufällig ausgewählten Trainingsdaten und den tatsächlichen Clusteringdaten zu erwarten waren, wird an dieser Stelle nicht näher auf das Problem des *over-fitting* eingegangen.

### 4.1.5. Ähnlichkeitsmaße

Genauso wichtig wie die Auswahl eines geeigneten Clusterinalgorithmus ist sicherlich auch die Wahl eines passenden Distanzmaßes zur Bestimmung des Abstandes bzw. der Ähnlichkeit zwischen zwei Objekten zueinander, auf Basis derer die Objekte in Cluster eingeteilt werden. Prinzipiell werden dabei Objekte als Punkte in einem x-dimensionalen Raum projiziert, zwischen denen mittels Distanzmaßen die Ähnlichkeit bestimmt werden soll. Grundlegend können folgende Eigenschaften von Ähnlichkeitsmaßen angenommen werden (siehe Aldenderfer u. Blashfield (1985, 18)):

- Symmetrie, das heisst, dass $d(x,y) = d(y,x)$
- Unterscheidbarkeit von nicht identischen Objekten. Das heisst, dass falls gilt $d(x,y) \neq 0$ dann $x \neq y$

- Nicht-Unterscheidbarkeit von identischen Objekten. Das heisst, dass falls gilt $d(x,y) = 0$ dann $x = y$

Obwohl mehrere Arten von Ähnlichkeitsmaßen existieren, wie etwa Korrelationskoeffizienten, Assoziationskoeffizienten und probabilistische Koeffizienten, wurden in diesem Projekt die intuitiveren Distanzmaße verwendet, die im folgenden Abschnitt näher erläutert werden. Neben der Intuivität von Distanzmaßen, spielen noch weitere Gründe für die Wahl dieser Ähnlichkeitsmaße eine Rolle. Zum einen hängen Korrelationskoeffizienten von der Form der geometrischen Repräsentation der Objekte ab, was dazu führen kann, dass zwei Objekte geometrisch identisch jedoch die Korrelation zwischen ihnen nicht identisch ist (siehe dazu Aldenderfer u. Blashfield (1985, 23f)). Dazu kommt, dass Assoziationskoeffizienten bei binären Variablen zum Einsatz kommen. Jedoch werden in diesem Projekt keine binären Variablen verwendet. Das selbe trifft zu für probabilistische Ähnlichkeitskoeffizienten (siehe Aldenderfer u. Blashfield (1985, 33)).

Was die meisten Distanzmetriken, im konkreten Fall die Euklidische Distanz, die City-block bzw. Manhattan-Metrik und die Chebychev-Metrik, gemeinsam haben, ist, dass sie alle mathematisch auf der verallgemeinerten Minkowski-Metrik beruhen (siehe Gleichung 4.6). Der Unterschied zwischen den Distanzmetriken entsteht basierend auf Gleichung 4.6 durch unterschiedliche Werte für die Variablen $r$ und $q$. Allgemein kann man jedoch sagen, dass „ein größerer Metrikparameter r bewirkt, daß größere Unterschiede in weniger Variablen stärker gewichtet werden als kleine Unterschiede in vielen Variablen“ (Bacher (1994, 222)).

$$d(x,y) = [\sum_i \mid x_i - y_i \mid^r]^{\frac{1}{q}} \tag{4.6}$$

### Euklidische Distanz

Als ein wichtiges Distanzmaß für intervallskalierte Daten kann hierbei sicherlich die Euklidische Distanz angesehen werden, welche die „geradlinige“ Distanz zwischen zwei Objekten auf der Basis der folgenden Gleichung berechnet:

$$\begin{aligned} d(x,y) &= \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_{n-1}-y_{n-1})^2+(x_n-y_n)^2} \\ &= \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \end{aligned} \tag{4.7}$$

$d(x,y)$ stellt dabei die Distanz zwischen den Objekten $x$ und $y$ und deren Gewichtsvektoren dar.

Diese Gleichung kann zur Distanzberechnung n-dimensionaler Gewichtsvektoren zweier Objekte angewandt werden. Für den Fall, dass nur eindimensionale Objekte miteinander verglichen werden sollen, kann obige Gleichung folgendermaßen gekürzt werden:

$$d(x,y) = |x_1 - y_1| \tag{4.8}$$

Eine besondere Variation der Euklidischen Distanz ist die quadrierte Euklidische Distanz:

$$\begin{aligned} d(x,y) &= (x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_{n-1}-y_{n-1})^2+(x_n-y_n)^2 \\ &= \sum_{i=1}^{n}(x_i-y_i)^2 \end{aligned} \tag{4.9}$$

Der einzige Unterschied bei dieser Variation ist lediglich der Wegfall der Quadratwurzel über die Summe der Quadrate der Differenzen aller Objekte einer Objektmenge.

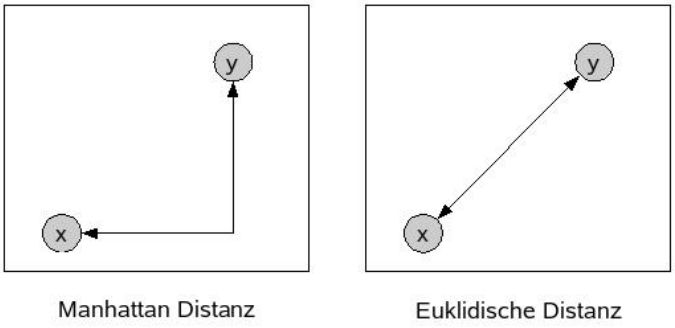



Abbildung 4.3.: Vergleich zwischen Euklidischer und Manhattan-Distanz

#### Manhattan-Distanz

Die Manhattan-Distanz bzw. City-block Distanz berechnet, anders als die Euklidische Distanz, nicht die „geradlinige“ Distanz, sondern, wie der Name der Distanzmetrik vermuten lässt, die Distanz zweier Punkte auf einem Raster. Das heisst, dass die Distanz nicht gerade zwischen zwei Punkten ermittelt wird, sondern unter der Annahme, dass die Punkte nur auf einem Pfad, der auf einem Raster liegt, erreichbar sind. Ausgedrückt wird dieser Zusammenhang folgendermaßen:

$$d(x,y) = \sum_{i=1}^{n} \mid x_i - y_i \mid \tag{4.10}$$

Zur Veranschaulichung der Unterschiede zwischen Manhattan- und Euklidischer Distanz soll Grafik 4.3 dienen.

#### Chebychev-Distanz

Im Gegensatz zu den anderen bereits erwähnten Distanzmaßen, berücksichtigt die Chebychev-Distanz lediglich die maximale Distanz zwischen zwei Elementen zweier Gewichtsvektoren (siehe Gleichung 4.11):

$$d(x,y) = max_i \mid x_i - y_i \mid \tag{4.11}$$

Daher bietet sich dieses Distanzmaß vor allem dann an, falls sich Unterschiede zwischen zwei Objekten zwischen einzelnen Dimensionen manifestieren im Gegensatz zu den vorhergegangenen Distanzmaßen, bei denen der Unterschied zwischen zwei Objekten als eine Gesamtzahl aller Dimensionen ermittelt wird. Daher wird dieses Distanzmaß hauptsächlich dann eingesetzt falls sich die Objekte untereinander auf Grund ihrer Daten nicht stark voneinander unterscheiden.

Die nachfolgenden Ähnlichkeitsmaße dienen zur Berechnung der Ähnlichkeit zwischen zwei Clustern im Gegensatz zu den vorhergegangenen Ähnlichkeitsmetriken, mit deren Hilfe die Ähnlichkeit zwischen zwei Objekten berechnet werden kann. Diese Metriken können auch für die Validierung von Clusteringalgorithmen verwendet werden, indem man mit ihrer Hilfe die Interclusterdistanz berechnet wie in Kapitel 5 zu sehen ist.

#### Single-linkage Distanzmaß

Das *single-linkage* Distanzmaß, auch *nearest neighbor* -Verfahren genannt, wird für die Clusterbestimmung im hierarchischen Clustering (siehe 4.1.2) eingesetzt. Dieses Distanzmaß beruht auf der Annahme, dass die Objekte zweier Cluster optimal zueinander sind, je geringer der Abstand der zwei nähesten Objekte ist. Dies wird in Gleichung 4.12 mathematisch ausgedrückt.

$$d(x,y) = min(d(x,y)) \tag{4.12}$$

Im Verfahren des hierarchischen Clusterings werden, basierend auf diesem Distanzmaß, Cluster kombiniert. Das Problem hierbei ist allerdings, dass Cluster verschoben werden, obwohl unter Umständen die Distanz der nähesten Objekte die durchschnittliche Distanz zwischen zwei Clustern nicht widerspiegelt und somit eine ungenaue Partitionierung entsteht.

#### Complete-linkage Distanzmaß

*Complete-linkage* stellt das genaue Gegenteil zu *single-linkage* dar, da bei diesem Distanzmaß die Ähnlichkeit zwischen zwei Clustern nicht auf der minimalen Distanz zweier Objekte beruht, sondern auf der maximalen Distanz zweier Objekte in unterschiedlichen Clustern. Genauso wie für das *single-linkage* -Maß gilt auch hier, dass diese Distanzmetrik nicht eingesetzt werden sollte, falls es viele Extremwerte in den Gewichtsvektoren der einzelnen Objekte gibt, weil sonst das Ergebnis des Clusterings verzerrt werden kann. Mathematisch wird dieses Ähnlichkeitsmaß wie in Gleichung 4.13 gegeben ausgedrückt.

$$d(x,y) = max(d(x,y)) \tag{4.13}$$

#### Average-linkage Distanzmaß

Als Mittelweg zwischen *single-linkage* und *complete-linkage* kann das *average-linkage*-Verfahren (4.14) angesehen werden, bei dem die Ähnlichkeit zwischen zwei Clustern auf der durchschnittlichen Ähnlichkeit aller möglichen Kombinationen von Objekten in beiden Clustern beruht. Im Gegensatz zu den beiden vorherigen Ähnlichkeitsmetriken, werden bei diesem Ähnlichkeitsmaß Extremwerte vor allem bei großen Datenmengen, die die Extremwerte im Durchschnitt nivellieren, nicht stärker berücksichtigt, weswegen die Qualität des Partitionierung höher ist.

$$d(x,y) = \frac{1}{n_x * n_y} * \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} d(x_i, y_j) \tag{4.14}$$

Allerdings ist dieses Ähnlichkeitsmaß auf Grund der zweifachen Summenberechnung rechenintensiver als die beidern vorangegangenen Distanzmaße, was sich andererseits wiederum in einer gesteigerten Clusteringqualität niederschlägt.

#### Centroid-linkage Distanzmaß

Dieses Distanzmaß beruht auf der Annahme, dass zwei Cluster sich ähnlicher sind, je näher die Zentroide der beiden Cluster zueinander sind. Mathematisch wird diese Annahme wie in Gleichung ausgedrückt.

$$d(x,y) = \| x - y \|_2 \tag{4.15}$$

Für die Bestimmung der Zentroiden $x$ bzw. $y$ kann die Gleichung 4.16 herangezogen werden.

$$x = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{4.16}$$

In der Gleichung 4.16 verweist $n$ auf die Anzahl der Objekte in einem Cluster x bzw. y und $x_i$ auf das i-te Objekt in dem entsprechenden Cluster.

## 4.2. Clustering genetischer Daten

Nach dieser allgemeinen Übersicht über wichtige Clusteringalgorithmen sowie Ähnlichkeitsmaße, wird im Folgenden auf Clusteringverfahren im Bereich der Biologie bzw. Medizin eingegangen, da sich das vorliegende Projekt mit dem Clustering genetischer Daten beschäftigt. Anders als bei diesem Projekt, bei dem nicht nur rein genetische Daten mit in den Clusteringprozess eingeflossen sind, sondern auch informationstheoretische Aspekte eine Rolle spielen, bauen die meisten Clusteringverfahren auf sogenannten *gene expression* Daten auf. Unter Genexpression versteht man im Allgemeinen den Prozess des Aufbaus von Proteinen bzw. RNA aus der im Zellkern vorhandenen DNA. Um diese Daten, die vor allem bei der Tumoranalyse und in der Pharmaindustrie zum Einsatz kommen, zu untersuchen gibt es sogenannte Microarrays oder Genchips, auf denen auf kleinstem Raum genetische Informationen gespeichert werden. Bei der Erstellung von Microarrays wird RNA auf eine feste Oberfläche, dem Chip, aufgetragen und mit Fluoreszenzfarbstoffen

markiert, um so ein Ablesen nach Experimenten, wie zum Beispiel der Veränderung von Temperatur, mittels Laser der Intensität der aufgetragenen Daten zu ermöglichen. Diese Intensitätswerte werden dann normalisiert um so spätere Verzerrungen in Analysen so gut wie möglich zu vermeiden. Im Bereich des Clusterings von Genen nehmen die einzelnen Zahlenwerte einer Genexpression eine eigene Dimension im Gewichtsvektor, der ein Gen repräsentiert, ein und werden an Hand dessen in Cluster aufgeteilt.

Wurden die Genexpressionsdaten ermittelt sollen sie mit Hilfe von Clusteranalysen strukturiert werden. Dazu wird oft entweder ein hierarchisches Clustering (vgl. Eisen u. a. (1998)) oder der k-means Algorithmus (vgl. Bolshakova u. Azuaje (2003) oder Potamias (2006)) verwendet. Wie bereits erwähnt haben diese beiden Algorithmen allerdings Nachteile. So muss zum Beispiel beim k-means Algorithmus die Anzahl der Cluster, auf die die Inputdaten aufgeteilt werden sollen, a priori bekannt sein. Es sei denn es werden mehrere Durchläufe mit veränderten Parametern gestartet und die Ergebnisse evaluiert, um so eine optimale Anzahl von Clustern zu erhalten. Allerdings ist das ein aufwendiger Prozess, der theoretisch für jede Analyse neu gestartet werden müsste und voraussetzt, dass sich die zu Grunde liegenden Daten stark gleichen. Auch der unterschiedliche Einsatz von Distanzmetriken beeinflusst das Ergebnis stark. In ihrem Projekt haben Bolshakova u. Azuaje (2003) zwar mehrere Distanzmetriken implementiert und neben dem k-means Algorithmus auch noch hierarchisches Clustering an Hand von *single linkage*, *complete linkage*, *average linkage*, *centroid linkage*, *average to centroids linkage* und *Hausdorff linkage* implementiert, allerdings beschränken sie sich in ihrer Applikation auf die Auswertung von Genexpressionsdaten als Datengrundlage. Dies führt leider zu einer eher einseitigen Analyse genetischer Daten, was hier in diesem Projekt behoben wird, da weitere Faktoren, die für die Genetik im Allgemeinen relevant sind, in den Clusteringprozess einfließen.

Auch wenn hierarchisches und k-means Clustering im Zusammenhang mit genetischen Daten sehr weit verbreitet sind, gibt es dennoch auch andere Algorithmen, die für das Clustering genetischer Informationen eingesetzt werden. Einer dieser

Algorithmen ist der CLICK-Algorithmus (Cluster identification via connectivity kernels) (siehe Shamir u. Sharan (2000)). Als Datengrundlage benutzt dieser Algorithmus genauso wie die soeben erwähnten Algorithmen Genexpressionsdaten, die zusammen einen Gewichtungsvektor für Gene vorgeben. In diesem Algorithmus gehen Shamir u. Sharan (2000) davon aus, dass Gene in einem gewichteten Graphen $G = (V, E)$ angeordnet werden können, wobei Gene Vertices und die Kanten zwischen den Vertices/Genen die Ähnlichkeit zwischen den verbundenen Knoten darstellen. Desweiteren ist eine Grundannahme, dass die Wahrscheinlichkeit, dass beide Knoten in einem Cluster auftreten können, von der Ähnlichkeit der Knoten zueinander abhängt. Um nun die Objekte zu clustern werden in jeder Iteration die Kanten entfernt, deren Ähnlichkeitswert einen vorgegebenen Schwellenwert unterschreiten. Die daraus resultierenden allein stehenden Objekte werden *singletons* genannt und am Ende des Algorithmus mit den optimalen *kernels* verbunden. Um nun *kernels*, sprich Subgraphen, die den Ausgangspunkt für spätere Cluster bilden, zu definieren, wird der Graph in Subgraphen partitioniert und für jeden Subgraphen wird sukzessive überprüft, ob seine Kanten nur auf ähnliche Objekte verweisen, das heisst die Kantengewichte bzw. die Ähnlichkeiten, die die Kanten repräsentieren, werden analysiert. Verweist ein Subgraph dabei nur auf ähnliche Objekte so wird er ein *kernel*. Nachdem so der Graph umstrukturiert wurde und *kernels* definiert wurden, werden die unter Umständen nicht verbundenen Elemente den entstandenen *kernels* zugewiesen. Diese Zuweisung erfolgt dabei unter Berücksichtigung der Durchschnittsähnlichkeit des *kernels*. Nachdem nun auch einzelne Elemente, sogenannte *singletons* auf die errechneten Cluster aufgeteilt wurden, werden Cluster, deren Ähnlichkeit einen definierten Schwellenwert überschreitet zusammengelegt, um so ein optimales Maß an Homogenität zu erreichen.

In ihrem Aufsatz haben Gat-Viks u. a. (2003) mehrere Clusteringalgorithmen - unter anderem SOM, CLICK und k-means - miteinander auf der Basis eines selbstentwickelten Qualitätsmaßes (CQS) evaluiert und kamen dabei zu dem Ergebnis, dass in allen Clusteringdurchläufen der k-means schlechter als die anderen eben erwähnten Algorithmen abgeschnitten hat. Die Datenbasis für den Vergleich be-

stand dabei aus Genattributen bzgl. der genetischen Funktionalität eines bestimmten Hierarchielevels innerhalb GO. Anders als in dem hier vorliegenden Projekt haben sie dabei allerdings auch wieder nur einen Aspekt genetischer Daten berücksichtigt. Während sich zum Beispiel Bolshakova u. Azuaje (2003) lediglich auf die Genexpressionsdaten konzentrierten, haben sich Gat-Viks u. a. (2003) andererseits nur auf die funktionellen Attribute von Genen, die in GO in einem bestimmten Hierarchielevel auftreten, beschränkt. Ein interessanter Schluss ihrer Arbeit besteht jedoch darin, dass „different biological attributes lead to different evaluations of clustering solutions“ (Gat-Viks u. a. (2003)). Auch wenn sich diese Aussage auf die Evaluierungsergebnisse von Clusteringalgorithmen bezieht, kann man dennoch Rückschlüsse ziehen, dass das Clustering für unterschiedliche Arten von Inputdaten unterschiedliche Ergebnisse liefert. Daher war es das Ziel dieses Projekts mehrere Arten von genetischen Faktoren gemeinsam zu untersuchen, um so ein breitgefächertes Clusteringergebnis zu erhalten, zumal für die Auswertung mit Hilfe anderer bioinformatischer Softwaretools ebenfalls mehrere Algorithmen und Datengrundlagen benutzt werden. So soll sichergestellt werden, dass die errechneten Cluster einen Großteil des Spektrums genetischer Informationen widerspiegeln.

Zusammenfassend kann man sagen, dass, auch wenn zum Beispiel Xiao u. a. (2003) mit SOM im Bereich Genetik arbeiten, der SOM-Algorithmus nur sehr selten zum Einsatz kommt, obwohl er für große Datenmengen gut geeignet ist. Insbesondere im Zusammenhang mit dem Clustering von genetischen Massendaten würde sich der SOM-Algorithmus besonders anbieten. Darüberhinaus werden fast ausschließlich Genexpressionsdaten zum Clustern von genetischen Daten herangezogen. Desweiteren ist die „interpretation of microarray results remains a crucial issue“ Slonim (2002), da „possible bias and confounding variables are substantial concerns“ Slonim (2002). Daher war es das Ziel dieser Arbeit sowohl zum einen nicht nur Genexpressionsdaten als Datengrundlage zu verwenden, sondern die vorliegenden genetischen Daten mit weiteren Faktoren anzureichern, und zum anderen diese, im Vergleich zu den meisten Clusteringanwendungen im Bereich Genetik, erweiterten Gewichtsvektoren mit einem SOM-Algorithmus zu clustern.

## 4.3. Daten und Parameter für das Clustering-Verfahren

Wie eben erwähnt soll für das hier erstellte Clusteringverfahren ein breiterer Gewichtsvektor als bei anderen, im vorangegangenen Kapitel erwähnten, Clusteringverfahren, eingesetzt werden. Im folgenden Kapitel wird daher detaillierter auf die in diesem Projekt verwendete Datengrundlage eingegangen. Allgemein lässt sich jedoch festhalten, dass die Gewichtsvektoren, die für das Clusteringverfahren gebildet werden, nicht wie üblich nur auf Genexpressionsdaten beruhen, sondern auch informationstheoretische Faktoren wie dem Informationsgehalt von SNPs oder der funktionellen Relevanz von Genen, die auf der Ähnlichkeit zwischen benutzerdefinierten genetischen Funktionen und den Funktionen, die einem Gen in GO zugewiesen wurden, beruht. Dies soll vor allem dazu dienen, eine breiter gefächerte Analyse von Gendaten zu ermöglichen und Variablen zu benutzen, die „represent the concept of similarity under which the study operates“ (Aldenderfer u. Blashfield (1985, 20)).

Seitens des Benutzers besteht der Input aus zwei Dateien, von denen eine die Namen aller Gene beinhaltet, die analysiert werden sollen, und die andere Signalpfade beinhaltet, in denen die Gene auftreten. Während die erste Datei lediglich eine Liste der Gen-IDs im HUGO-Format (Human Genome Organisation), das vom Gene Nomenclature Committee verwaltet wird, mit einer Gen-ID pro Zeile zum leichteren Lesen der IDs beinhaltet, ist die zweite Datei etwas komplexer. Diese beinhaltet eine Auflistung aller gefundenen Signalpfade, welche spezifisch für den jeweils verwendeten Signalpfadbrowser, das heisst einer Software zur Analyse von Signalpfaden, ist. Im Falle des Ingenuity Pathway Browsers (http://www.ingenuity.com), der in diesem Projekt verwendet wurde, sieht die Datei wie folgt aus:

Listing 4.1: Beispiel einer Signalpfad-Datei

```
1 1 C3AR1,C7ORF9,CA3,CALB2,CDKN2C,CEACAM1 (includes EG:634),
2 CHEK2,COL18A1,COX4I1,COX6A1,COX8B,CPN2,CSPG2,CST3,CSTA,
```

```
3  CYP11B2,CYP1B1,CYP2J2,EXPI,F2RL1,FBLN1,FRAT1,GLB1,
4  HERPUD1,JUN,MGP,PAX3,PITX1,PPGB,PPIB,PRL,PSME1,
5  PXDN,TPM3,WNT1  26  16  Cancer,
6  Tumor Morphology,Cell Death
7  2 ARG1,C1ORF24,C20ORF24,C21ORF7,C9ORF26,CBFA2T3,CCNB1,CIDEA,
8  CIDEC,CITED2,CORIN,CPT1B,CRYAB,CRYBB2,CRYGC,CTPS,CTSG,
9  CTSK,CUL7,FBXO2,G3BP,HDAC1 (includes EG:3065),IL13,MRC1,
10 MYC,PLS3,PPARG,RBX1,SERPINB4,SPRR1A,SRM,TGFB1,TSPAN7,ZC3HC1,
11 ZFP161  19  13  Cell Morphology,
12 Amino Acid Metabolism,Small Molecule Biochemistry
```

Da in diesem Projekt und der damit verbundenen IT-Infrastruktur lediglich der Ingenuity Pathway Browser zur Ermittlung von Signalpfaden zum Einsatz kommt, wird auf weitere Softwareprodukte bzw. Webseiten mit ähnlichen Funktionen nicht näher eingegangen. Die obene dargestellte Datei muss zuerst geparst werden, um sowohl die Anzahl der unterschiedlichen Signalpfade, in denen ein bestimmtes Gen auftritt als auch die Verbindungen eines Gens mit anderen Genen innerhalb eines Signalpfades zu ermitteln.

Ein weiterer Input sind, selbstverständlich, die eigentlichen Gendaten, die sich in einer Datenbank befinden (siehe dazu Kapitel 3.4.2). Sobald nun die Inputdateien geparst wurden, werden die entsprechenden Gendaten aus der Datenbank gelesen und weiter verarbeitet. Dabei werden nur die SNPs aus der Datenbank geholt, die sich auf einem Gen befinden, der als Input vom Benutzer angegeben wurde. Durch diese Vorauswahl von Genen wird die später zur Analyse benutzte Datenmenge stark reduziert. Dies ist insofern wichtig, da sich derzeit circa 820.000.000 SNP-Daten in der Datenbank befinden, deren komplette Auswertung erhebliche Performanzprobleme aufwerfen würde.

Um die enorme Datenmenge noch weiter einzuschränken, kann der Benutzer optional die Datenmenge um SNPs bereinigen, die einen bestimmten Schwellenwert im Zusammenhang mit dem *linkage disequilibrium* (LD) (siehe Kapitel 2.1.3) übersteigen. Neben der Reduzierung der Datenmenge gilt auch, dass „minimization

of pair-wise LD [...] should maximize informativity“ (Hampe u. a. (2003)). Dieser Schritt erfordert allerdings das frei zugängliche Softwarepaket PLink Purcell u. a. (2007), das in Kapitel 2.3.1 näher beschrieben wurde. In diesem Schritt werden alle SNPs, die sich auf den Inputgenen befinden zuerst in temporären Tabellen in der Datenbank gespeichert und im nächsten Schritt die damit verbundenen Phänotypdaten zusammen mit den entsprechenden Allelen in sogenannten PED-Dateien (Kapitel 2.2.1) exportiert. Da PLink nicht nur PED-Dateien, mit den Geno- sowie Phänotpydaten erwartet, sondern auch eine MAP-Datei, wird ebenfalls eine MAP-Datei, die die Position in kb (Kilobasen), das Chromosom, auf dem sich der SNP befindet sowie die ID des SNPs beinhaltet, exportiert. Diese beiden Dateien dienen dann als Input für das sogenannte SNP-Pruning, das PLink durchführt. Jedoch sind die beiden Dateien nicht das einzige was an PLink übergeben wird. Zusätzlich erwartet das Programm Angaben zu dem LD-Schwellenwert, der nicht übertroffen werden soll, die Fenstergröße, die bei jedem Schritt untersucht werden soll, sowie die Schrittgröße, d. h. die Anzahl der SNPs, in dem das Untersuchungsfenster weitergeschoben werden soll. Nachdem PLink die Berechnungen bzgl. der wahrscheinlichen LD-Werte der Input-SNPs beendet hat, liegen zwei Dateien vor, von denen die eine die SNPs beinhaltet, die den LD-Test bestanden haben. Die zweite Datei beinhaltet lediglich die SNPs, die den Schwellenwert übertroffen haben. Für das weitere Clustering werden dann lediglich die SNPs berücksichtigt, deren wahrscheinlicher LD-Wert unter dem Schwellenwert liegt.

Da zum Clustern von Daten ein Gewichtsvektor vorhanden sein muss bestand neben der Definition des Dateninputs noch das Problem, Faktoren zu ermitteln, die als Dimension in dem Gewichtsvektor, der ein Gen repräsentieren soll, dienen sollen und deren Gewichtung zu einander. Hierzu fanden mehrere Diskussionsrunden mit Experten aus den Bereichen Medizin, Biologie und genetischer Statistik statt mit dem Ergebnis folgende Faktoren in das Clustering einzubinden, die weiter unten in eigenen Unterkapiteln ausführlicher beschrieben werden:

- Rank von Genen basierend auf Googles PageRank-Algorithmus

- Die Phänotyp-Genotyp Korrelation
- Gewichtung der Publikationen bestimmter Gene
- Die funktionelle Relevanz von Genen
- Modellbasierte P-Werte, die mit Hilfe anderer statistischer Applikationen berechnet wurden

Diese Daten spiegeln zwar eine heterogene Sicht auf Gene wider, allerdings ist „discovering clinically significant knowledge from large-scale genome and molecular biology information [...] a complicated scientific process that draws from multiple overlapping sources of data“ (Wehbe u. a. (2009)). Insofern ist der Aspekt der Heterogenität des Gewichtsvektors gerechtfertigt, zumal all diese Faktoren aus zuverlässigen Quellen stammen und ein Gen an sich gut repräsentieren. Auch auf Grund der Tatsache, dass während der Berechnung der Distanz von Genen zueinander nur einzelne Dimensionen miteinander verglichen werden, sprich jeweils nur z. B. PageRank mit PageRank, besteht auch keine Gefahr eines ungerechtfertigten und das Ergebnis verzerrenden Vergleichs. Darüber hinaus hält (Xiong, 2006, 17) fest, dass es eine „overreliance on sequence information and related annotation“ gibt. Diesbezüglich bietet ein diversifizierter Gewichtsvektor wie er hier zum Einsatz kommt eine bessere und verlässlichere Sicht auf Gene.

Jedoch ist nicht nur die Auswahl geeigneter Faktoren, sondern auch deren Gewichtung im Bezug auf die Distanzberechnung von enormer Wichtigkeit. Diesbezüglich wurde wieder mit Hilfe von Fachexperten beschlossen, rein bioinformatische Faktoren wie P-Werte und die Phänotyp-Genotyp Korrelation stärker, um genau zu sein doppelt so stark, zu gewichten als die anderen Faktoren. Dies liegt darin begründet, da es sich bei diesen Faktoren um stichhaltige und weit verbreitete Maße in der Genetik handelt und die somit besser definierte Unterschiede zwischen Genen deutlich werden lassen.

## 4.4. Gewichtung von SNPs

Da die genetische Datengrundlage aus *single-nucleotide polymorphisms* besteht, dienen diese Daten als Ausgangspunkt für informationstheoretische Berechnungen der Relevanz von Genen und den darauf befindlichen SNPs vor einem bioinformatischen Hintergrund. Es werden nämlich nicht nur rein bioinformatische Werte, sondern auch informationstheoretische Aspekte, in das Clustering einbezogen. Dies soll vor allem dazu dienen eine quasi interdisziplinäre Auswertung von Gendaten zu ermöglichen im Gegensatz zu der Auswertung rein genetischer Faktoren wie etwa Genexpressionsdaten. Im Folgenden wird dazu der Ansatz näher erläutert, der dazu dient SNPs zu bewerten. Das Ergebnis dieser Berechnung, sprich der Informationsgehalt von SNPs, kann auch alleine für weitere Analysen, die rein auf die Analyse von SNP-Daten abzielen verwendet werden. Die berechneten Entropiewerte von SNPs werden dazu in der Datenbank gespeichert und können als Anhaltspunkt für die Auswahl von SNPs dienen. Das heisst, dass mit Hilfe dieses Wertes eine Vorauswahl von potentiell hochinformativen SNP-Daten getroffen werden kann.

### 4.4.1. Mathematische Grundlagen

Da SNPs lediglich aus zwei Allelen bestehen, besitzen sie leider per se einen sehr geringen Informationsgehalt, da sich für den Fall dass man alle denkbaren Mutationen berücksichtigt gerade 10 mögliche Kombinationen ergeben wenn die Reihenfolge der DNS-Stränge, auf denen sich die Allele befinden, unberücksichtigt bleibt, was bei den vorliegenden Daten grundsätzlich der Fall ist. Dieser geringe Informationsgehalt ist allerdings für die Durchführung von bzw. als Ausgangspunkt für Clusteringoperationen eher ungeeignet, da sich keine differenzierten und genauen Aufteilungen der Daten erreichen lassen, zumal die Mutationen von SNPs einzelner Individuen bzgl. einer SNP-ID zu einem großen Teil kaum voneinander abweichen. So gibt es maximal 3 Allelkombinationen pro SNP-ID in den zu Grunde liegenden SNP-Daten, was wiederum die Clusterqualität reduziert.

Daher war es notwendig eine Gewichtung von SNPs zu erstellen, die nicht ausschließlich auf die einzelnen Basenpaare innerhalb einer Individuenmenge abzielt, sondern darüber hinaus weitere Daten miteinbezieht. Deshalb wurde eine Gewichtung von SNPs basierend auf Claude Shannons Informationsentropie entwickelt. Shannons Informationsentropie beruht auf folgender mathematischer Gleichung für den Informationsgehalt, die besagt, dass je unwahrscheinlicher das Auftreten eines Ereignisses ist, desto höher ist sein Informationsgehalt, wobei in Gleichung 4.17 $i$ für ein Element einer Zeichenmenge steht und $H$ für den Informationsgehalt dieser Zeichenmenge (Shannon (1948)):

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{4.17}$$

Diese Gewichtungsmethode sieht vor, dass die Wahrscheinlichkeit für die Kombination zweier Allele zu einem SNP berücksichtigt wird und nicht wie in Shannons ursprünglicher Theorie die Wahrscheinlichkeit des Eintretens eines einzelnen Ereignisses. Als Daten dienen dazu alle Basenpaare, die einer SNP-ID zugeordnet sind, innerhalb der untersuchten Population. Zhao u. a. (2005) schlugen vor die Entropie eines biallelischen Markers $M$ mit den Allelen $A$ und $a$ (das häufigste bzw. seltenste Allel) mit folgender Formel zu berechnen:

$$H_M = -P(A) * logP(A) - P(a) * logP(a) \tag{4.18}$$

Die Gleichung 4.18 beinhaltet dabei Shannons ursprüngliche Gleichung zur Berechnung der Informationsentropie und wurde dahingehend erweitert, um sowohl die Wahrscheinlichkeit des Auftretens des häufigsten Allels $A$ als auch die des seltestens Allels $a$ zu berücksichtigen. Diese Gleichung lässt allerdings das Gütemaß für SNPs, die sogenannte *minor allele frequency* unberücksichtig. Wie bereits in Kapitel 2.1.1 erwähnt, dient die *minor allele frequency* dazu die Wahrscheinlichkeit zu bestimmen, dass ein genetischer Marker als SNP gesehen werden kann. Da dieses Maß auch im Zusammenhang mit der Berechnung des Informationsgehalts von

SNPs als Kriterium eingesetzt werden kann (vgl. Hwang u. a. (2007)), wurde die Gleichung 4.18 dahingehend erweitert, die *minor allele frequency*, das heisst das Verhältnis des seltenen Allels $a$ zum häufigen Allel $A$, zu berücksichtigen. Daraus ergibt sich die folgende Formel:

$$H_M = \frac{-P(a) * logP(a)}{-P(A) * logP(A)} \tag{4.19}$$

Diese Gleichung berücksichtigt die eben erwähnte *minor allele frequency* und bietet daher darüber hinaus ein besseres Informationsmaß als die Gleichung, die von Zhao u. a. (2005) entwickelt wurde.

Da die Informationsentropie für alle Basenpaare, die einer SNP-ID zugeordnet sind, aller untersuchten Individuen berechnet werden soll, dient folgende Gleichung als Ausgangspunkt für die Gewichtung von *single nucleotide polymorphisms*:

$$H_{M_{i\ldots j}} = \sum_{i=0}^{j} \frac{-P(a) * logP(a)}{-P(A) * logP(A)} \tag{4.20}$$

Die Gleichung 4.20 sieht vor, dass die Allele aller Individuen, im Falle der KORA-Studie von 1644 teilnehmenden Individuen, zueinander in Beziehung gesetzt werden. Daher werden alle Individual-SNPs, die einer SNP-ID zugeordnet werden, mit in die Berechnung aufgenommen und die Entropiewerte der einzelnen Individual-SNPs summiert, um so die komplette Informationsentropie eines SNPs zu erhalten. Lediglich die SNPs, die während der Genotypisierungsphase nicht bestimmt werden konnten und mit einem „-“ gekennzeichnet wurden, werden in dieser Phase nicht berücksichtigt.

### 4.4.2. Ergebnisse

Wie aus der Grafik 4.4 ersichtlich wird, liegen die höchsten Entropiewerte von *single nucleotide polymorphisms* mit vielen heterozygoten Basenpaaren - im Falle von RS2015993 1021 heterozygote Basenpaare - bei circa 480000, was einen hohen In-

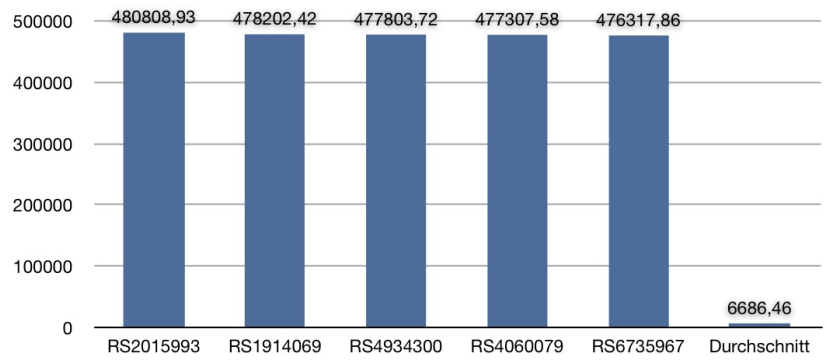



Abbildung 4.4.: Entropie der 5 informativsten SNPs im Vergleich zum Durchschnitt

formationsgehalt darstellt, denn „when it is applied to characterize DNA variation, entropy measures genetic diversity and extracts the maximal amount of information for a set of SNP markers“ (Zhao u. a. (2005)) . Wie zu erwarten beträgt der Informationsgehalt rein homozygoter SNPs 0 und bestätigt damit, die Erwartung, dass rein homozygote SNPs, d.h. SNPs mit Basenpaaren bestehend aus jeweils nur einer Aminosäure (z.B. 'AA'), keinen Informationsgehalt haben. Dies entspricht ebenfalls der informationstheoretischen Annahme, dass ein Würfel mit gleichen Seiten keinen Informationsgehalt hat, da die Ergebnisse aller Würfe zu 100% vorhergesagt werden können. So hat der SNP „RS9982203“ (siehe Grafik 4.5) (Entropiewert: 0,0045) lediglich ein heterozygotes Allelpaar. Im Durchschnitt haben die SNPs dieser Analyse einen Entropiewert von rund 6686,46. Einen Überblick wie sich die Werte aller zur Verfügung stehenden und berechneten ca. 500000 SNPs in ihrer Gesamtheit verteilen, bietet die Abbildung 4.6. Diese Abbildung zeigt deutlich, dass nur einige wenige SNPs einen sehr hohen Entropiewert haben also sehr informativ sind.

Die mit der oben erwähnten Gleichung errechneten Ergebnisse können dem Biologen wertvolle Hinweise in Bezug auf weiteres Auswertepotential von SNPs geben. Zum anderen dienen diese Ergebnisse auch dazu Gene vor deren Gewichtung

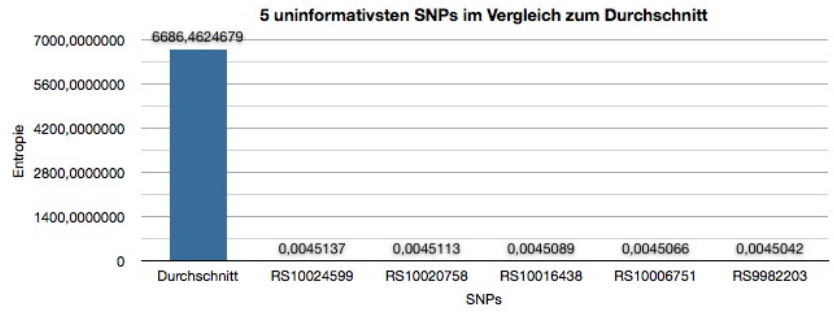



Abbildung 4.5.: Entropie der 5 uninformativsten SNPs im Vergleich zum Durchschnitt

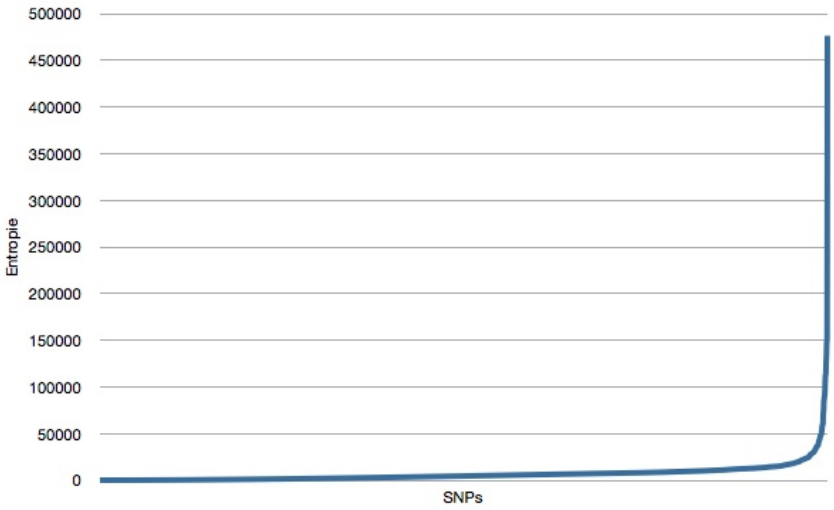



Abbildung 4.6.: Verteilung der Entropiewerte über alle ~500000 SNPs

zu initialisieren 4.5. Auch wenn dieser Initialwert keinen oder einen geringen Einfluss auf das tatsächliche Ergebnis des Rankings von Genen hat für den Fall, dass die Werte vor Erreichen der maximalen Iterationsanzahl konvergieren, so dient er dennoch als geeigneter Anfangswert, da sich in ihm der Informationsgehalt der auf einem Gen befindlichen SNPs widerspiegelt. Neben reinen SNP-Daten, deren Analyse und Ausgangspunkt für das Clustering genetischer Daten bzw. für die Gewichtung von Genen in diesem Abschnitt beschrieben wurde, spielt natürlich die eigentliche Berechnung des „Gewichts“ von Genen eine wesentliche Rolle für das Clustering. Daher wird im folgenden Abschnitt auf die Gewichtung von Genen näher eingegangen.

## 4.5. Ranking von Genen

Wie bereits erwähnt besteht ein Teil des Inputs aus einer Datei, die Informationen bzgl. des Auftretens von Genen in verschiedenen Signalpfaden beinhaltet (siehe Listing 4.1). Da ein Signalpfad im Endeffekt nichts anderes ist als ein Netz von Genen, die miteinander interagieren, kann man sich einen Signalpfad ebenso gut als Website vorstellen, deren einzelne Seiten mittels Hyperlinks miteinander verbunden sind. Die komplette Signalpfaddatei beinhaltet allerdings mehrere einzelne Gennetzwerke, in denen Gene mehr als einmal auftreten können und somit Hyperlinks zwischen verschiedenen Websites darstellen. Daher kann diese Inputdatei problemlos mit dem *World Wide Web* verglichen werden auch wenn die Datenmenge um einige Potenzen geringer ist. Nichtsdestotrotz lassen sich aus dem Aufbau der einzelnen Netze/Hyperlinkstrukturen und deren Abhängigkeiten untereinander wichtige und interessante Informationen extrahieren. Diese Linkinformationen dienen wiederum als Ausgangspunkt für die Gewichtung von Genen an Hand des *PageRank*-Werts, der von Page und Brin (vgl. Page u. a. (1998)) entwickelt wurde.

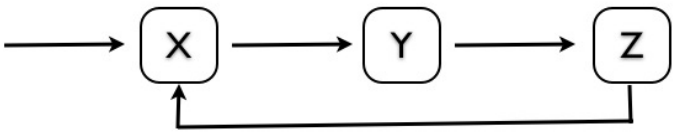



Abbildung 4.7.: Dangling Links

### 4.5.1. Der *PageRank*-Wert

Der PageRank wurde 1998 an der Stanford University von Lawrence Page und Sergej Brin entwickelt. In ihrem Aufsatz (Page u. a. (1998)) gehen Page und Brin davon aus, dass obwohl die Relevanz einer Website auf den ersten Blick ein eher subjektives Kriterium ist, man sie trotzdem objektiv messen kann. Dazu wird die Linkstruktur des Netzes genauer untersucht. Im Bezug auf die Linkstruktur eines durch Hyperlinks verbundenen Netzes kann man davon ausgehen, dass die Relevanz einer Seite proportional zu der Anzahl der Links ist, die auf diese Seite verweisen. D.h. je mehr Links auf eine Seite im WWW verweisen desto relevanter ist diese Seite für Suchanfragen. Dieser Zusammenhang zwischen Linkstruktur und Gewicht einer Seite wird in der folgenden Gleichung mathematisch ausgedrückt:

$$R(u) = (1-d) + d * \sum_{v \in B_u} \frac{R(v)}{N_v} \tag{4.21}$$

$d$ stellt dabei einen „Dämpfungsfaktor“ dar, mit Hilfe dessen sogenannte *dangling links* berücksichtigt werden sollen. Unter *dangling links* versteht man dabei Webseiten, auf die zwar Hyperlinks verweisen von der aber selber keine Links auf andere Seiten führen. Das könnte dazu führen, dass der PageRank-Wert endlos in einem Kreis aufeinander verweisender Seiten (siehe 4.7) berechnet wird, was wiederum das Ergebnis verfälschen würde.

In Bezug auf Gleichung 4.21 steht $R(v)$ für den Rang einer Seite, die auf die Seite $u$ verweist und $N_v$ ist die Anzahl der Links, die von der Seite $v$ wegführen. Mit Hilfe dieser Gleichung wird rekursiv der Rang bzw. die vermeintliche Qualität einer Seite

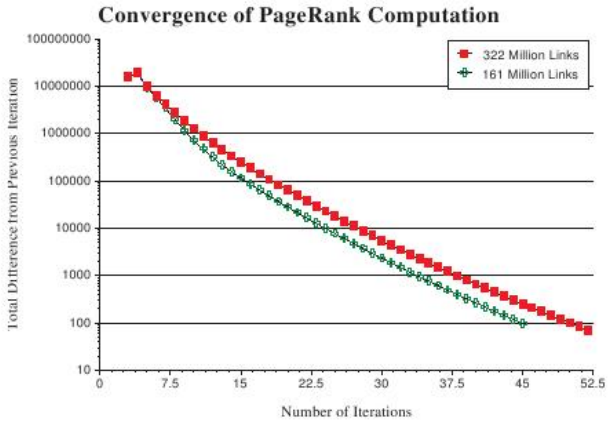



Abbildung 4.8.: Konvergenz der Rankingwerte abhängig von der Anzahl der Iterationen (Quelle: Page u. a. (1998))

im WWW basierend auf den Verbindungen mit anderen Webseiten berechnet. Dabei kann gesagt werden, dass je höher der PageRank eines Knotens ist, desto höher ist seine Qualität. Mit Hilfe dieser Berechnung konvergieren die berechneten Werte der einzelnen Webseiten nach einer bestimmten Anzahl von Iterationen (siehe Bild 4.8).

### 4.5.2. Vorteile des *PageRanks* im Vergleich zu anderen Rankingalgorithmen

Auf Grund des „explosive growth of the web“ (Manning u. a. (2009, 423)) des World Wide Web nahm das *Information Retrieval* einen immer größeren Stellenwert ein, um nur die relevantesten Dokumente für eine Suchanfrage im WWW zu finden. Zu diesem Zweck wurden etliche Algorithmen entwickelt, die dem Internetnutzer nur die Dokumenten präsentieren sollten, die am relevantesten im Bezug auf eine Suchanfrage sind. Im folgenden wird daher auf einige Algorithmen eingegangen und dargelegt warum der PageRank-Wert am passendsten für die Berechnung des Informationsgehaltes der zu Grunde liegenden genetischen Daten ist.

Neben dem PageRank zählt *Hypertext Induced Topic Selection* (HITS), der 1998 von Jon Kleinberg entwickelt wurde (Kleinberg (1998)), zu den wohl bekanntesten Ranking-Algorithmen, die die Linkstrukturen von Webseiten untersuchen. Der Ausgangspunkt bei dieser Berechnung des Rankings von Seiten/Knoten sind *Hubs* und *Authorities*. Unter *Hubs* versteht man dabei Webseiten bzw. Knoten im WWW, die auf viele andere authoritative Seiten/Knoten verlinken. *Authorities* hingegen sind Seiten, auf die viele Links verweisen. Diese beiden Arten von Seiten beeinflussen sich dabei gegenseitig. Das heisst, dass *gute Hubs* auf viele „gute“ *Authorities* verweisen und „ gute“ *Authorities* von vielen „guten“ *Hubs* verlinkt werden.

Bei dieser Berechnung des Rankings von Webseiten werden allen Seiten sowohl ein *Hub* -Gewicht $h_i$ und ein *Authority* -Gewicht $a_i$ zugewiesen, welche folgendermaßen berechnet werden.

$$x_p = \sum y_q \tag{4.22}$$

Diese Gleichung besagt nichts anderes als dass der *Hub*-Wert $x$ einer Seite $p$ sich aus der Summe der *Authority*-Werte $y$ derjenigen Seiten zusammensetzt, auf die die Seite $x$ verlinkt. Ähnlich verhält es sich mit dem *Authority* -Wert (siehe 4.23).

$$y_p = \sum x_q \tag{4.23}$$

Hier setzt sich der Wert $p$ einer *Authority* $y$ aus der Summe der *Hub*-Gewichte $q$ der Seiten zusammen, die auf $y$ verlinken. Im Falle des HITS-Wertes wird dabei $k$-Mal über einen Graph iteriert und dabei immer wieder die oben erwähnten y- bzw. x-Werte neu berechnet, um somit eine Übersicht über *Hubs* und *Authorities* zu bekommen. Nach dem Durchlaufen von $k$ Iterationen werden die Seiten entsprechend ihrer Werte sortiert, um so eine Übersicht darüber zu bieten, welche Seiten auf Grund der Linkstruktur von $x$ gefundenen WWW-Seiten am relevantesten sind.

Der Nachteil von HITS gegenüber PageRank im Zusammenhang mit Genen und Signalpfaden ist, dass der Inhalt der Dateien, die Signalpfadinformationen beinhalten, leider keinen Rückschluss darüber zulässt, welche Gene als *Hubs* bzw. als *Authorities* gelten, da nur ermittelt werden kann, welche Gene untereinander ver-

bunden sind. Es fehlt jedoch die Information welches Gen verlinkt und auf welches Gen verlinkt wird. Ähnlich verhält es sich mit dem Hilltop-Algorithmus (siehe Bharat u. Mihaila (1999)), der ebenfalls auf der Verbindung von authoritativen Seiten zu anderen Webseiten ausgeht, weswegen hier auf Grund der eben erwähnten Problematik nicht näher auf diesen Algorithmus eingegangen wird.

### 4.5.3. PageRank und Gene

Auf Grund der Graphstruktur eines genetischen Signalpfades kann man Parallelen zwischen Genen bzw. deren Strukturen in Signalpfaden und der Hyperlinkstruktur des *World Wide Web* ziehen. So kann man auch im Bereich der Genetik sagen, dass je mehr gewichtige Gene innerhalb eines Signalpfades auf ein Gen verweisen, desto wichtiger scheint dieses Gen zu sein. Zwar wird bei diesem Ansatz keine inhaltliche Gewichtung von Genen durchgeführt, da die Gewichtung nur auf der Graphenstruktur eines Signalpfades beruht, jedoch beinhalten die Signalpfaddateien inhaltliche Informationen, da der Biologe bzw. Mediziner die Signalpfadanalyse vor einem inhaltlichen Hintergrund durchführt. Das soll heissen, dass nur Gene, die bestimmte inhaltliche Voraussetzungen aufweisen, wie zum Beispiel, dass sie in metabolischen Signalpfaden auftreten, exportiert werden und somit als Ausgangspunkt für das Ranking dienen bzw. in die spätere Clusteranalyse einfließen. Sprich es findet eine manuelle Vorauswahl durch Experten vor einem medizinische/biologischen Hintergrund statt.

Allerdings fehlt eine wichtige Information in den Dateien, die die Signalpfade beinhalten, die derzeit auch nicht manuell gepflegt werden können. Es wird leider nicht spezifiziert welche Gene mit welchen anderen Genen interagieren. Stattdessen werden nur die Gene aufgelistet, die sich gemeinsam in einem Signalpfad befinden. Da aber nicht jedes Gen mit allen anderen innerhalb eines Signalpfades interagiert, ist das Fehlen dieser Information ein Problem. Um dieses Problem zu lösen, wurden im Falle der vorliegenden Arbeit Informationen, die in GeneOntology (GO) (http://www.geneontology.org) vorhanden sind, mit eingebunden. Jedoch

wurde nicht nur, wie in Morrison u. a. (2005), überprüft, ob zwischen zwei Genen eine Verbindung besteht, sondern es wurde auch die Zahl der Verbindungen zwischen zwei Genen bzw. Genprodukten mit Hilfe von aufbereiteten GO-Daten ermittelt, denn die Anzahl der Verbindungen zwischen zwei Genen/Genprodukten lässt Rückschlüsse auf die Qualität und Informativität eins Gens zu.

GO beinhaltet Daten bezüglich Genen und deren Produkte und bietet daher die Möglichkeit herauszufinden welche Gene mittels ihrer Produkte Ähnlichkeiten und daher Verbindungen aufweisen (siehe dazu Morrison u. a. (2005)). Diese Daten wurden aus den original GO-Daten extrahiert und so aufbereitet, dass sie problemlos aus der zu Grunde liegenden Datenbank ermittelt werden können. Zu diesem Zweck wurden sowohl die GO-IDs, die genetische(n) Funktion(en), Informationen darüber. aus welcher Teilontologie die Daten stammen, Synonyme der Gennamen sowie das Level der Gene innerhalb der GO-Taxonomie in der Datenbank gespeichert. Dabei kann es allerdings auch vorkommen, dass eine GO-ID mehrere Gene referenziert. Jedoch kann man sagen, dass „for a well annotated protein there are multiple lines of annotations“ (siehe Dimmer). Dies wiederum lässt die Schlussfolgerung zu, dass zwei Gene, die in GO viele Verknüpfungen miteinander aufweisen, auch hochqualitative Verbindungen im Bezug auf Informativität haben. Aus diesem Grund wird die Verbindungsinformation, die in der Ausgangsdatei fehlt, über diesen Umweg ermittelt. Um nun zu ermitteln wie viele Verbindungen zwei Gene aufweisen wird überprüft wieviel GO-Identifizierer zwei Gene gemeinsam haben:

$$C_{g_i \ldots g_j} = ID_i \cap ID_j \tag{4.24}$$

In Gleichung 4.24 steht dabei $C_{gi \ldots g_j}$ für die Anzahl der Verbindungen zwischen den Genen $i$ und $j$ und $ID_i$ bzw. $ID_j$ für die GO-IDs der Gene $i$ und $j$. Leider ist dieser dabei erhaltene Wert nur ein Näherungswert, da GO nicht 100% vollständig ist, was die darin enthaltenen genetischen Informationen angeht. Allerdings lässt sich aber somit zumindest ein Näherungswert ermitteln, der Auskunft darüber gibt inwieweit zwei Gene miteinander verbunden sind.

Die so ermittelte Anzahl an Verbindungen zwischen zwei Genen $i$ und $j$ fließt in die ursprüngliche Gleichung wie sie von Page und Brin entwickelt wurde mit ein (siehe Gleichung 4.25). Daher steht der Faktor $c$ für die Anzahl der Verbindungen zwischen den beiden Genen $u$ und $v$. Auf diese Weise werden Gene, die viele Verbindungen miteinander aufweisen, stärker gewichtet, da, wie bereits erwähnt, Gene mit häufigem Auftreten in GO ein hohes Maß an informativer Qualität aufweisen.

$$R(u) = (1-d) * |log(x_u)| + d * \sum_{v \in V} c * \frac{R(v)}{N_p} \tag{4.25}$$

Ausserdem spielen P-Werte, die der Benutzer/Experte manuell auswählen kann eine Rolle. Die Variable $x$ verweist daher in der obigen Gleichung auf den P-Wert des Gens $u$. Diese Variable wird allerdings nur berücksichtigt falls überhaupt P-Werte seitens Benutzer vorgegeben wurden. Der Logarithmus des P-Werts ist dabei notwendig, da der in diesem Falle berücksichtigte P-Wert unter der Annahme einer Nullhypothese, dass ein Gen keinen Einfluss auf einen Phänotypus hat, berechnet wurde und daher gilt:

$$p_{best} = \min(p_i p_j) \tag{4.26}$$

Da der Logarithmus für P-Werte < 1 einen negativen Wert ergibt, wird nur der absolute Betrag berücksichtigt. Diese Vorgehensweise setzt die oben erwähnte Annahme der Nullhypothese bzgl. des P-Werts eines Gens um und verbindet so sowohl die P-Werte eines Gens mit dem Rank eines Gens ähnlich wie in Morrison u. a. (2005), in deren Paper die Expressionswerte von Genen aus Microarray-Experimenten in eine ähnliche Rankingberechnung eingeflossen sind. $R_v$ steht in Gleichung 4.27 für den PageRank-Wert des Gens $v$ und $N_p$ verweist auf die Größe, das heisst die Anzahl der Gene, des Signalpfads $p$. Da allerdings die Inputdatei mehrere Signalpfade beinhaltet, müssen alle Signalpfade, in denen das zu untersuchende Gen auftritt summiert werden, was zur nächsten Gleichung führt.

$$R(u) = (1-d) * |log(x_u)| + d * \sum_{p \in P} \sum_{v \in V} c * \frac{R(v)}{N_p} \tag{4.27}$$

Hierbei gibt die Gleichung 4.27 an, dass für alle Gene $v \in V$ in allen Signalpfaden $p \in P$ der PageRank-Wert berechnet wird, um so den Rank eines Gens $u$ zu bestimmen. Sollte dabei der Fall auftreten, dass ein Gen $u$ in keinem Signalpfad vorhanden ist, so sinkt der Rank dieses Gens auf den Minimalwert von 0.05 was genau $1-d$ entspricht.

Der Dämpfungsfaktor $d$ bestimmt dabei, wie stark die Linkstruktur des Netzwerkes gewichtet wird im Verhältnis zu den Gewichten der einzelnen Seiten bzw. Genen (siehe Langville u. Meyer (2006, 59)). Je höher der Dämpfungsfaktor $d$ desto stärker wird die Linkstruktur gewichtet und desto schneller konvergieren die Rankingwerte von Seiten/Genen. Je mehr sich $d$ 0 nähert desto stärker wird dabei das Gewicht der einzelnen Gene gewertet. In ihrem Aufsatz verwenden Page und Brin (Page u. a. (1998)) einen Dämpfungsfaktor von 0.85, was jedoch für die hier zu Grunde liegenden Daten geändert wurde. Im Falle der hier durchgeführten Berechnungen wurde $d$ auf 0.95 erhöht, um so die tatsächliche Linkstruktur von Signalpfaden stärker zu gewichten. Der Grund dafür ist, dass im Fall der vorliegenden Arbeit die eigentliche Struktur der Signalpfade und der Verbindungen innerhalb der Signalpfade einen höheren Stellenwert in der Analyse der genetischen Daten hat als das Gewicht der einzelnen Gene, sprich nicht ein einzelnes Gen soll untersucht werden, sondern vielmehr die Signalpfadstruktur, in der die einzelnen Gene auftreten und miteinander verknüpft sind. Ähnlich wie im Falle von Webseiten konvergieren die Werte des PageRanks mit Genen als Datengrundlage ebenfalls nach nur wenigen Iterationen wie Abbildung 4.9 für das „schwerste“ bzw. informativste, sowie das uninformativste Gen und den durchschnittlichen Informationsgehalt von Genen basierend auf der Informativität von SNPs auf dem jeweiligen Gen zeigt.

## 4.6. Phänotyp-Genotyp-Korrelation

Ein weiterer wichtiger Faktor bei der Erstellung der Gewichtsvektoren, die als Input für das Clusteringverfahren dienen, ist der Wert der Phänotyp-Genotyp-Korrelation.

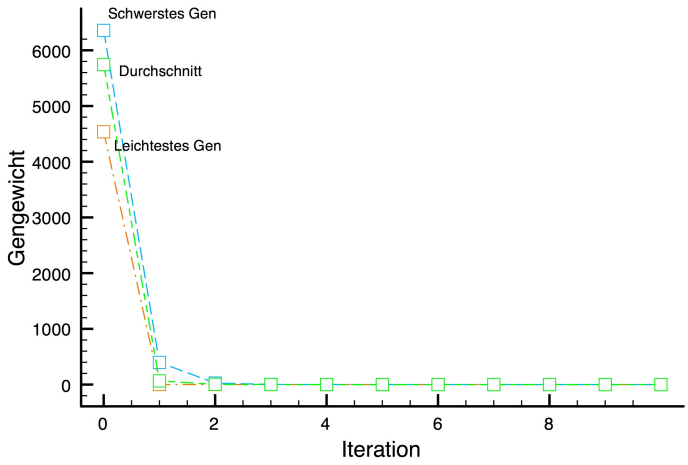



Abbildung 4.9.: Konvergenz von Genen

Dieser Wert gibt dabei Auskunft über die Wahrscheinlichkeit des Eintretens der Nullhypothese, dass ein Genotyp keinen Einfluss auf einen bestimmten Phänotyp hat. Dies bedeutet, dass je kleiner der errechnete p-Wert ist desto größer ist die tatsächliche Wahrscheinlichkeit, dass ein bestimmter Genotyp mit einer phänotypischen Ausprägung in Beziehung steht. Da für die Berechnung dieser Korrelation bereits diverse Softwaretools zur Verfügung stehen, wurde dazu das frei zugängliche Tool PLink Purcell u. a. (2007) (siehe Kapitel 2.3.1) eingesetzt. Als Input erwartet dieses Tool eine *ped*-Datei (siehe Kapitel 2.2.1). Diese Datei wird zur Laufzeit aus den zur Verfügung stehenden Daten erstellt. Dazu ist es nötig, dass der Benutzer die entsprechenden Phänotypen auswählt, die für die durchzuführende Analyse von Belang sind. Daraufhin werden die Phänotypdaten aller in der Datenbank enthaltenen Individuen gelesen und zusammen mit deren Genotypen in einer Datei gespeichert. Als zweiten Input erwartet PLink eine sogenannte *map*-Datei, die die Positionen der einzelnen SNPs auf dem menschlichen Chromosom enthält. Sobald diese beiden Dateien erstellt wurden, wird PLink mit folgendem Befehl aus der laufenden JAVA-Anwendung heraus aufgerufen: *plink* $--$*file PED−DATEI* $--$*map3* $--$*pheno PHAENOTYP−DATEI* $--$*assoc* Dieser Prozess berechnet die

Assoziation zwischen Genotyp und Phänotyp, die in den Inputdateien *PED−Datei* und *PHAENOTYP−DATEI* angegeben werden, und läuft als eigenständiger Thread im Hintergrund, so dass es möglich ist normal in GUI4DB weiterzuarbeiten. Allerdings muss das Clusteringverfahren auf das Ende dieses Prozesses warten, da es von den errechneten Daten abhängt. Als Output wird in diesem Schritt eine Datei namens „plink.qassoc“ erstellt, die die eigentliche Assoziation zwischen Phäno- und Genotyp beinhaltet.

Die mit Hilfe von PLink errechneten Phänotyp-Genotyp-Korrelationen werden im nächsten Schritt in der Datenbank zwischengespeichert. Zu Beginn des eigentlichen Clusterings werden die gespeicherten Werte normalisiert und in den Gewichtsvektoren der *Gene*-Objekte gespeichert und fließen somit in die Berechnung der Clusterzugehörigekt, genauer gesagt in die Distanzberechnung zwischen Genen, ein.

Darüber hinaus können optional weitere P-Werte, die mit Hilfe anderer Bioinformatikapplikationen oder Statistiksoftware, wie etwa JMP, berechnet worden sind, eingebunden werden. Zumal diese P-Werte an Hand verschiedener genetischer Modelle, sprich genetischer Grundannahmen bzgl. der zu Grunde liegenden Daten berechnet wurden, stellt diese Dimension des Gewichtsvektors eine interessante Datengrundlage dar. Es werden zwar hierbei quasi modellbasierte P-Werte in eine modellfreie Clusteringanalyse integriert, jedoch bietet dies dem Experten den Vorteil, mehrere P-Werte und deren Auswirkungen auf die Clusterzugehörigkeit von Genen zu untersuchen und so genauere Schlüsse ziehen zu können.

## 4.7. Publikationsgewichtung

Da man davon ausgehen kann, dass Gene, die häufig in der medizinischen bzw. biologischen Fachliteratur auftauchen, ein höheres Informationspotential haben als Gene, die nie oder nur selten referenziert werden, wird diese Informationsquelle ebenfalls mit in das Clustering einbezogen. Auch Wehbe u. a. (2009) ver-

wenden wissenschaftliche Aufsätze zusammen mit anderen Faktoren zur Erkenntnis medizinischen Wissens. Eine ähnliche Datengrundlage verwendet die Software Bibliosphere der Firma Genomatix (siehe Kapitel 2.3.2). Dazu wird mit Hilfe des Java APIs des *National Center for Biotechnology Information* (NCBI) EUtils (http://www.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html) zum Beispiel auf die Pubmed-Literaturdatenbank zugegriffen. Als Suchterme kann der Benutzer dabei aus dem kontrollierten Vokabular der GeneOntology Terme auswählen, die dann zu Abfragen nach folgendem Muster kombiniert werden:

$$Query = (G_{name}AND(t_1ORt_2OR...ORt_n))OR(G_{name}AND(u_1ORu_2OR...ORu_n)) \tag{4.28}$$

Wie in Gleichung 4.28 zu sehen ist, wird eine Abfrage aus dem Namen des Gens ($G_{name}$) und den einzelnen Termen ($t_1...t_n$) der ausgewählten GO-Terme ($t$ bzw. $u$), die mit OR-Operatoren verbunden werden, zusammengesetzt. Die Suchterme können dabei mehr als ein einzelnes Wort beinhalten, da es sich um Mehrwortbegriffe aus dem kontrollierten Vokabular der GeneOntology handelt. Diese Abfragen werden dann an medizinische Literaturdatenbanken wie etwa Pubmed geschickt und im Anschluss daran die Anzahl der gelieferten Resultate überprüft. Die Anzahl der gefundenen Publikationen werden dann in dem entsprechenden *Gene*-Objekt im Arbeitsspeicher gespeichert. Um die Werte, die in das Clusteringverfahren einfließen, nicht zu verzerren, wird der Publikationswert anschließend normalisiert indem mit Hilfe der folgenden Gleichung die Anzahl der Publikationen auf den Bereich [0,1] normalisiert werden.

$$W_{norm} = \frac{X_i - X_{min}}{X_{max} - Xmin} \tag{4.29}$$

Hierbei steht $W_{norm}$ für den normalisierten Publikationswert, $X_i$ für die Anzahl der Publikationen, die im Gen $i$ gespeichert wurden und $X_{min}$ bzw. $X_{max}$ für die minimale bzw. die maximale Anzahl von ermittelten Publikationen für alle zu clusternden Gene.

Problematisch hierbei ist allerdings, dass zum einen nur Abstracts von wissenschaftlichen Aufsätzen durchsucht werden können und zum anderen nicht sichergestellt werden kann, dass ein häufiges Auftreten eines Gens in Zusammenhang mit bestimmten medizinischen oder biologischen Termen nicht zwangsläufig bedeutet, dass dieses Gen tatsächlich positiv mit diesen Termen assoziiert ist. Das folgende fiktive Beispiel soll diesen Punkt näher verdeutlichen:

> Gene XYZ is strongly associated with the glycosphingolipid biosynthetic process.

Verweist ganz deutlich auf eine Assoziation mit dem *glycosphingolipid biosynthetic process* wohingegen das folgende Beispiel diesen Schluss nicht zulässt.

> Gene XYZ is definitely not associated with the glycosphingolipid biosynthetic process.

Der erste Satz weist direkt im Text eine deutliche Assoziation zu dem möglichen Suchterm *glycosphingolipid biosynthetic process* wohingegen der zweite Satz zwar syntaktisch ebenfalls mit demselben Suchterm assoziiert ist allerdings keine biologisch/medizinische Verbindung zu dem gesuchten Term aufweist. Da diese Problematik allerdings eher im Bereich des Text Minings anzusiedeln ist und die Analyse semantischer Verbindungen einzelner Terme zueinander den Rahmen dieses Projektes sprengen würde, wird dieses Problem vernachlässigt und die ermittelten Daten übernommen wie sie von den jeweiligen Literaturdatenbanken geliefert werden.

## 4.8. Funktionelle Relevanz von Genen

Als weitere Größe bzw. Dimension im Gewichtsvektor von zu clusternden Genen dient die funktionelle Relevanz von Genen. Funktionelle Relevanz bedeutet dabei inwieweit Funktionen von Genen wie sie in GO annotiert wurden mit gesuchten, als

relevant für eine Untersuchung erachteten Funktionen im Bereich der Genetik übereinstimmen. Hierzu muss der Benutzer zuerst aus einer Liste von genetischen Funktionen eine oder mehrere Funktionen auswählen, die für eine Untersuchung von besonderer Relevanz sind. Dazu bietet die Benutzerschnittstelle die Möglichkeit nach Funktionen zu suchen, die dann basierend auf dem Suchterm aus der Datenbank gelesen werden. Diese vollständige Liste kann vom Benutzer manuell gefiltert werden, sprich der Benutzer kann nur einige Funktionen aus der kompletten Liste auswählen. Diese ausgewählten Funktionen werden dann mit Hilfe des Dice-Algorithmus (siehe Gleichung 4.30) zur Ähnlichkeitsberechnung zwischen Dokumenten mit den Funktionen verglichen, die mit den Inputgenen in GO assoziiert sind. Der Dice-Algorithmus wurde deshalb eingesetzt, weil in Testläufen mit anschließender Expertenevaluierung bessere Ergebnisse zugeschrieben wurden als dem im Vergleich dazu eingesetzten Cover-Density-Algorithmus (siehe Clarke u. a. (2000)), der bereits im Datenbankserver PostgreSQL implementiert wurde.

$$d(a,b) = \frac{2 \mid T(a) \cap T(b) \mid}{\mid T(a) \mid + \mid T(b) \mid} \tag{4.30}$$

Nachdem die rein morphologische Ähnlichkeit zwischen den Genfunktionen und den Inputfunktionen ermittelt wurde, wird dieser Wert noch mit dem normalisierten Level eines Gens innerhalb der GeneOntology kombiniert. Der Grund dafür ist, dass Gene bzw. deren Produkte mehrmals in GO annotiert sein können wobei gilt, dass die Wichtigkeit eines Gens bzw. Genprodukts direkt proportional zur Anzahl der Annotationen steht (vgl. Dimmer). Zum anderen kann gesagt werden, dass je tiefer ein Gen in der Taxonomie zu finden ist desto genauer wurde das Gen beschrieben (siehe Lord u. a. (2002)), was wiederum annäherungsweise auf eine größere Genauigkeit der Beschreibung und daraus resultierend eine höhere Relevanz des Gens schließen lässt. Auch wenn Schlicker u. a. (2006) festhalten, dass „the depth of a term in the GO graph is not representative of the specifity of the underlying concept“ kann dieser Wert dennoch als Näherungswert und Ausgangspunkt für die Bestimmung der Genauigkeit eines Gens genommen werden, da in diesem Fall das Level

eines Gens, das mehrmals in GO annotiert wurde berücksichtigt wird. Wurde nämlich ein Gen $x$ mehrmals annotiert so wurden die Annotationen des Gens $x$ grundsätzlich hierarchisch strukturiert. Obwohl diese hierarchische Strukturierung und das Einfügen neuer Genannotationen in die Hierarchie manuell durchgeführt wird und somit der subjektiven Einschätzung des Experten unterliegt, so kann man doch davon ausgehen, dass die Annotation eines Gens in der Gesamtheit an Hand ihrer Genauigkeit in die GO-Hierarchie eingebunden werden. Ausserdem halten Schlicker u. a. (2006) zwar fest, dass „terms on the same rank [...] usually are not equally specific“ jedoch beruht diese Aussage auch wiederum auf subjektiven Einschätzungen. Der normalisierte Wert eines Levels eines Gens innerhalb der GeneOntology wird zur feineren Bestimmung der Ähnlichkeit eines Gens zu einer gesuchten Genfunktion mit dem Ähnlichkeitswert des Stringvergleichs multipliziert.

Ein weiterer Aspekt, der in Betracht gezogen wird ist das Auftretens der gesuchten Genfunktionen in GO, da man davon ausgehen kann, dass eine häufig auftretende Genfunktion bzw. ein häufig auftretendes Genprodukt eine höhere Relevanz hat als eine Genfunktion, die lediglich ein- oder zweimal auftritt. Da es sich bei GO um eine Ontologie mit einem kontrollierten Vokabular handelt, ist es offensichtlich, dass Genprodukte bzw. -funktionen mehr als einmal in GeneOntology auftreten. Daher wird das Auftreten einer Genfunktion ermittelt und ebenfalls mutlipliziert, was zu Gleichung 4.31 führt.

$$rel_{g_i} = sim(func_{g_i}, func_{user}) * lvl_{g_i} * occ_{func_{g_i}} \tag{4.31}$$

Dabei steht $rel_{g_i}$ für die funktionelle Relevanz eines Gens $i$, $sim(func_{g_i}, func_{user})$ für die Ähnlichkeit zwischen der Funktion(en) eines Gens $i$ und Inputfunktionen des Benutzers, $lvl_{g_i}$ für das normalisierte Durchschnittslevel eines Gens $i$ und $occ_{func_i}$ für das Auftreten einer Funktion $i$ in GO.

Nachdem die Gewichtsvektoren der Gen-Objekte mit Hilfe der vorangegangenen Faktoren aufgebaut wurden, werden sie an Hand der so errechneten Vektoren geclustert mit Hilfe der im nächsten Kapitel beschriebenen Clusteringsoftware.

## 4.9. Die Clusteringsoftware

Die Software, die das Clustering der genetischen Daten durchführt wurde mit dem Java Development Kit in der Version 6 implementiert. Damit sich die Clusteringsoftware reibungslos in das Gesamtprojekt einbinden lässt, wurde als Benutzerschnittstelle ein Plugin namens „iGene“ für GUI4DB (siehe Kapitel 3.6) erstellt. Auf diese Weise kann der Benutzer die Clusteringanalysen in einer gewohnten Umgebung durchführen und gleichzeitig andere datenbankorientierte Funktionen im Bezug auf klinische Studien ausführen.

Um ein einfach zu wartendes Programm zu erstellen wurde dabei großer Wert auf die für Java typische Objektorientierung gelegt. So müssen alle zu clusternden Daten das Interface *CNode* implementieren. Dieses Interface bietet Methodenrümpfe für alle clusteringrelevanten Funktionen wie etwa der Berechnung der Distanz zweier Objekte zueinander (siehe B.1).

Vor allem die Methode ***public double*** *getDistance(CNode n, DistanceFunction df)* ist hier von großer Bedeutung. Diese Methode soll für alle implementierenden Klassen die Distanz zu einem weiteren Objekt berechnen, das dieses Interface implementiert. Da es durchaus von Interesse sein kann, die Distanz zwischen zwei Objekten an Hand verschiedener Distanzmaße zu berechnen, wird eine *DistanceFunction* übergeben, die auf ein *enum* im Interface *CNode* verweist und in dem folgende Variablen für verschiedene Distanzmaße enthalten sind:

- EUCLIDEAN_DISTANCE - berechnet die Euklidische-Distanz zwischen zwei Objekten
- MANHATTAN_DISTANCE - berechnet die Manhatten-Distanz zwischen zwei Objekten
- CHEBYCHEV_DISTANCE - berechnet die Chebychev-Distanz zwischen zwei Objekten

Um die Software so abstrakt wie möglich zu halten und um sie zu einem späteren Zeitpunkt eventuell zu erweitern, wurde die Software so entworfen, dass jede implementierende Klasse für alle möglichen Distanzmaße die Distanz zu anderen Objekten berechnen kann ohne dass dabei im Code des eigentlichen Clusteringalgorithmus etwas geändert werden muss.

Obwohl sich der Hauptteil dieses Projektes mit dem Clustering von Genen mit Hilfe des SOM-Algorithmus beschäftigt, wurde neben diesem auch noch ein K-Means-Algorithmus implementiert, um so eine Vergleichsgröße zu haben und um Aussagen darüber treffen zu können, ob ein ausgefallenerer Clusteringalgorithmus wie der SOM-Algorithmus wirklich Vorteile gegenüber schnell zu implementierenden und weitverbreiteten Algorithmen wie beispielsweise dem K-Means-Algortihmus bietet. Daher wir im folgenden auf die Implementierung beider Algorithmen näher eingegangen.

### 4.9.1. Implementierung des SOM-Algorithmus

Auch wenn im Bereich der Bioinformatik bzw. der Genetik der SOM-Algorithmus noch nicht allzu stark vertreten ist, sondern eher ein Schattendasein fristet und nur von wenigen Forschergruppen (siehe zum Beispiel Tamayo u. a. (1999)) angewandt wird, so hat dieser Algorithmus trotz seiner Komplexität im Vergleich zu anderen Clusteringalgorithmen wie etwa dem k-Means Algorithmus ein großes Potential, um vor allem performanzkritische Clusteringanalysen und Analysen von großen Datenmengen durchzuführen (siehe zum Beispiel Xiao u. a. (2003), He u. a. (2004)).

Im Zentrum des SOM-Clusterings steht die Klasse *KohonenNetworkHandler* (siehe dazu auch das Klassendiagramm in A.1), da sie sich um die ganze Verwaltung der Knoten, sowohl der Inputknoten als auch der Knoten in der Kohonenkarte sowie um den Aufbau bzw. das Training der SOM, kümmert. Mit Hilfe der Methode *train (* ***int*** *amountOfNodes, CNode[] input)* kann ein Kohonennetzwerk trainiert werden. Dabei wird zuerst ein Netzwerk mit *amountOfNodes* Knoten erzeugt, das dann sukzessive in mehreren Iterationen trainiert wird. Die Anzahl der Iterationen

wird dabei als Parameter in einer Konfigurationsdatei gespeichert und dem Plugin iGene übergeben. Dazu wird wie bereits im Zusammenhang mit SOM-Clustering beschrieben, zuerst ein Inputknoten zufällig aus dem Array *CNode[] input* ausgewählt und der erzeugten Kohonenkarte in Form der Klasse *KohonenNetwork* als Input übergeben. Diese Klasse führt dann zuerst die Distanzberechnungen zwischen dem Inputvektor und den Knoten innerhalb der Karte durch, um so die sogenannte *best-matching unit* (BMU) zu definieren.

Nach diesem ersten Trainingsschritt wird der Nachbarschaftsradius um die BMU angepasst, um so mit steigenden Iterationen die Anzahl der zu trainierenden Knoten innerhalb der Kohonenkarte zu minimieren und so ein optimales Netzwerk zu erhalten. Dazu wird bei der ersten Iteration der Radius folgendermaßen initialisiert:

$$R = \frac{d(n_0, n_{k-1})}{1.15} \tag{4.32}$$

In Gleichung 4.32 steht dabei $d(n_0, n_{k-1})$ für die Distanz der beiden am weitesten von einander entfernten Objekten innerhalb der Kohonenkarte, der Wert 1.15 ist ein konstanter Wert, der dazu dient den Anfangsradius auf einen annehmbaren Wert zu setzen, was auf Kohonens Aussage beruht, dass „the initial radius of $N_c$ can even be more than half the diameter of the network“ (Kohonen (1997, 88)). Der Distanzwert $d(n_0, n_{k-1})$ wird berechnet nachdem die Karte aufsteigend sortiert wurde, was auch deshalb gemacht wird, um so die Rechenzeit bei der Bestimmung der Knoten, die innerhalb eines bestimmten Radius um die BMU liegen und somit trainiert werden, zu reduzieren. Für alle weiteren Iterationen > 1 wird der Radius mit einem parametrisierbaren Fakor multipiliziert, der derzeit 0.998 beträgt, um so den Radius bei steigenden Iterationen monton sinken zu lassen.

Der Lernfaktor $\alpha(t)$ wird bei der ersten Iteration auf 0.999 gesetzt und für jede nachfolgende Iteration um $0.02^{\frac{t}{max_{iteration}}}$ reduziert. Dies entspricht zwar micht dem ursprünglichen Vorschlag von Kohonen (1997, 88), bietet aber dennoch eine Möglichkeit den Wert stetig für $t \longrightarrow max_{iteration}$ zu reduzieren. Bei der Größe des zu erwartenden Netzwerkes von nicht mehr als 400 Genen, sind diese Punkte, sprich

das Anpassen der Parameter, ohnehin vernachlässigbare Größen, da „if the SOM network is not very large (say, a few hundred nodes at most), selection of process parameters is not very crucial“ (Kohonen (1997, 88)).

Nach der Berechnung der BMU werden die Knoten um die BMU angepasst, sprich die Knoten in einem bestimmten Radius um die BMU werden trainiert. Nach der Definition der BMU und der Anpassung des Radius um die BMU werden die Gewichtsvektoren der Knoten, die innerhalb des Radius um die BMU liegen angepasst. Dazu wird zuerst der Nachbarschaftsparameter mit Hilfe der Gausschen Nachbarschaftsfunktion (siehe Kapitel 4.4) neu berechnet. Nachdem die Nachbarschaftsfunktion angewandt wurde, wird für jeden Knoten innerhalb des Radius das Gewicht mit Hilfe der Gleichung 4.3 angepasst. Diese Schritte werden solange wiederholt bis die Anzahl der maximalen Iterationen, die im Bereich von 100000 Iterationen liegt, erreicht wurde.

Nachdem auf diese Weise nach und nach das Kohonennetzwerk im Hinblick auf die zu erwartenden Inputdaten verbessert wurde, wird das trainierte und an die Inputdaten angepasste Netzwerk serialisiert. Das heisst, dass die notwendigen Daten auf dem lokalen Rechner in einer Datei gespeichert werden, um sie für spätere Clusteringanalysen wieder laden zu können. Selbstverständlich besteht die Möglichkeit vor jeder Analyse ein Netzwerk zu erstellen und zu trainieren. Sollen nun tatsächliche Inputdaten geclustert werden, so wird automatisch das letzte serialisierte Kohonennetzwerk in den Speicher geladen und zum Clustering der Inputdaten verwendet.

Für das eigentliche Clustering wird dann das optimierte und in einer Datei gespeicherte Kohonennetzwerk wieder in den Arbeitsspeicher geladen. Diesem Netzwerk wird ein Array von zu clusternden Objekten, die das Interface *CNode* implementieren als Parameter übergeben, woraufhin der eigentliche Clusteringprozess startet. Dazu wird zuerst das Kohonennetzwerk sortiert, um so nicht immer über das komplette Netzwerk iterieren zu müssen bevor ein optimaler Knoten gefunden wird, was unter Umständen bei großen Inputmengen enorm Rechenzeit sparen kann. Da es sich bei den Vektoren um fünfdimensionale Gewichtsvektoren handelt, wurde eine eigene Sortierfunktion implementiert, da die Standardsortiermöglichkeiten des

*Java Development Kits* nicht auf mehrdimensionale Vektoren angewendet werden können.

Wurden alle Knoten des Netzwerks sortiert, so wird für jeden Inputknoten der optimale Knoten des Netzwerks, sprich der Knoten mit der geringsten Distanz zum Inputknoten, mit Hilfe einer angegebenen Distanzfunktion errechnet und diesem optimalen Knoten zugewiesen. Da das Netzwerk bereits trainiert ist, um Objekte eines bestimmten Typs zu klassifizieren, reicht bei diesem Clusteringansatz lediglich ein Durchgang, um alle Inputknoten ihren optimalen Klassen zu zuweisen. Im Gegensatz zu anderen Clusteringverfahren wie etwa dem *k-means* Clustering oder dem hierarchischen Clustering stellt das eine enorme Performanzsteigerung dar. Denn auch wenn die Trainingsphase unter Umständen lange dauern kann, so ist die Zeit- und Ressourcenersparnis bei diesem Clusteringansatz nicht zu übersehen, da für Objekte eines bestimmten Typs einfach ein trainiertes Netzwerk geladen werden kann. Vorausgesetzt, das trainierte Netzwerk passt zu den gegebenen Inputdaten.

### 4.9.2. Implementierung des k-means Algorithmus

Um eine Vergleichsgröße bzgl. der Qualität des oben beschriebenen SOM-Algorithmus im Hinblick auf das Clustering von genetischen Daten zu haben wurde der k-means Algorithmus in Java mit den gleichen Inputdaten implementiert. Wie der SOM-Algorithmus wurde auch der k-means Algorithmus auf der Basis des Java Development Kits in der Version 6 implementiert auf der Grundlage des Schnittstellendesigns für genetische Daten.

Als Ausgangspunkt für das k-means Clustering dient die Klasse *KMeansClusterer* (siehe dazu auch das Klassendiagramm in A.2), die mit der Funktion ***public void*** *cluster (CNode[] input, String analysis, CNode.DistanceFunction df)* ***throws*** *SQLException* die übergebenen Daten klassifiziert. Dazu werden entweder bis zum Erreichen einer vorher festgelegten maximalen Anzahl an Iterationen oder bis keine Objekte mehr zwischen Clustern verschoben werden, die Gene den zu dem Zeitpunkt optimalen Klassen zugewiesen. Dazu wird in jeder Iteration der Array an

*CNode*-Objekten durchlaufen und für jedes Objekt der optimale Cluster errechnet auf der Basis einer parametrisierbaren Distanzfunktion. Nachdem die beste Klasse für ein Objekt gefunden wurde, wird dieses dorthin verschoben und der Zentroid der Zielklasse als Durchschnitt der einzelnen Dimensionen der Objekte im Cluster neu berechnet. Diese Neuberechnung führt dazu, dass für die nachfolgenden *CNode*-Objekte die Ausgangssituation im Bezug auf die vorhandenen Klassen ändert. Dieses Vorgehen bewirkt daher, dass die Inputdaten nicht kontinuierlich denselben Klassen zugewiesen werden, sondern immer wieder neu verteilt werden, solange bis sich die Klassen an die Inputdaten angenähert haben. Diese Annäherung findet statt, da das Gewicht des Zentroiden einer Klasse auf der Summe der Gewichtsvektoren der Objekte in einer Klasse beruht. Das Problem ist allerdings hierbei, dass auf Grund der zufälligen Reihenfolge der Inputobjekte unter Umständen Verzerrungen der Klassen auftreten können. Dies wiederum führt zu einem Mangel an Stabilität des Clusteringalgorithmus, was in unterschiedlichen Ergebnissen mit denselben Inputobjekten beobachtet werden kann.

# Kapitel 5.

# Evaluierung

Nachdem im vorangegangenen Kapitel Clustering im Bereich der Genetik sowie das hier entwickelte Clusteringverfahren näher beschrieben wurden soll, das folgende Kapitel dazu dienen, zuerst eine Einführung in die Evaluierung von Clusteringergebnissen und der damit verbundenen Problematik zu liefern und darauf aufbauend, die Ergebnisse der Validierung des hier vorliegenden Clusteringalgorithmus darzustellen und auszuwerten. Prinzipiell versteht man unter Evaluierung die objektive Überprüfung, ob ein Gegenstand, Prozess oder ein Zustand die in ihn gesetzten Voraussetzungen erfüllt. Im Bereich der Informationstechnologie werden Evaluierungen vor allem zur Qualitätssicherung von Softwareprodukten eingesetzt wie zum Beispiel zur Überprüfung der Benutzerfreundlichkeit, Sicherheit und Schnelligkeit von Internetbrowsern. Für diese Arten von Evaluierungen sind jedoch die Qualitätskriterien und Bewertungskriterien objektiv festgelegt wie etwa im Falle der ACID-Tests für Internetbrowser. Im Falle von Clusterevaluierungen stellt dich die Sachlage anders dar. Obwohl es für die Evaluierung von Clusterergebnissen ebenfalls notwendig ist, objektive Metriken zu haben, um die Qualität der Ergebnisse zu überprüfen, so ist die Überprüfung von Clusteringergebnissen auch ein subjektives Unterfangen. Denn nicht immer lässt sich eindeutig klären, ob die Zuordnung eines Objekts zum Cluster $A$ besser ist als die Zuordnung zum Cluster $B$, da beide vertretbar sein können. Hinzu kommt noch, dass „a large number of these evaluation measures are capable of validating a narrow range of in-house clustering methods only“ (He u. a. (2004)). Das heisst, dass die Validierung von Clustering-

ergebnissen genauso speziell ist wie die zu Grunde liegenden Daten. Ausserdem gilt, dass „typically, different clustering algorithms yield different clustering solutions on the same data, and there is no agreed upon guideline for choosing among them“ (siehe Gat-Viks u. a. (2003)). So hängt das Ergebnis der Validierung genauso wie das Ergebnis des Clusterings selbst von der verwendeten Distanzmetrik ab, was folglich eine objektive Aussage bzgl. der Qualität des Clusterings erschweren kann. Auch die abstrakten objektiven Qualitätskriterien wie Homogenität und Heterogenität haben einen eher relativen Stellenwert. Wurden nämlich während einer Clusteringanalyse alle Objekte in eigene Cluster verschoben, das heisst ein Cluster pro Objekt, so hat dies ein optimales Maß an Homogenität zur Folge. Selbstverständlich hätte eine derartige Analyse folglich ein geringes Maß an Heterogenität. Dieses Extrembeispiel soll daher auch nur verdeutlichen, dass abhängig von der Datengrundlage und der Fragestellung, die der Analyse vorausgeht, ein subjektives Maß zur Evaluierung eingesetzt werden muss. Natürlich gibt es, wie unten zu sehen sein wird, weitverbreitete Evaluierungsmetriken, mit deren Hilfe es möglich ist, die Qualität von Clusteringanalysen zu bestimmen. Nichtsdestotrotz, muss erwähnt werden, dass diese Metriken nur Anhaltspunkte liefern und deshalb mehrere Metriken eingesetzt werden sollen und, falls möglich, die Daten von Experten überprüft werden sollten. Letzteres ist allerdings bei großen Datenmengen nur bedingt möglich.

Grundsätzlich lassen sich vor dem Hintergrund der hier durchgeführten Clusteranalysen zwei grundlegende Arten von Evalierungen unterscheiden. Zum einen gibt es allgemein gültige, generische Evaluierungsheuristiken wie etwa das Silhouette-Maß (siehe 5.1.2), zum anderen gibt es domänenspezifische Evaluierungsalgorithmen, die sich nur für die Evaluierung von Clusteringergebnissen einer spezifischen fachlichen Domäne eignen. Erstere beruhen dabei auf mathematischen Erkenntnissen, die eine optimale Einteilung von Objekten in Cluster widerspiegeln. Hierbei muss allerdings berücksichtigt werden, dass man unter einer „optimalen Einteilung“ keineswegs immer das gleiche versteht. Zwar sind ein hoher Grad an Homogenität und Heterogenität eine grundlegende Voraussetzung für eine positiv zu bewerten-

de Clustereinteilung, allerdings können diese Maße unterschiedlich berechnet und interpretiert werden. Dazu gibt es mathematische Gleichungen, die jeweils die Homogenität und Heterogenität auf der Basis unterschiedlicher Annahmen berechnen. So kann zum Beispiel die Distanz zwischen zwei Clustern unter Berücksichtigung der maximalen Distanz zwischen den Objekten der Cluster (siehe Kapitel 4.1.5) oder der durchschnittlichen Distanz zwischen den Objekten zweier Cluster (siehe Kapitel 4.1.5) berechnet werden. Domänenspezifische Evaluierungsalgorithmen dagegen beruhen auf gültigen fachspezifischen Annahmen über die geclusterten Objekte, die annerkannten Gesetzmäßigkeiten folgen und die sich nicht generisch, also domänenübergreifend, in mathematischen Gleichungen ausdrücken lassen.

Da eine Vielzahl von Clusteringalgorithmen verfügbar ist, ist es sehr komplex eine zuverlässige Aussage über die Qualität und Stabilität von Clusterinalgorithmen und welcher Algorithmus besser für bestimmte Daten geeignet ist als andere zu treffen, denn „different clustering algorithms often lead to markedly different results“ Levine u. Domany (2001, 2573), wobei nicht eindeutig gesagt werden kann, dass ein Ergebnis weniger richtig als ein anderes ist. Obwohl verschiedene Algorithmen zum Teil sehr unterschiedliche Ergebnisse für die gleiche Datengrundlage liefern, ist dies jedoch bei grundlegend unterschiedlichen Algorithmen wohl eher eine logische Folge, da sie unter Umständen dieselben zu Grunde liegenden Daten komplett unterschiedlich interpretieren. Dieser Aspekt allerdings erschwert die objektive Auswertung von Clusteringergebnisse ungemein. Im Falle von per definitionem instabileren Algorithmen wie etwa k-Means (siehe Kapitel 4.1.1), deren Clustereinteilung auf der Reihenfolge der geclusterten Objekte beruht, kann es sogar vorkommen, dass unterschiedliche Ergebnisse für dieselben Daten berechnet werden, was die Erstellung eines objektiven Qualitätsmaßes noch zusätzlich erschwert.

Ein weiterer kritischer Aspekt bei der Auswertung von Clusteringergebnissen ist der Einfluss einzelner Parameter auf den Clusteringalgorithmus. So muss bei der Evaluierung der Ergebnisse in Betracht gezogen werden, dass verschiedene Parameter das Ergebnis beeinflussen, was sich wiederum allerdings nicht notwendiger-

weise auf das tatsächliche Ergebnis niederschlagen muss wie zum Beispiel im Falle von Parametern, die den Detaillierungsgrad des Clusterings bestimmen.

Auch die Tatsache, dass die Evaluierungsergebnisse wiederum von den jeweiligen Validierungsalgorithmen abhängen, erschwert die objektive Auswertung von geclusterten Daten. So kann ein Evaluierungsalgorithmus einem Clusteringalgorithmus mit bestimmten gesetzten Parametern eine hohe Qualität bescheinigen während ein anderer Evaluierungsalgorithmus eine sehr viel geringere Qualität des Algorithmus ermittelt. Hier gilt es nun mit Expertenwissen zu entscheiden welcher Algorithmus nun wirklich besser ist, indem man die Validierungsergebnisse nur als Anhaltspunkt für eine Expertenevaluierung nimmt. Diese Expertenevaluierung stößt allerdings selbst an ihre Grenzen, sollten dem Clusteringalgorithmus große Datenmengen zu Grunde liegen.

Im Falle von partitionierenden Algorithmen wie etwa k-means im Gegensatz zu hierarchischen Clusteringalgorithmen oder neuronalen Netzen besteht noch das Problem, dass die Anzahl der Cluster a priori bekannt sein muss. Hierfür können Validierungsmetriken ebenfalls nützlich sein, da durch das Anpassen der Clusteranzahl die optimalste Anfangsanzahl an Cluster gefunden werden kann. Im Falle von Clusteringalgorithmen, die, wie self-organizing maps, auf neuronalen Netzen beruhen, kann dieser Aspekt allerdings vernachlässigt werden, da sich das neuronale Netz ohnehin dynamisch an die zu clusternden Daten anpasst.

Bei der hier vorliegenden Datengrundlage wird die Evaluierung noch dahingehend erschwert, dass eine aussagekräftige biologisch-medizinische Validierung nur mit Hilfe von in-vitro- oder Tierversuchen möglich ist. Dieser Prozess nimmt dabei allerdings mehrere Jahre in Anspruch bis schlüssige und zuverlässige Ergebnisse vorliegen, weswegen im Folgenden nur auf heuristische und statistische Validierungsergebnisse eingegangen wird. Darüber hinaus darf auch der finanzielle Aspekt nicht vernachlässigt werden. Während zum Beispiel für in-vitro- oder Tierversuche sowohl viel Zeit als auch viel Geld investiert werden muss, um zu zuverlässigen Ergebnissen zu kommen, sind die benötigten Ressourcen - sowohl im Hinblick auf Zeit und Hard- bzw. Software - für heuristische Validierungsansätze bedeutend ge-

ringer. Deshalb werden auch im Bereich der Bioinformatik zumeist heuristische Validierungsansätze auf der Basis von Genexpressionsdaten verfolgt wie etwa in Bolshakova u. a. (2005), Bolshakova u. Azuaje (2003), Jiang u. Zhang (2002), Yeung u. Haynor (2001) oder auch Levine u. Domany (2001).

Nichtsdestotrotz ist es nötig, Clusteringergebnisse zu validieren, um somit eine Aussage über deren Qualität treffen zu können und falls nötig den Algorithmus anzupassen, um eine spätere Analyse der Ergebnisse auf eine solide Grundlage stellen zu können.

## 5.1. Validierungsmetriken

Der folgende Abschnitt soll dazu dienen, eine Übersicht über einige heuristische Validierungsmetriken wie sie in diesem Projekt verwendet wurden, zu liefern. Darauf aufbauend werden später die Validierungsergebnisse näher erläutert. Für eine Übersicht über benutzte Ähnlichkeitsmetriken siehe Kapitel 4.1.5.

### 5.1.1. Davies-Bouldin-Metrik

Die Davies-Bouldin-Metrik ist wie in Gleichung 5.1 spezifiziert:

$$DB(U) = \frac{1}{c}\sum_{i=1}^{c}\max_{i\neq j}\{\frac{\Delta(X_i)+\Delta(X_j)}{\delta(X_i,X_j)}\} \tag{5.1}$$

Diese Gleichung berechnet für eine Partition $U$, das heisst eine Menge an $c$ Clustern, ein Gütekriterium wobei $\Delta(X_j)$ bzw. $\Delta(X_i)$ auf die Intraclusterdistanzen der Cluster $j$ bzw. $i$ verweisen und $\delta(X_i,X_j)$ auf die Interclusterdistanz zwischen den beiden Clustern $i$ und $j$ verweist. Für die Berechnung sowohl der Intraclusterdistanzen sowie der Interclusterdistanzen stehen wie in Kapitel 4.1.5 beschrieben mehrere Ähnlichkeitsmetriken zur Verfügung. Das Qualitätsmaß, das mit Hilfe der Gleichung 5.1 berechnet wird, basiert auf Homogenität und Heterogenität der Cluster. Das bedeutet, dass Cluster ein hohes Qualitätsmaß aufweisen sobald die Cluster ein

hohes Maß an Homogenität, d.h. die Intraclusterdistanz muss so gering wie möglich sein, und ein hohes Maß an Heterogenität aufweisen, d.h. die Distanz zwischen den unterschiedlichen Cluster sollte so hoch wie möglich sein. Da in Gleichung 5.1 der maximale Wert des Verhältnisses von Intraclusterdistanz zu Interclusterdistanz ($\frac{\Delta(X_i)+\Delta(X_j)}{\delta(X_i,X_j)}$) ermittelt wird, gilt für die Metrik, dass je niedriger der errechnete Wert ist, desto besser wurde die Einteilung in Cluster durchgeführt. Daraus ergibt sich dass die Parameterkonfiguration, die $DB(U)$ minimiert, als optimal angesehen werden kann.

Für diese Metrik muss allerdings in Betracht gezogen werden, dass mindestens zwei Cluster vorhanden sein müssen, da die Interclusterdistanz in die Berechnung mit einfließt.

### 5.1.2. Silhouette-Metrik

Für die Berechnung der Silhouette-Metrik sind ebenfalls mindestens zwei Cluster notwendig, da hier, genauso wie für die Berechnung der Davis-Bouldin Metrik, die Ähnlichkeit zwischen den Objekten eines Clusters mit allen Objekten aller anderen Cluster einer Partition berechnet werden muss. Die Silhouette-Metrik berechnet für jedes Element *i* eines Clusters *C* ein Gütemaß, das angibt inwiefern eine Zuweisung zu einem bestimmten Cluster gerechtfertigt war. Diese sogenannte *silhouette width* (vgl. Bolshakova u. Azuaje (2003)) wird wie in Gleichung 5.2 berechnet.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{5.2}$$

In Gleichung 5.2 steht $a(i)$ für die durchschnittliche Distanz zwischen dem i-ten Element und allen anderen Elementen eines Clusters und $b(i)$ für die kleinste Durchschnittsdistanz zwischen dem i-ten Element und den Elementen aller anderen Cluster. Die kleinste Durchschnittsdistanz ergibt sich aus dem minimalen Wert aller durchgeführten Interclusterdistanzberechnungen. Der Wertebereich der Silhouette-Metrik geht dabei von [-1,1] wobei dieser folgendermaßen interpretiert wird: Je

mehr ein Silhouettewert sich dem Wert 1 annähert desto besser ist die Qualität der Zuordnung. Ist der Wert bei 0 so hätte der Wert auch einem anderen benachbarten Cluster zugeordnet werden können. Je mehr der Wert sich -1 annähert desto größer ist die Wahrscheinlichkeit, dass das Objekt, für das der Silhouettewert berechnet wurde, einem falschen Cluster zugeordnet worden ist. Um nun die Qualität eines gesamten Clusters zu berechnen und nicht nur die Güte einzelner Objekte eines Clusters bietet sich Gleichung 5.3 an.

$$C_i = \frac{1}{m} \sum_{i=1}^{m} s(i) \tag{5.3}$$

In dieser Gleichung steht die Variable $m$ für die Anzahl der Objekte im Cluster $C_i$ und $s(i)$ für die Silhouettewerte der einzelnen Objekte. Für eine Clusterpartition, das heisst für eine Menge an Clustern einer Analyse, kann desweiteren ein globaler Silhouettewert berechnet werden. Die Gleichung hierfür ist in 5.4 gegeben.

$$GC_i = \frac{1}{c} \sum_{i=1}^{c} C_i \tag{5.4}$$

Wie in Gleichung 5.4 ersichtlich berechnet sich dieser globale Silhouettewert als Durchschnittswert der Silhouettewerte aller Cluster einer Analyse. Dieser Wert kann als Anhaltspunkt für die Qualität einer kompletten Clusteranalyse dienen.

### 5.1.3. Evaluierung mit Hilfe von Bibliosphere

Ein weiterer Ansatz zur Evaluierung von geclusterten genetischen Daten speziell bei diesem Vorhaben ist die Verwendung von Bilbiosphere der Firma Genomatix Software GmbH bzw. der Vergleich zwischen den Clustern, die Bibliosphere an Hand von Text-Mining-Verfahren errechnet, und den mit Hilfe des SOM- oder k-means-Algorithmus berechneten Clustern. Anders als die vorhergegangenen Metriken, ist dieser Ansatz domänenspezifisch und hängt daher von dem den Daten zu Grunde liegenden Gegebenheiten ab. Das heisst in diesem Fall, dass das Clustering

von Genen auf Basis von bereits validiertem Wissen über die Zusammenhänge und Beziehungen zwischen Genen validiert wird.

Wie bereits in Kapitel 2.3.2 erwähnt ist Bibliosphere eine Applikation der Firma Genomatix Software GmbH und wird von etlichen namhaften Forschungsinstituten verwendet, darunter das Max Planck-Institut, das amerikanische *National Institute of Health* und die Universität Regensburg. Bibliosphere dient zur Ermittlung von Signalpfaden, Gen-Gen-Interaktionen, etc. Diese Informationen werden an Hand von Verfahren des Text-Mining in wissenschaftlichen Publikationen ermittelt. Mit Hilfe dieses Softwaretools ist es möglich sich die Gen-Gen-Interaktionen, die in Form von Clustern visualisiert werden, zu exportieren. Diese exportierten Daten dienen wiederum als Input für eine mögliche Clustervalidierung.

Die Evaluierung der Clusteringergebnisse an Hand von Bibliosphere beruht auf der Annahme, dass Gene, für die von Bibliosphere eine Verbindung ermittelt wurde, ebenfalls in einem gemeinsamen Cluster auftreten sollten. Daher dient eine Datei, in der die Genverbindungen gespeichert wurden, als Input für diese Art der Validierung. Im ersten Schritt wird diese Inputdatei geparst und die darin enthaltenen Informationen in einer eigenen Datenstruktur gespeichert, die es ermöglicht leicht zu ermitteln, welche Gene gemeinsam in einem Cluster auftreten. Daraufhin wird für alle Cluster einer Analyse überprüft, wieviele Gene gemeinsam in demselben Cluster im Vergleich zu den Bibliospheredaten auftreten. Das Gütemaß wird dann folgendermaßen berechnet:

$$B(c) = \frac{(x_c * 100)}{\mid c \mid} \tag{5.5}$$

Hierbei steht $B(c)$ für die Qualität des Clusters $c$, $x_c$ für die Anzahl der Übereinstimmungen zwischen den Bibliospheredaten und den Genen des Clusters $c$ und $\mid c \mid$ für die Anzahl von Objekten im Cluster $c$. Logischerweise ist $B(c)$ ein Wert aus dem Intervall [0,1] und je näher $B(c)$ sich dem Wert 1 nähert, desto besser ist die Qualität des Clusterings, da mehrere Übereinstimmungen zwischen den Bibliospheredaten

und der entsprechenden Clusteranalyse mit Hilfe von SOM oder k-means gefunden wurden.

### 5.1.4. Evaluierung mit Hilfe von Resampling

Ein weiterer Schritt in der Evaluierung von Clusteringergebnissen ist die Überprüfung der Stabilität des Algorithmus. Das heisst, dass nicht nur die geclusterten Daten auf die richtige Zuordnung zu Clustern hin überprüft werden, sondern der Algorithmus per se einer Überprüfung unterzogen wird. Denn nur wenn ein Algorithmus die Daten stabil in Cluster einteilt, das heisst, dass die Unterschiede zwischen einzelnen Clusteringläufen gering sind, kann man von einem ausgereiften und benutzbaren Algorithmus sprechen. Auch für zukünftige Clusteringanalysen ist es wichtig, dass die Ergebnisse nicht auf „sample artifact and fluctuation“ (siehe Levine u. Domany (2001)) beruhen, sondern auf einem stabilen und zuverlässigen Clusteringprozess, um stabile und zuverlässige Ergebnisse zu erhalten.

Deswegen ist ein weiterer Teil der Evaluierungsphase die Validierung der Klassifizierungsergebnisse beruhend auf einer sogenannten Resampling -Methode (Levine u. Domany (2001)). Auch wenn andere Clustervalidierungsmetriken, wie etwa die Davies-Bouldin-Metrik die Qualität eines Clusteringalgorithmus basierend auf einem statistischen Ansatz berechnen, ist es dennoch schwer, an Hand von rein statistischen Metriken Aussagen darüber treffen zu können, welcher Clusteringalgorithmus bzw. welche Konfiguration zuverlässiger und stabiler ist als eine andere. Daher wird im folgenden Validierungsalgorithmus eine Ausgangsmenge von $M$ Clustern in $x$ Submengen $S$ aufgeteilt, so dass gilt

$$S_x = \sum_{i=0}^{n} x \in M \tag{5.6}$$

Das bedeutet, dass jeder Submenge $S$ eine bestimmte Anzahl an zufällig ausgewählten geclusterten Objekte zugewiesen wird. Diese Submengen werden dann wiederum mit derselben Konfiguration, die für die Gesamtmenge $M$ verwendet wurde,

geclustert. So kann das Risiko reduziert werden, dass die Ergebnisse des Clusterings der Menge $M$ nicht auf etwaigem *noise* in den Ausgangsdaten beruhen. Die Möglichkeit diesen Aspekt des *noise* ausschließen zu können ist von enormer Wichtigkeit, da man, vor allem bei großen und komplexen Datenmengen, nicht immer zu 100% ausschließen kann, dass die Daten nicht etwa auf Grund von Messfehlern oder ähnlichem verzerrt wurden. Solche Datenmängel verzerren unter Umständen wiederum das Ergebnis des Clusterings, was in der Folge zu falschen Annahmen über die Ausgangsdaten führen kann.

Nachdem nun der Clusteringalgorithmus mit einer bestimmten Parameterkonfiguration auf alle Submengen $S_x$ angewendet wurde, wird überprüft, wieviele Objekte in denselben Clustern auftreten in Relation zur Ausgangsklassifizierung. Je höher der dabei erhaltene Wert, desto stabiler und zuverlässiger kann das Clusteringergebnis angesehen werden. Der Vorteil dieser Validierungsmetrik sowie der Evaluierung mit Hilfe von Bibliospheredaten im Vergleich zu rein heuristischen Validierungsmetriken ist der, dass bei diesem Ansatz die zu Grunde liegenden Daten selbst und die daraus entstandenen Cluster untersucht werden und nicht nur die durch das Clustering erhaltenen Ergebnisse.

Das letztendliche Validierungsmaß ist in folgender Gleichung gegeben:

$$H = \langle\langle c\rangle x\rangle \tag{5.7}$$

In Gleichung 5.7 steht die Variable $H$ für das Qualitätsmaß, das mittels Resampling ermittelt wurde, $\langle c\rangle$ für die durchschnittliche Verteilung der koinzidierenden Objekte wobei 0 Übereinstimmungen ignoriert werden und $x$ ist der Durchschnitt von $c$ über eine gesamte Clusteranalyse verteilt. Auf diese Weise werden sowohl die Übereinstimmungen der Cluster untereinander als auch clusterübergreifend berücksichtigt.

## 5.2. Aufbau und Ergebnisse der Clusterevaluierung

Nachdem die für diese Evaluierung ausgewählten Metriken und Algorithmen erläutert wurden, wird im Folgenden die Evaluierung selbst näher beschrieben. Die Auswahl der Metriken und Algorithmen beruht dabei zum einen auf dem weit verbreiteten Einsatz der Davies-Bouldin-Metrik und des Silhouette-Maßes sowohl in bioinformatischen Clusteringanalysen als auch in anderen Fachbereichen, in denen Clustering durchgeführt wird. Die Evaluierung der Clusteringergebnisse an Hand domänenspezifischen Wissens kommt zum Einsatz, da Expertenwissen über eine Domäne ein wichtiger Bestandteil für eine zuverlässige Evaluierung ist. Da allerdings die Datenmengen oftmals zu groß sind, um sie manuell von mehreren Fachexperten evaluieren zu lassen, wurde hier auf genetisches Fachwissen, das mit Hilfe von Bibliosphere ermittelt wurde, als Vergleichsgröße herangezogen. Um nicht nur die Qualität der Ergebnisse zu untersuchen, sondern auch den gesamten Algorithmus und seine Implementierung auf seine Qualität hin zu überprüfen, wurde die oben beschriebene Resamplingmethode eingesetzt, deren Ergebnisse zusammen mit den Ergebnissen der anderen Evaluierungsmetriken im Folgenden dargelegt werden.

Im folgenden Abschnitt soll der Aufbau der Evaluierung näher beschrieben werden, bevor die eigentlichen Evaluierungsergebnisse erläutert und ausgewertet werden. Für die Evaluierung des Clusterings sowohl auf Basis des SOM- als auch des k-means Algorithmus wurden 450 zufällig ausgewählte Gene aus der Datenbank exportiert. Die Anzahl der ausgewählten Gene beruht dabei auf der Annahme, dass für eine manuell durchgeführt Signalpfadanalyse, deren Daten als Ausgangspunkt für das Clustering dienen, nicht mehr als 300 Gene zum Einsatz kommen. Basierend auf der Liste mit 450 Genen wurden mit Hilfe des Ingenuity Pathway Browsers die entsprechenden genetischen Signalpfade ermittelt und als Input für das Informationsranking der Gene verwendet. Für das Ranking der Gene bzw. der funktionellen Relevanz von Genen wurden die Genfunktionen auf den Bereich für metabolische Prozesse eingeschränkt. Für die Bereinigung von SNPs mit Hilfe der

Software PLink 2.3.1 wurde als Schwellenwert für die LD-Bereinigung der Wert 0,7 gewählt. Als Fenstergröße wurde 50 und als Schrittgröße 5 gewählt. Der für den SOM spezifische Wert des initialen alpha-Wertes lag für die Evaluierung bei 0.999 und der anfängliche Radius um die *best-matching unit* innerhalb der Kohonenkarte lag bei $d(x_0, x_k)/1.15$, wobei $d(x_0, x_k)$ die Distanz zwischen dem ersten und letzten Element der Karte nach einer Sortierung angibt. Ein weiterer SOM spezifischer Parameter - der Faktor für das Anpassen des Radius um die BMU - wurde für diese Evaluierung auf 0.998 gesetzt, um ein möglichst geringes und sanftes Sinken des Radius zu erreichen.

Für die Evaluierung der Stabilität des Clusterings an Hand von Resampling wurden 90 Gene pro Algorithmus und Clusteringdurchlauf aus der ursprünglichen Menge von 450 Genen ausgewählt. So wurden insgesamt pro Clusteringalgorithmus jeweils 90 Gene 30 Mal mit 50.000, 100.000 und 200.000 Iterationen geclustert. Die Ergebnisse dieser Evaluierungsphase finden sich zusammen mit den Ergebnissen der anderen Evaluierungen in den Tabellen 5.1 bzw. 5.2.

Der wichtigste Parameter jedoch war, wie in Tabelle 5.1 ersichtlich die Anzahl der Iterationen, da dieser Parameter direkt das komplette Netzwerk während der Trainingsphase beeinflusst. Zur Überprüfung der Qualität des Algorithmus wurden daher zuerst die Ergebnisse basierend auf 50.000, 100.000 und 200.000 Iterationen gewählt und verglichen. Die Überprüfung der Ergebnisse beruht dabei auf den oben erwähnten Algorithmen. Tabelle 5.1 zeigt eine Übersicht über die Evaluierungsergebnisse des SOM-Algorithmus im Bezug auf die Anzahl der Iterationen.

Wie in Tabelle 5.1 zu sehen ist, nimmt die Anzahl der gefüllten Cluster mit der Anzahl der Iterationen während der Trainingsphase kontinuierlich ab. Um dieses Verhalten beurteilen zu können, muss man jedoch einen Blick auf die Ergebnisse der einzelnen Evaluierungsmetriken werfen. So legt der niedrige Wert der Davies-Bouldin-Metrik für 100.000 Iterationen nahe, dass eine Aufteilung in 10 Cluster ein nahezu optimales Ergebnis darstellt und dass die geringere Anzahl an nicht leeren Clustern bei 200.000 Trainingsiterationen ein suboptimales Ergebnis darstellt, was jedoch wiederum ein besseres Resultat ist als bei 50.000 Trainingsiterationen. Die

| Iterationen | Anzahl nicht leerer Cluster | Globale Silhouette-Metrik | Davies-Bouldin | Bibliosphere | Resampling |
|---|---|---|---|---|---|
| 50.000 | 12 | 0,487 | 0,66% | 58,33% | 75,17 |
| 100.000 | 10 | 0,837 | 0,13% | 75,39% | 84,20 |
| 200.000 | 6 | 0,726 | 0,37% | 91,86% | 82,88 |

Tabelle 5.1.: Evaluierungsergebnisse des SOM-Algorithmus

Tatsache, dass das Ergebnis des Clusterings basierend auf 200.000 Iterationen ein schlechteres Ergebnis als bei 100.000 Iterationen liefert ist hierbei vermutlich auf den Effekt des *over-fitting* zurückzuführen. Darunter versteht man, dass das trainierte Modell zu stark trainiert wurde und so nicht mehr in der Lage ist, die zu clusternden Daten zu reproduzieren (siehe dazu Weijters u. a. (1997) bzw. Kapitel 4.1.4). Im Falle von 50.000 Iterationen scheint die Clusteraufteilung ein nicht sehr homogenes Ergebnis darzustellen, da auf Grund der hohen Anzahl von 12 Clustern im Vergleich zu den anderen Konfigurationen, Gene fälschlicherweise in unterschiedliche Cluster aufgeteilt wurden. Desweiteren legt dieses Ergebnis nahe, dass obwohl Gene in zu viele Cluster aufgeteilt wurden, diese 12 Cluster untereinander ein geringes Maß an Heterogenität aufweisen, da die Heterogenität den Divisor der Davies-Bouldin-Metrik (siehe Gleichung 5.1) darstellt.

Ein ähnliches Bild liefern die Werte der Silhouette-Metrik. Im Falle der Werte der globalen Silhouette-Metrik liefert ebenfalls der Clusteringlauf mit 100.000 vorangegangenen Trainingsiterationen mit 0,837 das beste Ergebnis im Vergleich zu den anderen Ergebnissen. Eine weitere Übereinstimmung mit der Davies-Bouldin-Metrik ist, dass 200.000 Trainingsdurchläufe bzw. 6 nicht leere Cluster das zweit beste Ergebnis (0,726) liefern und 12 gefüllte Cluster wiederum den schlechtesten Validierungswert (0,487) ergeben. Genauso wie bei der Berechnung der Davies-Bouldin-Metrik legt dieses Ergebnis den Schluss nahe, dass die Trainingsphase mit

| Iterationen | Anzahl nicht leerer Cluster | Globale Silhouette-Metrik | Davies-Bouldin | Bibliosphere | Resampling |
|---|---|---|---|---|---|
| 50.000 | 7 | 0,66 | 1,018% | 50,793% | 69,67 |
| 100.000 | 10 | 0,74 | 0,864% | 41,865% | 73,876 |
| 200.000 | 7 | 0,445 | 1,298% | 41,468% | 68,83 |

Tabelle 5.2.: Evaluierungsergebnisse des k-means Algorithmus

100.000 Iterationen ein nahezu optimales Ergebnis in Hinblick auf Heterogenität und Homogenität der Cluster liefern.

Die Ergebnisse der Validierung mittels Resampling unterstreichen die oben erwähnten Ergebnisse. Auch hier liefert die Clusteranalyse auf der Basis von 100.000 Iterationen den besten Wert (84,2) und das Clustering mit 50.000 Iterationen mit 75,17 das schlechteste Ergebnis. Dieses Ergebnis lässt wiederum den Schluss zu, dass das Clustering mit Hilfe des SOM-Algorithmus ein sehr stabiles, wenn auch nicht perfektes, Ergebnis liefert und nicht anfällig für eventuellen *noise* ist.

Anders verhält es sich im Hinblick auf den Vergleich zwischen Clustern, die mit Hilfe des SOM-Algorithmus berechnet wurden, und Clustern, die mit Hilfe der Bibliosphere-Software automatisch erstellt wurden. Bei dieser Validierung steigt die Qualität der Cluster direkt proportional zur Anzahl der Trainingsschritte. So erhält das SOM-Clustering mit 50.000 Trainingsschritten lediglich eine Übereinstimmung von ca. 58%, während 200.000 Iterationen zu einer Übereinstimmung von ca. 92% führen.

Um eine Vergleichsgröße zum Clustering mit Hilfe des SOM-Algorithmus zu haben, wurde ebenfalls das Clustering mittels k-means näher validiert. Dazu wurde der k-means Algorithmus ebenfalls mit 50.000, 100.000 und 200.000 Iterationen durchgeführt mit jeweils 100 Ausgangsclustern. Tabelle 5.2 zeigt die Ergebnisse mit den verschiedenen Parametern.

Was bei den Clusteringdurchläufen auf Basis des k-means Algorithmus ins Auge sticht ist, dass, wie im Falle des SOM-Algorithmus, das Clustering mit 100.000 Iterationen die besten Ergebnisse im Bezug auf Silhouettewert und Davies-Bouldin-Metrik liefert. Im Hinblick auf den Vergleich mit Bibliosphere-Clustern schneidet das k-means Verfahren allerdings erheblich schlechter ab als SOM, bei dem der schlechteste Wert 58,3% beträgt, wohingegen das k-means Verfahren bestenfalls lediglich 50,793% Übereinstimmung erreicht. Ein ähnliches, jedoch nicht so stark ausgeprägtes, Ergebnis zeigt sich auch im Hinblick auf den Silhouettewert. Auch wenn, wie im Falle von SOM, das Clustering mit 100.000 Iterationen den besten Wert erreicht, so ist dieser jedoch nur wenig besser als der zweit beste Wert beim Clustering mit SOM. Dieses Bild spiegeln auch die Durchschnittswerte der Silhouette-Metrik wider. Während der durchschnittliche Silhouettewert beim SOM-Algorithmus rund 0,683 beträgt, so liegt er für das k-means Clustering lediglich bei 0,615. Ähnlich verhält es sich bei der Davies-Bouldin-Metrik. Obwohl hier auch 100.000 Iterationen das beste, sprich niedrigste, Ergebnis liefern, so sind doch die Unterschiede zwischen den beiden Clusteringalgorithmen doch frappierend. Während der SOM im Durchschnitt einen DB-Wert von 0,387 erhält, was ein im Durchschnitt nahezu optimales Ergebnis darstellt, so hat das k-means Verfahren einen durchschnittlichen DB-Wert von 1,06, was einer Differenz zwischen beiden von 0,673 entspricht. Auch der Vergleich mit Bibliosphere-Clustern unterstreicht die negativen Ergebnisse der Clusteringqualität mittels k-means. Während der SOM-Algorithmus eine durchschnittliche Übereinstimmung von 75,2% aufweist, erreicht das k-means Clustering lediglich einen durchschnittlichen Wert von 44,7%. Nichtsdestotrotz zeigt sich im Vergleich der beiden Algorithmen, dass eine Aufteilung der Daten in 10 Cluster für diese Daten das beste Ergebnis liefern.

Auch die Ergebnisse des Resamplings lassen den Schluss zu, dass der k-means Algorithmus dem SOM-Algorithmus deutlich schlechter abschneidet, da wie bei den anderen Validierungsmetriken das beste Ergebnis (73,876) bedeutend geringer ist als das schlechteste Ergebnis der Resamplevalidierung des SOM-Algorithmus. Dies liegt allerdings in der Natur des k-means Algorithmus, da, wie bereits in Kapi-

tel 4.1.1 erwähnt, der Algorithmus stark anfällig für Schwankungen ist, die auf der zufälligen Auswahl und Verteilung von Clusteringobjekten beruhen. Insofern unterstreicht dieses Ergebnis die grundlegende Annahme bzgl. der Unzulänglichkeit des k-means Algorithmus im Vergleich zu anderen Clusteringalgorithmen wie sie auch Gat-Viks u. a. (2003) festgestellt haben.

Vergleicht man Clusteringanalysen auf der Basis von Expressionsdaten (vgl. Bolshakova u. Azuaje (2003)) und das Clustering mit Hilfe eines diversifizierteren Gewichtsvektors, das heisst mit einem Gewichtsvektor, der nicht nur die numerischen Ergebnisse einer Genexpressionsanalyse beinhaltet sondern wie in Kapitel 4 beschrieben mehrere sowohl rein bioinformatische als auch Metadaten beinhaltet, so führt letzteres zu besseren Ergebnissen. So erreichten Bolshakova u. Azuaje (2003) etwa einen durchschnittlichen DB-Wert von 2,53 im Falle von Clustering von Leukämiedaten und einen durchschnittlichen Silhouettewert 0,43. Dies lässt wiederum den Schluss zu, dass ein breiter Gewichtsvektor mit vielen unterschiedlichen Faktoren einen besseren Ausgangspunkt für das Clustering genetischer Daten liefert als das Clustering rein auf der Basis von Genexpressionsdaten.

Auch wenn die hier vorliegenden Ergebnisse vielversprechend sind, muss dennoch erwähnt werden, dass aus finanziellen Gründen eine weiterführende Analyse der Ergebnisse und der Auswirkungen der einzelnen Faktoren der Clusteringsoftware nicht möglich war. So bleibt noch offen wie sich die Clusteringalgorithmen, also SOM und k-means, bei veränderten Parametern wie zum Beispiel dem Dämpfungsfaktor des PageRank-Algorithmus verhalten würden. Ebenfalls sollte noch evaluiert werden, wie der k-means Algorithmus mit unterschiedlichen Anzahlen von Ausgangsclustern abschneidet. Ein weiterer interessanter Evaluierungsansatz wäre darüber hinaus wie sich unterschiedliche Nachbarschaftsfunktionen im SOM-Algorithmus auf das Ergebnis auswirken. Darüber hinaus sollte festgehalten werden, dass „when combining multiple data types, establishing a convincing evaluation framework is a tedious task. [...] labor-intensive biological evaluations are required and usually have to start from educated guesses on good cluster parameterizations.“ (Glenisson u. a. (2003)). Nichtsdestotrotz wurde hier ein Grundstein

gelegt für weitere Untersuchungen im Bereich des Clusterings genetischer Massendaten auf der Basis anderer Faktoren als den weit verbreiteten Expressionsdaten.

# Kapitel 6.

# Fazit

Auch wenn das „Human Genome Project“ im Jahr 2003 das selbst gesteckte Ziel der Erforschung und Entschlüsselung des menschlichen Genoms nach 13 Jahren intensiver Forschung erreicht hatte, so steckt die Analyse und Auswertung genetischer Daten immer noch in den Anfängen. Trotz oder gerade wegen der immensen Mengen genetischer Daten, die weltweit in Datenbanken zur Verfügung stehen und mit Hilfe neuester Technologien wie etwa den 500K-SNP-Chips generiert werden, ist es von großer Wichtigkeit Algorithmen und Verfahren zu entwickeln, um mit informationstechnologischer Unterstützung eine Hilfestellung bei der Analyse genetischer Massendaten und der damit verknüpften medizinischen Forschung zu bieten. Jedoch sollte bei so weitgesteckten Zielen wie der Entschlüsselung und der kompletten Analyse des menschlichen Genoms nicht die Grundlagenarbeit wie zum Beispiel die Erhebung und statistische Analyse von Patientendaten ausser Acht gelassen werden, da die Schlüsse genetischer Forschung direkt auf dieser Grundlagenarbeit beruhen.

Daher war es das Ziel der vorliegenden Arbeit ein Framework zu erstellen, das zum einen die Arbeit im Zusammenhang mit klinischen Studien, das heisst die Datenspeicherung und -verwaltung, vereinheitlicht und standardisiert und zum anderen Auswertungen genetischer Daten an Hand von Clusteringanalysen ermöglicht. Dazu wurde mit Hilfe des Java Development Kits 5 ein Softwareframework mit starkem Praxisbezug entwickelt. Während der Entwicklungsphase wurde daher großer Wert auf eine enge Zusammenarbeit und gute Kommunikation mit dem Fachper-

sonal, also den Studienverantwortlichen und den Verantwortlichen für Genanalysen aus den Bereichen Medizin und Biologie, gelegt.

Um die Ziele zu vewirklichen wurde zum einen eine Software entwickelt, die es ermöglicht aus XML-Daten Formulare zu erzeugen, mit deren Hilfe die Daten von Studienteilnehmern in einer zentralen Datenbank gespeichert werden. Zur Erstellung der XML-Daten steht desweiteren ein einfach Hand zu habendes Plugin zur Verfügung, das es auch ungeübteren Benutzern ermöglicht auf einfache Art und Weise mit „drag and drop“ Formulare und qualitätssichernde Kriterien für die Dateneingabe zu erstellen. Die Formulare bieten darüber hinaus die Möglichkeit die Daten schon während der Eingabe auf ihre Korrektheit hin zu überprüfen.

Das hier entwickelte Softwareframework stellt einen Ausgangspunkt für weitere Untersuchungen im Bereich von UIMS-Software dar. Vor allem im Bereich klinischer Studien bieten flexible und leicht anpassbare Softwareprodukte einen großen Vorteil im Vergleich zu starren Softwaresystemen, da sich unterschiedliche klinische Studien doch in ihrem Aufbau sehr ähnlich sind und sich nur im jeweiligen Datenbankdesign unterscheiden. Das heisst, dass im Vergleich zu starrer Datenbanksoftware im Falle von GUi4DB lediglich ein Formular erstellt oder lediglich angepasst werden muss. Desweiteren wäre eine genaue Evaluierung des Systems von großem Interesse, um mögliche Schwachstellen und Probleme in Zukunft zu vermeiden.

Neben der Software zur Erstellung von Datenbankformularen auf Basis von XML wurde ebenfalls ein Datenmodell entwickelt, das zum einen reine Gendaten beinhaltet und zum anderen Metainformationen über Gene. Darüber hinaus beinhaltet die erstellte Java-API Möglichkeiten Abfragen an einschlägige Onlinedatenbanken wie etwa Pubmed zu senden und die Ergebnisse auszuwerten. Ausgehend von diesem Datenmodell bzw. der Java-API kann in Zukunft ein weiterführendes System für bioinformatisches bzw. medizinisches Information Retrieval zu erstellen, das einerseits die Auswertung genetischer Daten oder den Erkenntnisgewinn im klinischen Alltag verbessern helfen könnte. Dazu könnte das hier vorgstellte Datenmodell um

weitere Aspekte wie etwa einen medizinischen Thesaurus oder die Klassifizierung von Dokumenten wie es Palakal u. a. (2002) umgesetzt haben erweitert werden.

Zur Durchführung von Clusteringanalysen wurde der „self organizing maps"-Algorithmus von Teuvo Kohonen implementiert. Dabei gilt es zu erwähnen, dass für das Clustering genetischer Daten nicht wie bisher üblich nur Genexpressionsdaten verwendet werden, sondern ein Gewichtsvektor erstellt wird, der mehrere Faktoren verwendet, um so ein möglichst genaues Bild der zu clusternden Gene zu zeichnen. Dazu gehören der Informationsgehalt von „single-nucleotide polymorphisms" und darauf aufbauend das Informationsgewicht von Genen, das mit Hilfe eines angepassten PageRank-Algorithmus berechnet wird, die rein bioinformatische Korrelation zwischen Genotyp und Phänotyp, das Publikationsgewicht eines Gens, der funktionellen Relevanz von Genen und die P-Werte die vorab mit Hilfe anderer Bioinformatikapplikationen berechnet wurden. Wie in Kapitel 5.2 erwähnt wurde, werden mit Hilfe dieses breit gefächerten Clusteringalgorithmus bessere heuristische Evaluationsergebnisse erzielt. Allerdings konnte die Evaluierung der Clusteringergebnisse nicht komplett durchgeführt werden, da der Zugriff auf die zu Grunde liegenden Daten nicht mehr möglich war. Jedoch stimmen die ersten Evaluierungsergebnisse sehr zuversichtlich, dass das Clustering wie es hier durchgeführt wurde Vorteile im Vergleich zum Clustering auf der Basis von Genexpressionsdaten bietet.

Auch wenn die hier vorgelegten Entwicklungen und Ergebnisse zuverlässig stimmen, so muss dennoch erwähnt werden, dass sie als Ausgangspunkt für weitere Arbeiten auf diesem Gebiet gesehen werden sollten. So gilt es beispielsweise noch zu klären inwiefern sich Veränderungen der Parameter für die Clusteringanalyse auf die Ergebnisse auswirken. Desweiteren ist noch unklar wie die Ergebnisse der Clusteringanalysen im Vergleich mit In-vitro-Versuchen abschneiden, was leider aus finanziellen und zeitlichen Gründen im Rahmen dieses Vorhabens nicht mehr untersucht werden konnte. Darüber hinaus wäre sicherlich auch ein Vergleich mit anderen Clusteringapplikation im Bereich Genetik wie etwa PLink (siehe Kapitel 2.3.1) von großem Interesse. Ein weiterer Aspekt für weiterführende Untersuchungen ist der Einsatz anderer Clusteringalgorithmen wie etwa DBSCAN (siehe Kapi-

tel 4.1.3) für mehrdimensionale Gewichtsvektoren im Bereich der Genetik und der daraus resultierende Vergleich zum SOM-Algorithmus.

# Anhang A.

# Abbildungen

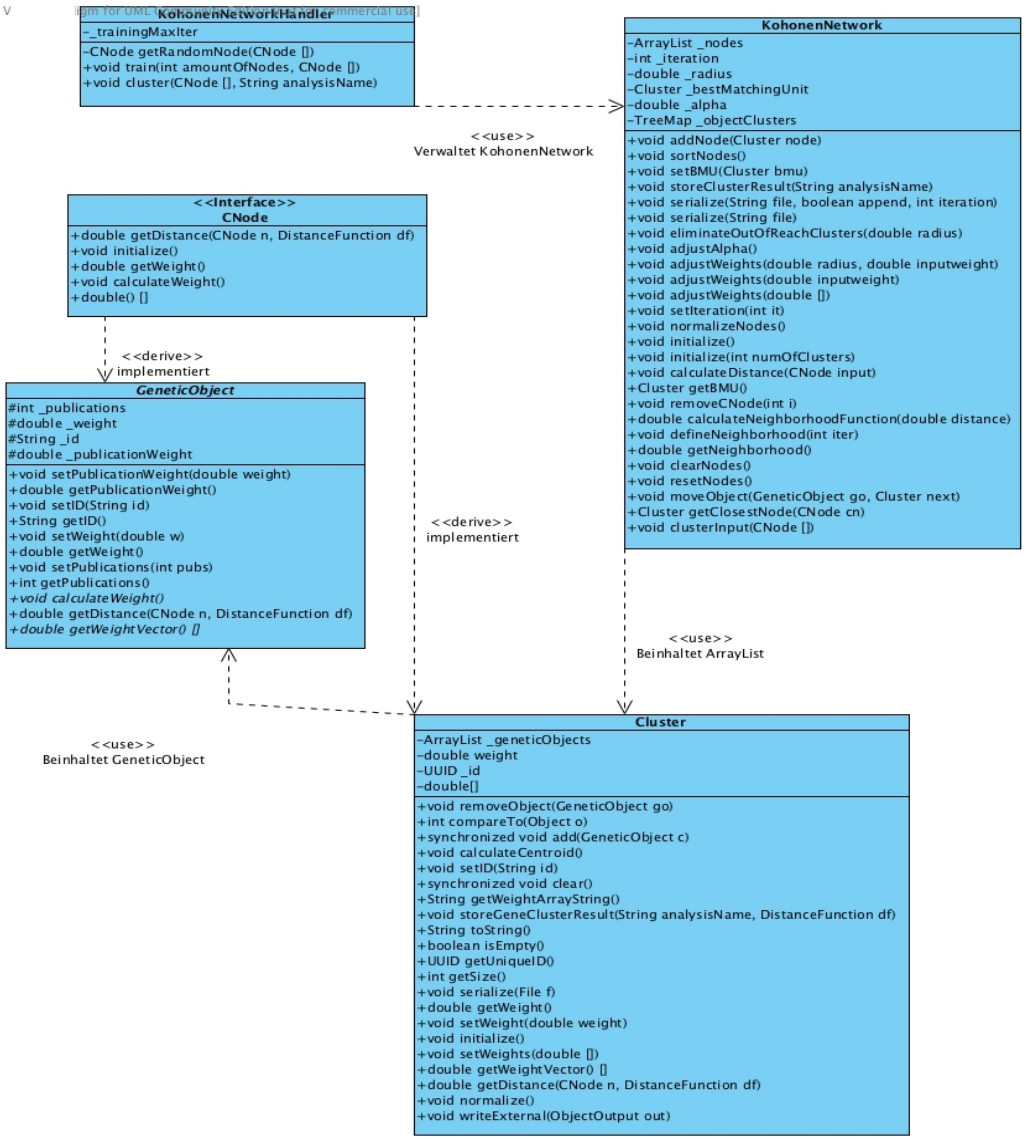



Abbildung A.1.: Klassendiagramm der SOM-Clusteringsoftware

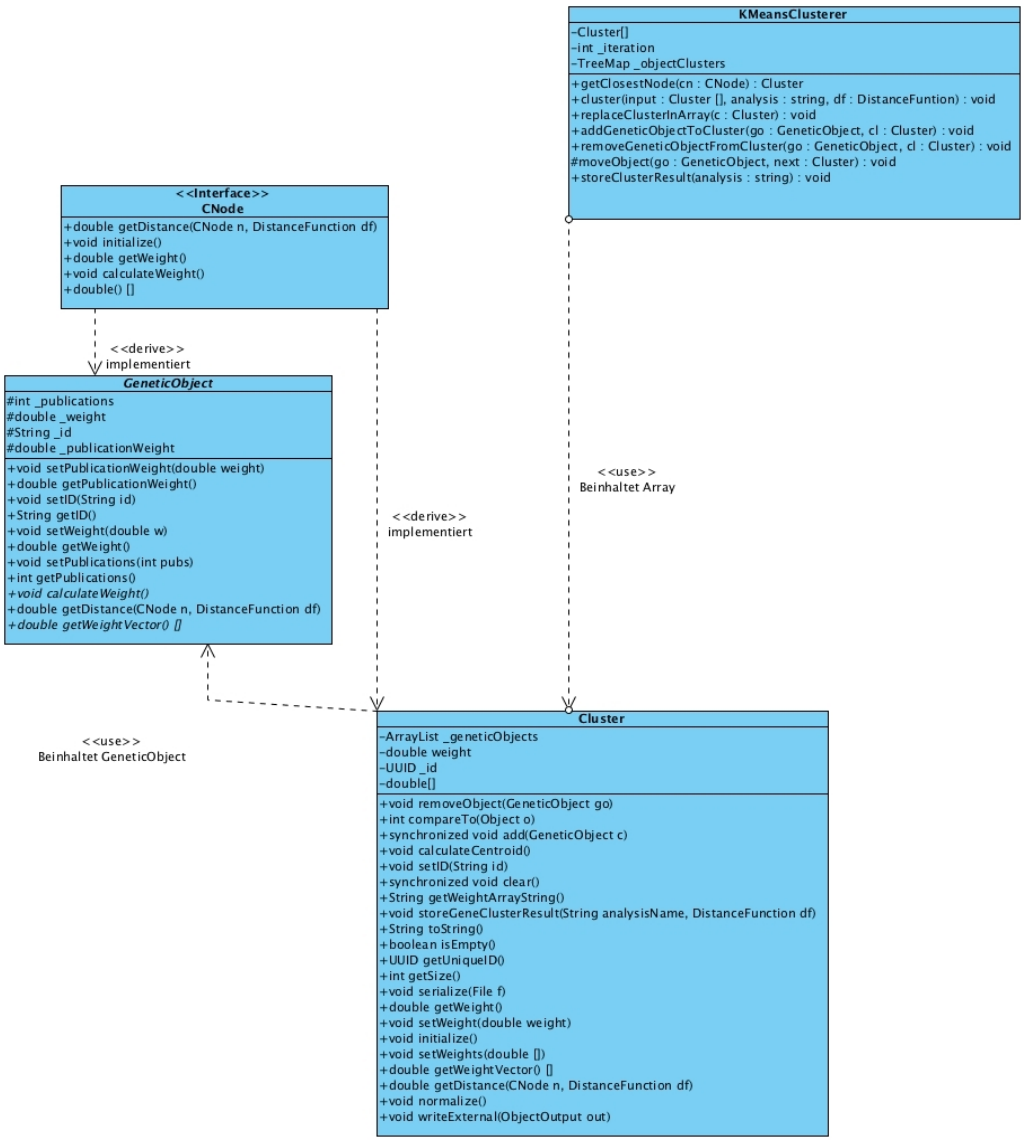



Abbildung A.2.: Klassendiagramm der KMeans-Clusteringsoftware

# Anhang B.

# Code-Beispiele

## B.1. Interface CNode

Listing B.1: Interface CNode

```
1 public double getDistance(CNode n, DistanceFunction df);
2 public void initialize();
3
4 public double getWeight();
5
6 public void calculateWeight();
7
8 public double[] getWeightVector();
```

## B.2. SQL-Abfragen

Listing B.2: Abfrage der Allele eines Patienten auf Chromosom 10

```
1 SELECT t.allele_1, t.allele_2
2 FROM snp_allel_chr_10_kora_pt7 t
3 INNER JOIN s snps_map USING(snp_id)
4 WHERE t.pid='ZZZ000110011' AND s.chromosome=10
5 ORDER BY s.chromosome, s.pos;
```

## B.3. Ermitteln des Genlevels

Listing B.3: Ermittlung des Levels eines Gens innerhalb der GO

```
1   foreach $goid (<FILE>)
2   {
3     chomp($goid);
4     //Ermitteln des Knotens im GO-Graph einer GO-Id
5     my $node = $ontology->nodeFromId($goid);
6     if(defined $node)
7     {
8       //Ermitteln des Pfades zum Knoten/GO-Id
9       my @pathsToRoot = \newline $node->pathsToRoot;
10      // Laenge des Pfades -> Tiefe des Knotens in der
            GeneOntology
11      my $length = \newline @pathsToRoot;
12      if($length > 0)
13      {
14        //Speichern der Daten in der Datenbank
15        writeToDB(\newline"$goid", "$length");
16      }
17    }
18  }
```

## B.4. Code zur Trigger-Generierung

Listing B.4: Erstellung eines Triggers

```
1 CREATE TRIGGER trig_check_ptca_erstiv
2   BEFORE INSERT OR UPDATE
3   ON gokard_erstiv.ptca
4   FOR EACH ROW
5   EXECUTE PROCEDURE gokard_erstiv.check_ptca();
```

## B.5. Stored procedures

Listing B.5: Beispiel einer stored procedure zur Überprüfung von PTCA-Daten

```
1  CREATE OR REPLACE FUNCTION gokard_erstiv.check_ptca()
2    RETURNS "trigger" AS
3  $BODY$
4  DECLARE _var RECORD;
5  BEGIN
6  --Ueberpruefung, der Patenten mit der ID NEW.id vorliegen
7  SELECT * INTO _var FROM gokard_erstiv.vorerkrankungen WHERE
       gokard_erstiv.vorerkrankungen.id=NEW.id;
8  --Falls keine Daten vorliegen, wird eine Fehlermeldung
       generiert, da keine PTCA-Werte gespeichert werden duerfen
       .
9  IF NOT _var.ptca=1 AND NEW.jahr IS NOT NULL THEN
10   RAISE EXCEPTION 'Der␣Patient␣%␣hatte␣nie␣PTCA␣-␣es␣koennen
         ␣keine␣PTCA-Werte␣eintragen␣werden.', NEW.id;
11   RETURN NULL;
12 END IF;
13 IF _var.ptca<>1 THEN
14   RETURN NULL;
15 END IF;
16
17 --Das Datum der PTCA-Erkrankung wird auf Gueltigkeit hin
       ueberprueft
18 IF NEW.jahr > (SELECT date_part('year', now())) OR NEW.jahr
       <= 1899 THEN
19   RAISE EXCEPTION 'Der␣PTCA␣(%)␣muss␣zwischen␣1900␣und␣heute
         ␣stattgefunden␣haben!', NEW.jahr;
20   RETURN NULL;
21 END IF;
```

```
22
23 --Ueberpruefung auf Dummy-Werte; Falls ja wird ein Eintrag
       in der Tabelle fehlende_werte gespeichert
24 IF NEW.jahr=999 OR NEW.jahr=1900 OR NEW.jahr IS NULL THEN
25   IF NEW.jahr=1900 OR NEW.jahr=999 THEN
26     INSERT INTO gokard_erstiv.fehlende_werte(id, tabelle,
           feld, unbekannt) VALUES(NEW.id, TG_TABLE_SCHEMA||'.'
           ||TG_TABLE_NAME, 'jahr', true);
27   ELSE
28     INSERT INTO gokard_erstiv.fehlende_werte(id, tabelle,
           feld, unbekannt) VALUES(NEW.id, TG_TABLE_SCHEMA||'.'
           ||TG_TABLE_NAME, 'jahr', false);
29   END IF;
30   NEW.jahr = NULL;
31 END IF;
32 IF NEW.kh = '999' OR NEW.kh IS NULL OR char_length(NEW.kh) <
       2 THEN
33   IF NEW.kh='999' THEN
34     INSERT INTO gokard_erstiv.fehlende_werte(id, tabelle,
           feld, unbekannt) VALUES(NEW.id, TG_TABLE_SCHEMA||'.'
           ||TG_TABLE_NAME, 'kh', true);
35   ELSE
36     INSERT INTO gokard_erstiv.fehlende_werte(id, tabelle,
           feld, unbekannt) VALUES(NEW.id, TG_TABLE_SCHEMA||'.'
           ||TG_TABLE_NAME, 'kh', false);
37   END IF;
38   NEW.kh = NULL;
39 END IF;
40 IF NEW.ort = '999' OR NEW.ort IS NULL OR char_length(NEW.ort
       ) < 2 THEN
41   IF NEW.ort='999' THEN
```

```
42      INSERT INTO gokard_erstiv.fehlende_werte(id, tabelle,
            feld, unbekannt) VALUES(NEW.id, TG_TABLE_SCHEMA||'.'
            ||TG_TABLE_NAME, 'ort', true);
43    ELSE
44      INSERT INTO gokard_erstiv.fehlende_werte(id, tabelle,
            feld, unbekannt) VALUES(NEW.id, TG_TABLE_SCHEMA||'.'
            ||TG_TABLE_NAME, 'ort', false);
45    END IF;
46    NEW.ort = NULL;
47  END IF;
48  RETURN NEW;
49  END;$BODY$
50    LANGUAGE 'plpgsql' VOLATILE;
```

Listing B.6: Stored procedure zum Loggen von Benutzeraktionen

```
1  CREATE OR REPLACE FUNCTION gokard_erstiv.
       maintain_history_ptca()
2    RETURNS "trigger" AS
3  $BODY$
4  DECLARE _var VARCHAR;
5  DECLARE _query VARCHAR;
6  DECLARE valid BOOL;
7  DECLARE act VARCHAR;
8  DECLARE tab VARCHAR;
9  DECLARE field VARCHAR;
10 BEGIN
11  valid = false;
12 --Ermitteln der Art der Datenbankoperation
13 --Falls Daten geloescht werden sollen wird die Variable act
       dementsprechend gesetzt
14  IF(TG_OP = 'DELETE') THEN
15    act = 'DELETION␣of␣row␣with␣id:␣' || OLD.id;
```

```
16 --Falls es sich um einen Update handelt, wird ermittelt
       welche Daten sich geaendert haben und dementsprechend in
       der Tabelle history ein Eintrag vorgenommen
17 ELSIF(TG_OP = 'UPDATE') THEN
18 IF NEW.id<>OLD.id THEN
19   field = 'id␣from:␣' || OLD.id || '␣TO␣' || NEW.id;
20 ELSIF NEW.jahr<>OLD.jahr THEN
21   valid = (NEW.jahr>1900) AND (date_part('year', now()) >
       NEW.jahr) AND NEW.jahr<>999 AND NEW.jahr IS NOT NULL;
22   _var = 'jahr';
23   field = 'field␣jahr␣from:␣' || OLD.jahr || '␣TO␣' || NEW.
       jahr;
24 ELSIF NEW.kh<>OLD.kh THEN
25   valid = char_length(NEW.kh) > 2 AND NEW.kh IS NOT NULL AND
        NOT NEW.kh='999';
26   _var = 'kh';
27   field = 'field␣kh␣from:␣' || OLD.kh || '␣TO␣' || NEW.kh;
28 ELSIF NEW.ort<>OLD.ort THEN
29   valid = char_length(NEW.ort) > 2 AND NEW.ort IS NOT NULL
       AND NEW.ort='999';
30   _var = 'ort';
31   field = 'field␣ort␣from:␣' || OLD.ort || '␣TO␣' || NEW.ort
       ;
32 ELSIF NEW.aendat<>OLD.aendat THEN
33   field = 'field␣aendat␣from:␣' || OLD.aendat || '␣TO␣' ||
       NEW.aendat;
34 ELSIF NEW.aennam<>OLD.aennam THEN
35   field = 'field␣aennam␣from:␣' || OLD.aennam || '␣TO␣' ||
       NEW.aennam;
36 END IF;
37   act = 'UPDATE␣OF␣' || field || '␣with␣id:␣' || OLD.id;
```

```
38   --Falls es sich um gueltige Werte handelt werden - falls
       vorhanden - die Eintraege in der Tabelle fehlende_werte
       geloescht.
39   IF valid = true THEN
40     _query := 'DELETE FROM gokard_erstiv.fehlende_werte
         WHERE tabelle=''||TG_TABLE_SCHEMA||'.'||
         TG_TABLE_NAME||''' AND feld='||quote_literal(_var);
41     EXECUTE _query;
42   END IF;
43 END IF;
44 tab = TG_TABLE_SCHEMA || '.' || TG_TABLE_NAME;
45 INSERT INTO history(aennam, action, table_name) VALUES(
     current_user, act, tab);
46 RETURN NULL;
47 END;$BODY$
48   LANGUAGE 'plpgsql' VOLATILE;
```

## B.6. Java-Methoden zur Ermittlung von Metadaten

- String getCommentForField(MetaField field, MetaTable table) - Ermittelt den Kommentar, der mit einem Feld assoziiert wurde.
- boolean fieldExists (MetaTable table, MetaField field) - Überprüft ob ein Feld in einer Tabelle existiert.
- String getDataTypeOfField(MetaField field, MetaTable table) - Ermittelt den Datentyp eines Feldes in einer Tabelle.
- boolean allowsNullValues(MetaField field, MetaTable table) - Überprüft ob Null-Werte zulässig sind in einem Feld in einer Tabelle.
- int getCharacterLength(MetaField field, MetaTable table) - Ermittelt die zulässige Länge eines Feldes mit einem Text-Datentypen.

- float getNumericPrecision(MetaField field , MetaTable table ) - Ermittelt die numerische Präzision eines Feldes.
- ArrayList<MetaTableConstraint> getConstraintsForTable (MetaTable table ) - Ermittelt alle „constraints“, die mit einer Tabelle assoziiert sind.
- ArrayList<MetaField> getFieldsForTable (MetaTable table ) - Ermittelt alle Felder innerhalb einer Tabelle.
- ArrayList<MetaTable> getTables ( String schema) - Liest alle Tabellen innerhalb eines Schemas.
- Vector<String> getTableNamesOfSchema(String schema) - Liest die Namen aller Tabellen innerhalb eines Schemas.
- boolean hasUniqueConstraint(MetaTable table , MetaField field ) - Überprüft ob ein Feld in einer Tabelle nur eindeutige Werte beinhalten darf.
- String getCommentForTableConstraint(MetaTableConstraint con, MetaTable table ) - Liest den Kommentar für eine Tabellenbedingung.
- String getCommentForTable(MetaTable table) - Liest den Kommentar einer Tabelle.
- String getReferenceOfTableConstraint ( MetaTableConstraint con, MetaTable table ) - Ermittelt die Tabelle, die von einer bestimmten Bedingung referenziert wird.
- ArrayList<MetaField> getFieldsOfTableConstraint ( MetaTableConstraint con) - Liest die Felder, die in einer Tabellenbedingung beinhaltet sind.
- String getTableSize ( String schema, String table ) - Liest den Speicherplatz einer Tabelle.

- String getDatabaseSize () - Ermittelt den kompletten benötigten Speicherplatz der Datenbank.
- String getCurrentDatabase () - Ermittelt den Namen der Datenbank, mit der der Benutzer gerade verbunden ist.

## B.7. XML-Schema für GUI4DB

Listing B.7: XML-Schema zur Benutzerschnittstellenbeschreibung

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3    elementFormDefault="qualified">
4
5    <!-- Elements -->
6    <!-- Definition des XML-Elements "table", das ein Attribut
         "field" vom Datentyp String beinhalten muss -->
7    <xs:element name="table">
8      <xs:complexType>
9        <xs:simpleContent>
10         <xs:extension base="xs:string">
11           <xs:attribute name="field" type="xs:string" />
12         </xs:extension>
13       </xs:simpleContent>
14     </xs:complexType>
15   </xs:element>
16
17   <!-- Definition einer Oberklasse "uitype", von der alle
        einzelnen GUI-Elemente erben -->
18   <xs:complexType name="uitype">
19     <!-- Jede GUI-Beschreibung muss mindestens Element
          dieses oder eines Untertyps beinhalten -->
20     <xs:sequence>
21       <xs:element minOccurs="1" maxOccurs="unbounded" ref="
           table" />
22     </xs:sequence>
23     <xs:attribute name="name" type="xs:string" use="required
         " />
```

```
24      <xs:attribute name="data_type" type="xs:string" use="
            required" />
25      <xs:attribute name="is_multiple" type="xs:boolean" />
26      <xs:attribute name="check_unique" type="xs:boolean" />
27      <xs:attribute name="prepared_query" type="xs:string" />
28    </xs:complexType>
29
30    <!-- Eine Spezialisierung des Typs "uitype", das Textfeld-
          spezifische Attribute beinhaltet -->
31    <xs:complexType name="textfieldType">
32      <xs:complexContent>
33        <xs:extension base="uitype">
34          <xs:attribute name="max_value" type="xs:double" />
35          <xs:attribute name="min_value" type="xs:double" />
36          <xs:attribute name="columns" type="
                xs:positiveInteger"
37            use="required" />
38                <xs:attribute name="string_length" type="
                      xs:positiveInteger" />
39          <xs:attribute name="input_required" type="xs:boolean
                "
40            default="false" />
41        </xs:extension>
42      </xs:complexContent>
43    </xs:complexType>
44
45    <!-- Eine Erweiterung des Obertyps uitype um Combobox-
          spezifische Parameter -->
46    <xs:complexType name="comboboxType">
47      <xs:complexContent>
48        <xs:extension base="uitype">
```

```
49          <xs:sequence minOccurs="1" maxOccurs="unbounded">
50            <xs:element name="item" minOccurs="1"
51              maxOccurs="unbounded" type="xs:string" />
52          </xs:sequence>
53        </xs:extension>
54      </xs:complexContent>
55    </xs:complexType>
56
57    <!--Definition eines "parameterTypes", der dazu dient
        Parameter in Form von Schluessel-Wert-Paaren an Plugins
         zu uebergeben -->
58    <xs:complexType name="parameterType">
59      <xs:attribute name="name" type="xs:string" use="required
          " />
60      <xs:attribute name="value" type="xs:string" use="
          required" />
61    </xs:complexType>
62
63    <xs:element name="textfield" type="textfieldType"></
        xs:element>
64    <xs:element name="combobox" type="comboboxType"></
        xs:element>
65
66    <!-- Definition eines Plugin-Elements zum Laden externer
        Erweiterungen -->
67    <xs:element name="plugin">
68      <xs:complexType>
69        <xs:sequence minOccurs="1" maxOccurs="unbounded">
70          <xs:element name="parameter" minOccurs="0"
71              maxOccurs="unbounded" type="parameterType" />
72        </xs:sequence>
```

```
73      <xs:attribute name="plugin_name" type="xs:string"
74        use="required" />
75      <xs:attribute name="entry_point" type="xs:string"
76        use="required" />
77      <xs:attribute name="tooltip" type="xs:string" />
78    </xs:complexType>
79  </xs:element>
80
81  <!-- Definition eines Panel-Elements, das einem Reiter in
        der fertigen GUI entspricht und eines der oben
        definierten Elemente beinhalten muss. -->
82  <xs:element name="panel">
83    <xs:complexType>
84      <xs:sequence minOccurs="1" maxOccurs="unbounded">
85        <xs:choice>
86          <xs:element ref="combobox" minOccurs="0"
87            maxOccurs="unbounded" />
88          <xs:element ref="textfield" minOccurs="1"
89            maxOccurs="unbounded" />
90          <xs:element ref="plugin" minOccurs="0"
91            maxOccurs="unbounded" />
92        </xs:choice>
93      </xs:sequence>
94      <xs:attribute name="panel_name" type="xs:string"
95        use="required" />
96    </xs:complexType>
97  </xs:element>
98
99  <!--Das Hauptelement der GUI-Beschreibung des mindestens
        ein Panel-Element beinhalten muss. -->
100 <xs:element name="GUI4DB">
```

```
101       <xs:complexType>
102         <xs:sequence>
103           <xs:element ref="panel" minOccurs="1"
104             maxOccurs="unbounded" />
105         </xs:sequence>
106       </xs:complexType>
107     </xs:element>
108
109 </xs:schema>
```

# Literaturverzeichnis

[Abecasis u. a. 2002] ABECASIS, Goncalo R. ; CHERNY, Stacey S. ; COOKSON, William O. ; CARDON, Lon R.: Merlin - Rapid Analysis of Dense Genetic Maps using Sparse Gene Flow Trees. In: *Nature Genetics* 30 (2002), S. 97 – 101

[Aldenderfer u. Blashfield 1985] ALDENDERFER, Mark S. ; BLASHFIELD, Roger K.: *Cluster Analysis*. Beverly Hills, London, New Delhi: Sage Publications, 1985 (Quantitative Applications in the Social Sciences)

[Allen 2001] ALLEN, Rob: *Workflow: An Introduction*. http://www.wfmc.org/Download-document/Workflow-An-Introduction.html, 2001

[Alschuler 2005] ALSCHULER, Liora: *Quick Start Guide for Simple CDA Release 2.0 Documents*. Oktober 2005

[Arasu u. a. 2001] ARASU, Arvind ; NOVAK, Jasmine ; TOMKINS, Andrew ; TOMLIN, John A.: *PageRank Computation and the Structure of the Web: Experiments and Algorithms*. 2001

[Ashburner u. a. 2000] ASHBURNER, Michael ; BALL, Catherine A. ; BLAKE, Judith A. ; BOTSTEIN, David ; BUTLER, Heather ; CHERRY, J. M. ; DAVIS, Allan P. ; DOLINSKI, Kara ; DWIGHT, Selina S. ; EPPIG, Janan T. ; HARRIS, Midori A. ; HILL, David P. ; ISSEL-TARVER, Laurie ; KASARSKIS, Andrew ; LEWIS, Suzanna ; MATESE, John C. ; RICHARDSON, Joel E. ; RINGWALD, Martin ; RUBIN, Gerald M. ; SHERLOCK, Gavin: Gene Ontology: Tool for the Unification of Biology. In: *Nature Genetics* 25 (2000), S. 25 – 29

[Bacher 1994] BACHER, Johann: *Clusteranalyse: Anwendungsorientierte Einführung*. München: Oldenbourg, 1994

[Backofen u. a. 2004] BACKOFEN, Rolf ; BADEA, Mike ; BURGER, Albert ; FAGES, Francois ; LAMBRIX, Patrick ; NUTT, Werner ; SCHROEDER, Michael ; SOLIMAN, Sylvain ; WILL, Sebastian: *State-of-the-art in Bioinformatics*. August 2004

[Bafna u. a. 2003] BAFNA, Vineet ; HALLDORSSON, Bjarni V. ; SCHWARTZ, Russel ; CLARK, Andrew G. ; ISTRAIL, Sorin: Haplotypes and Informative SNP Selection Algorithms: Don't Block out Information. In: *Proceedings of the seventh annual international conference on Research in computational molecular biology*, 2003, S. 19–27

[Bar-Yossef u. a. 2002] *Kapitel* K-ary Clustering with Optimal Leaf Ordering for Gene Expression Data. In: BAR-YOSSEF, Ziv ; DEMAINE, Erik D. ; GIFFORD, David K. ; HAMEL, Angele M. ; JAAKKOLA, Tommi S. ; SREBRO, Nathan: *Algorithms in Bioinformatics*. Berlin, Heidelberg, New York: Springer Verlag, 2002, S. 506–520

[Bar-Yossef u. a. 2004] BAR-YOSSEF, Ziv ; KUMAR, Ravi ; BRODER, Andrei Z. ; TOMKINS, Andrew: Sic Transit Gloria Telae: Towards an Understanding of the Web's Decay. In: *Proceedings of the 13th international conference on World Wide Web WWW '04*, 2004, S. 328–337

[Bauer 2003] BAUER, Ingmar: *Link-basierte Ranking-Verfahren in Internet-Suchsystemen*, University of Leipzig, Diplomarbeit, 2003

[Ben-Dor u. a. 1999] BEN-DOR, Amir ; SHAMIR, Ron ; YAKHINI, Zohar: Clustering Gene Expression Patterns. In: *Journal of Computational Biology* 6 (1999), Nr. 3/4, S. 281–297

[Bevan 2001] BEVAN, Nigel: International Standards for HCI and Usability. In: *International Journal of Human Computer Studies* 55 (2001), Nr. 4, S. 533–552

[Bharat u. Mihaila 1999] BHARAT, Krishna ; MIHAILA, George A.: *Hilltop: A Search Engine based on Expert Documents*. http://ftp.cs.toronto.edu/pub/reports/csrg/405/hilltop.html, 1999

[Bilu u. Linial 2002] *Kapitel* Functional Consequences in Metabolic Pathways from Phylogenetic Profiles. In: BILU, Yonatan ; LINIAL, Michal: *Algorithms in Bioinformatics*. Berlin, Heidelberg, New York: Springer Verlag, 2002, S. 263–276

[Boehm 2000] BOEHM, Markus: *Entwicklung von Workflow-Typen*. Berlin, Heidelberg, New York: Springer Verlag, 2000

[Bolshakova u. Azuaje 2003] BOLSHAKOVA, N ; AZUAJE, F: Cluster Validation Techniques for Genome Expression Data. In: *Signal Processing* 83 (2003), Nr. 4, S. 825 – 833

[Bolshakova u. a. 2005] BOLSHAKOVA, Nadia ; AZUAJE, Francisco ; CUNNINGHAM, Padraig: An integrated tool for microarray data clustering and cluster validity assessment. In: *Bioinformatics* 21 (2005), Nr. 4, S. 451–455

[Broeckel u. a. 2002] BROECKEL, Ulrich ; HENGSTENBERG, Christian ; MAYER, Björn ; HOLMER, Stephan ; MARTIN, Lisa J. ; COMUZZIE, Anthony G. ; BLANGERO, John ; NÜRNBERG, Peter ; REIS, André ; RIEGGER, Günter A.J. ; JACOB, Howard J. ; SCHUNKERT, Heribert: A Comprehensive Linkage Analysis for Myocardial Infarction and its Related Risk Factors. In: *Nature Genetics* 30 (2002), S. 210 – 214

[Calinescu u. a. 2007] CALINESCU, Radu ; HARRIS, Steve ; GIBBONS, Jeremy ; DAVIES, Jim ; TOUJILOV, Igor ; NAGL, Sylvia B.: Model-driven architecture for cancer research. In: *Proc. 5th IEEE Int. Conf. on Software Engineering and Formal Methods*, 2007, S. 59–68

[Carlson u. a. 2004] CARLSON, Christopher S. ; EBERLE, Michael A. ; RIEDER, Mark J. ; YI, Qian ; KRUGLYAK, Leonid ; NICKERSON, Deborah A.: Selecting a Maximally Informative Set of Single-Nucleotide Polymorhpisms for Association Analyses Using Linkage Disequilibrium. In: *American Journal of Human Genetics* 74 (2004), S. 106–120

[Cheng u. Atlee 2007] CHENG, Betty H. ; ATLEE, Joanne M.: Research Directions in Requirements Engineering. In: *FOSE '07: 2007 Future of Software Engineering*, IEEE Computer Society, 2007, S. 285–303

[Chin 2003] CHIN, J J.: The Use of Information Technology in Medicine: Defining its Role and Limitations. In: *Singapore Med Journal* 44 (2003), Nr. 3, S. 149–151

[Clarke u. a. 2000] CLARKE, Charles L A. ; CORMACK, Gordon V. ; TUDHOPE, Elizabeth A.: Relevance Ranking for One to Three Term Queries. In: *Information Processing and Management* 36 (2000), S. 291–311

[Clayton 2005] CLAYTON, David: *Linkage Disequilibrium*. http://www-gene.cimr.cam.ac.uk/clayton/courses/florence05/lectures/ld-lecture.pdf, 2005

[Coburger u. a. 2002] *Kapitel* Medizinische Statistik. In: COBURGER, S. ; HELLMICH, M. ; HILGERS, R. D. ; LEHMACHER, W. ; REINEKE, T. ; WASSMER, G.: *Handbuch der Medizinischen Informatik*. München, Wien: Carl Hanser Verlag, 2002, S. 225–276

[Cohen 2004] COHEN, Jacques: Bioinformatics:an introduction for computer scientists. In: *ACM Computing Surveys (CSUR)* 36 (2004), Nr. 2, S. 122 – 158

[Consortium ] CONSORTIUM, Gene O.: *An Introduction to the Gene Ontology*. `http://www.geneontology.org/GO.doc.shtml`, Abruf: January 2010

[Curtis u. a. 2001] CURTIS, D. ; NORTH, B. V. ; SHAM, P. C.: Use of an Artificial Neural Network to Detect Association between a Disease and Multiple Marker Genotypes. In: *Annals of Human Genetics* 65 (2001), S. 95 – 107

[Curtis 2000] *Kapitel* Data Analysis for Genetic Studies. In: CURTIS, David: *SNP and Microsatellite Genotyping: Markers for Genetic Analysis*. Natick, MA: Eaton Publishing, 2000, S. 81 – 116

[Date 2004] DATE, C J.: *An Introduction to Database Systems*. 8. Boston, London: Pearson Education, 2004

[Dhillon u. a. 2002] DHILLON, Inderjit S. ; GUAN, Yuqiang ; KOGAN, J: Iterative Clustering of High Dimensional Text Data Augmented by Local Search. In: *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, IEEE Computer Society, 2002, S. 131 – 138

[Dicks u. Savva 2003] *Kapitel* Comparative Genomics. In: DICKS, J. ; SAVVA, G.: *Handbook of Statistical Genetics*. West Sussex: John Wiley and Sons Ltd., 2003, S. 139 – 161

[Dimmer ] DIMMER, Emily: *Private Email*

[Dix 1991] DIX, Alan: *Formal Methods for Interactive Systems*. London u.a., Academic Press, 1991

[Dolin u. a. 2001] DOLIN, Robert H. ; ALSCHULER, Liora ; BEEBE, Calvin ; BIRON, Paul V. ; BOYER, Sandra L. ; ESSIN, Daniel ; KIMBER, Elliot ; LINCOLN, Tom ; MATTISON, John E.: The HL7 Clinical Document Architecture. In: *Journal of the American Medical Informatics Association* 8 (2001), Nov/Dec, Nr. 6, S. 552 – 569

[Dugas u. a. 2002] DUGAS, M. ; SCHOCH, C. ; SCHNITTGER, S. ; KERN, W. ; HAFERLACH, T. ; MESSERER, D. ; ÜBERLA, K.: Impact of integrating clinical and genetic information. In: *In Silico Biology* 2 (2002), Nr. 3, S. 383–391

[Eiron u. a. 2004] EIRON, Nadav ; MCCURLEY, Kevin S. ; TOMLIN, John A.: Ranking the web frontier. In: *WWW '04: Proceedings of the 13th international conference on World Wide Web*, ACM Press, 2004, S. 309–318

[Eisen u. a. 1998] EISEN, Michael B. ; SPELLMAN, Paul T. ; BROWN, Patrick O. ; BOTSTEIN, David: Cluster analysis and display of genome-wide expression patterns. In: *Proc Natl Acad Sci* 95 (1998), S. 14863–14868

[Erciyeş 2010] ERCIYEŞ, Kayhan: *Design of Algorithms for Bioinformatics*. 2010

[Ester u. a. 1996] ESTER, Martin ; KRIEGEL, Hans-Peter ; SANDER, Jörg ; XU, Xiaowei: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996

[etu ] ETU: *Brenner discusses turning data into knowledge*. Webpage. `http://www.princeton.edu/main/news/archive/S00/98/95S60/index.xml`, Abruf: August 2009

[Falconer 1996] FALCONER, D. S.: *Introduction to Quantitative Genetics*. 4. Essex: Pearson Educational Ltd., 1996

[Furlanello u. a. 2003] FURLANELLO, Cesare ; SERAFINI, Maria ; MERLER, Stefano ; JURMAN, Guiseppe: Entropy-based gene ranking without selection bias for the predictive classification of microarray data. In: *BMC Bioinformatics* 54 (2003), November, Nr. 4

[Gaasterland u. a. 2005] GAASTERLAND, Terry ; JAGADISH, H. V. ; RASCHID, Louiqa: Special issue on data management, analysis, and mining for the life sciences. In: *The VLDB Journal* 14 (2005), Nr. 3, S. 279 – 280

[Gabriel u. Röhrs 2003] GABRIEL, Roland ; RÖHRS, Heinz-Peter: *Gestaltung und Einsatz von Datenbanksystemen: Data Base Engineering und Datenbankarchitekturen*. Berlin, Heidelberg, New York: Springer Verlag, 2003

[Gat-Viks u. a. 2003] GAT-VIKS, I. ; SHARAN, R. ; SHAMIR, R.: Scoring clustering solutions by their biological relevance. In: *Bioinformatics* 19 (2003), Nr. 18, S. 2381–2389

[Genomatix 2007] GENOMATIX: *Bibliosphere Pathway Edition*. Bayerstr. 85a 80335 Munich, 2007

[Geßner 2006] GESSNER, Christof: *Kodierung von Einheiten physikalischer Messgrößen in HL7-Nachrichten mit UCUM*. HL7 Benutzergruppe in Deutschland e. V., Juli 2006

[Gillen u. a. 2004] GILLEN, John E. ; TSE, Tony ; IDE, Nicholas C. ; MCCRAY, Alexa T.: *Design, Implementation and Management of a Web-Based Data Entry System for ClinicalTrials.gov*. 2004

[Glenisson u. a. 2003] GLENISSON, Patrick ; MATHYS, Janick ; MOOR, Bart de: Meta-clustering of gene expression data and literature-based information. In: *SIGKDD Explorations* (2003), S. 101–112

[Graubner 1990] GRAUBNER, Bernd: Quantitative und qualitative Resultate einer routinemäßigen Basisdokumentation. In: GIANI, Guido (Hrsg.) ; REPGES, Rudolf (Hrsg.): *Biometrie und Informatik - neue Wege zur Erkenntnisgewinnung in der Medizin: 34. Jahrestagung der GMDS* Bd. 71, Berlin, Heidelberg, New York: Springer Verlag, 1990, S. 159–166

[Group 2001] GROUP, The International SNP Map W.: A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. In: *Nature* 409 (2001), S. 928 – 933

[Guha u. a. 2004] *Kapitel* Techniques for Clustering Massive Data Sets. In: GUHA, Sudipto ; RASTOGI, Rajeev ; SHIM, Kyuseok: *Clustering and Information Retrieval*. Dordrecht: Kluwer Academic Plublishers, 2004, S. 35 – 84

[Gumbel u. a. 2002] *Kapitel* Modellierung biologischer Prozesse. In: GUMBEL, Markus ; GREBE, Reinhard ; KNAPP-MOHAMMADY, Michaela ; ULLMANN, G. M. ; LANGOWSKI, Joerg: *Handbuch der Medizinischen Informatik*. München, Wien: Carl Hanser Verlag, 2002, S. 169–224

[Hampe u. a. 2003] HAMPE, Jochen ; SCHREIBER, Stefan ; KRAWCZAK, Michael: Entropy-based SNP selection for genetic association studies. In: *Human Genetics* (2003), S. 36–43

[Hartmann u. a. 2010] HARTMANN, Sven ; KÖHLER, Henning ; WANG, Jing: Ontology Consolidation in Bioinformatics. In: LINK, Sebastian (Hrsg.) ; GHOSE, Aditya K. (Hrsg.): *Proc. 7th Asia-Pacific Conference on Conceptual Modelling (APCCM 2010)*, 2010

[He u. a. 2004] *Kapitel* On Quantitative Evaluation of Clustering Systems. In: HE, Ji ; TAN, Ah-Hwee ; TAN, Chew-Lim ; SUNG, Sam-Yuan: *Clustering and Information Retrieval*. Dordrecht: Kluwer Academic Plublishers, 2004, S. 105 – 134

[Hengstenberg 2006] HENGSTENBERG, Christian: Herzinfarkt-Familienstudie schafft neue Möglichkeiten zur Erkennung von Hochrisikopatienten. In: *Deutsche Zeitschrift fuer klinische Forschung* (2006)

[Houle u. a. 2000] HOULE, John L. ; CADIGAN, Wanda ; HENRY, Sylvain ; PINNAMANENI, Anu: *Database Mining in the Human Genome Initiative*. 2000

[Hubbard u. et al. 2005] HUBBARD, T. ; AL., D. A.: Ensembl 2005. In: *Nucleic Acids Research* 33 (2005), S. 447 – 453

[Huett u. Dehnert 2006] HUETT, Marc-Thorsten ; DEHNERT, Manuel: *Methoden der Bioinformatik*. Berlin, Heidelberg, New York: Springer Verlag, 2006

[Hultsch 1992] *Kapitel* Systematisierte Nomenklatur und Klassifikation in der Medizin. In: HULTSCH, E.: *Traumatologie aktuell - Dokumentation und Archivierung im Krankenhaus*. Bd. 7. New York: Thieme, 1992, S. 32 – 43

[Hultsch 1990] HULTSCH, Ekhard: Prinzipien zur semantischen Strukturierung von Dokumenten. In: GIANI, Guido (Hrsg.) ; REPGES, Rudolf (Hrsg.): *Biometrie und Informatik - neue Wege zur Erkenntnisgewinnung in der Medizin: 34. Jahrestagung der GMDS* Bd. 71, Berlin, Heidelberg, New York: Springer Verlag, 1990, S. 153–158

[Hwang u. a. 2007] HWANG, S.-H. ; OH, H.-B. ; CHOI, S.-E. ; HONG, S. P. ; YOO, W.: Effective screening of informative single nucleotide polymorphisms using the novel method of restriction fragment mass polymorphism. In: *Journal of international medical research* 35 (2007), Nr. 6, S. 827–835

[IHTSDO 2009] IHTSDO: *SNOMED Clinical Terms User Guide*. The International Health Terminology Standards Development Organisation, Juli 2009

[Jablonski u. a. 1997] JABLONSKI, Stefan ; BÖHM, Markus ; SCHULZE, Wolfgang: *Workflow-Management: Entwicklung von Anwendungen und Systemen*. Heidelberg: dpunkt Verlag, 1997

[Jagadish u. Olken 2004] JAGADISH, H. V. ; OLKEN, Frank: Database management for life sciences research. In: *SIGMOD Rec.* 33 (2004), Nr. 2, S. 15–20

[Janssen u. a. 1993] JANSSEN, Christian ; WEISBECKER, Anette ; ZIEGLER, Jürgen: Generating user interfaces from data models and dialogue net specifications. In: *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, ACM, 1993, S. 418–423

[Janssen u. a. 2003] JANSSEN, Paul ; ENRIGHT, Anton J. ; AUDIT, Benjamin ; CASES, Ildefonso ; GOLDOVSKY, Leon ; HARTE, Nicola ; KUNIN, Victor ; OUZOUNIS, Christos A.: COmplete GENome Tracking (COGENT): a flexible data

environment for computational genomics. In: *Bioinformatics* 19 (2003), Nr. 11, S. 1451–1452

[Jiang u. Zhang 2002] JIANG, Daxin ; ZHANG, Aidong: Cluster Analysis for Gene Expression Data: A Survey. In: *IEEE Transactions on Knowledge and Data Engineering* 16 (2002), Nr. 11, S. 1370–1386

[Jukic 2006] JUKIC, Nenad: Modeling strategies and alternatives for data warehousing projects. In: *Communications of the ACM* 49 (2006), Nr. 4, S. 83–88

[Jüni u. a. 2001] JÜNI, Peter ; ALTMANN, Douglas G. ; EGGER, Matthias: Assessing the quality of controlled clinical trials. In: *British Mecial Journal* (2001), S. 42–6

[Kaufman u. Rousseeuw 1990] KAUFMAN, L. ; ROUSSEEUW, P. J.: *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990

[Kho u. a. 2007] KHO, Abel ; ZAFAR, Atif ; TIERNEY, William: Information Technology in PBRNs: The Indiana University Medical Group Research Network (IUMG ResNet) Experience. In: *Journal of the American Board of Family Medicine* 20 (2007), Nr. 2, S. 196 – 203

[Kleinberg 1998] KLEINBERG, Jon: Authoritative sources in a hyperlinked environment. In: *Proc. Ninth Ann. ACM-SIAM Symp. Discrete Algorithms* Bd. 46 ACM, ACM Press, 1998, 604–632

[Koehler u. a. 2002] *Kapitel* Medizinische Informatik. In: KOEHLER, Claus O. ; BEXTEN, Erdmuthe M. ; LEHMANN, Thomas M.: *Handbuch der Medizinischen Informatik*. München, Wien: Carl Hanser Verlag, 2002, S. 1–21

[Kohonen 1997] KOHONEN, Teuvo: *Self-Organizing Maps*. Berlin, Heidelberg, New York: Springer Verlag, 1997

[Krcmar 2005] KRCMAR, Helmut: *Informationsmanagement*. Berlin, Heidelberg, New York: Springer Verlag, 2005

[Kriegel u. Kröger 2004] KRIEGEL, Karin KailingAND Hans-Peter ; KRÖGER, Peer: Density-Connected Subspace Clustering for High-Dimensional Data. In: *Proceedings of 4th SIAM Int. Conf. on Data Mining*, 2004, S. 246–257

[Kruglyak 1999] KRUGLYAK, Leonid: Prospects for whole-genome linkage disequilibrium mapping of common disease genes. In: *Nature Genetics* 22 (1999), June, S. 139–144

[Langville u. Meyer 2006] LANGVILLE, Amy N. ; MEYER, Carl D.: *Google's PageRank and Beyond: The Sciene of Search Engine Rankings*. Princeton: Princeton University Press, 2006

[Leiner u. a. 2003] LEINER, Florian ; GAUS, Wilhelm ; HAUX, Reinhold ; KNAUP-GREGORI, Petra ; PFEIFFER, Karl-Peter: *Medizinische Dokumentation: Grundlagen einer qualitätsgesicherten integrierten Krankenversorgung*. Stuttgart, New York: Schattauer, 2003

[Levine u. Domany 2001] LEVINE, Erel ; DOMANY, Eytan: Resampling Method for Unsupervised Estimation of Cluster Validity. In: *Neural Computation* 13 (2001), S. 2573–2593

[Li u. Miller 2007] LI, Hui ; MILLER, Eva: *Clinical Teams and Tools: New Approaches for Smarter Clinical Trials*. 2007

[Lim u. a. 2008] LIM, Jeongheui ; BHAK, Jong ; OH, Hee-Mock ; KIM, Chang-Bae ; PARK, Yong-Ha ; PAEK, Woon K.: An Integrated Korean Biodiversity and Genetic Information Retrieval System. In: *BMC Bioinformatics* 9 (2008)

[Liu u. Logvinenko 2003] *Kapitel* Bayesian Methods in Biological Sequence Analysis. In: LIU, J. S. ; LOGVINENKO, T.: *Handbook of Statistical Genetics*. West Sussex: John Wiley and Sons Ltd., 2003, S. 66 – 93

[Lloyd 1982] LLOYD, S. P.: Least square quantization in PCM. In: *IEEE Transactions on Information Theory* 28 (1982), S. 129–137

[Lord u. a. 2002] LORD, P. W. ; STEVENS, R. D. ; BRASS, A. ; GOBLE, C. A.: Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. In: *Bioinformatics* 19 (2002), Nr. 10, S. 1275–1283

[Manning u. a. 2009] MANNING, Christopher D. ; RAGHAVAN, Prabhakar ; SCHÜTZE, Hinrich: *An Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2009

[Martin u. a. 2004] MARTIN, David ; BRUN, Christine ; REMY, Elisabeth ; MOUREN, Pierre ; THIEFFRY, Denis ; JACQ, Bernard: GOToolBox: functional analysis of gene datasets based on Gene Ontology. In: *Genome Biology* 5 (2004), Nr. 12, S. 5:R101

[Mathworks ] MATHWORKS, The: *Hierarchical Clustering*. `http://www.mathworks.com/access/helpdesk/help/toolbox/stats/bq_679x-3.html`, Abruf: Februar 2010

[Matuschka 1975] MATUSCHKA, Michael: *Heuristik: Geschichte des Wortes und der Versuche zur Entwicklung allgemeiner und spezieller Theorien von der Antike bis Kant*. Universitaet Duesseldorf, 1975

[McDonald u. a. 2009] MCDONALD, Clem ; HUFF, Stan ; MERCER, Kathy ; HERNANDEZ, Jo A. ; VREEMAN, Daniel J.: *Logical Observation Identifiers Names and Codes (LOINC) Users' Guide*. LOINC, Dezember 2009

[Meenen 1992] *Kapitel* Dialoggesteuerte PC-Dokumentationssysteme. In: MEENEN, N. M.: *Traumatologie aktuell - Dokumentation und Archivierung im Krankenhaus*. New York: Thieme, 1992, S. 44 – 53

[Meunier u. Eyre-Walker 2001] MEUNIER, Julien ; EYRE-WALKER, Adam: The Correlation Between Linkage Disequilibrium and Distance: Implications for Re-

combination in Hominid Mitochondria. In: *Molecular Biology and Evolution* 18 (2001), S. 2132–2135

[Miyamoto 2003] MIYAMOTO, Sadaaki: Information Clustering Based on Fuzzy Multisets. In: *Information Processing and Management* 39 (2003), S. 195–213

[Morrison u. a. 2005] MORRISON, Julie L. ; BREITLING, Rainer ; HIGHAM, Desmond J. ; GILBERT, David R.: GeneRank: Using Search Engine Technology for the Analysis of Microarray Experiments. In: *BMC Bioinformatics* 6 (2005), S. 233

[Mougenot 1995] MOUGENOT, Isabelle: Genetic Sequence Annotation within Biological Databases. In: *Database Systems for Advanced Applications*, 1995, S. 333–341

[Mueller 1992] *Kapitel* Bericht am Krankenbett - Zum Beginn der medizinischen Dokumentation und Statistik. In: MUELLER, I.: *Traumatologie aktuell - Dokumentation und Archivierung im Krankenhaus*. Bd. 7. New York: Thieme, 1992, S. 1–7

[OASIS 2008] OASIS: *User Interface Markup Language (UIML) Version 4.0*. http://www.oasis-open.org/committees/download.php/28457/uiml-4.0-cd01.pdf, January 2008

[Oestereich 2001] OESTEREICH, Bernd: *Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language*. München: Oldenbourg, 2001

[Onkamo u. a. 2002] ONKAMO, Päivi ; OLLIKAINEN, Vesa ; SEVON, Petteri ; TOIVONEN, Hannu T. T. ; MANNILA, Heikki ; KERE, J.: Association Analysis for Quantitative Traits by Data Mining: QHPM. In: *Annals of Human Genetics* 66 (2002), S. 419 – 429

[Opgen-Rhein u. Strimmer 2007] OPGEN-RHEIN, Rainer ; STRIMMER, Korbinian: Accurate Ranking of Differentially Expressed Genes by a Distribution-Free Shrinkage Approach. In: *Statistical Applications in Genetics and Molecular Biology* 6 (2007), Nr. 1

[Page u. a. 1998] PAGE, Lawrence ; BRIN, Sergey ; MOTWANI, Rajeev ; WINOGRAD, Terry: The PageRank Citation Ranking: Bringing Order to the Web / Stanford Digital Library Technologies Project. 1998. – Forschungsbericht

[Pal u. Mitra 2004] PAL, Sankar K. ; MITRA, Pabitra: *Pattern Recognition Algorithms for Data Mining*. London, New York: Chapman & Hall, 2004

[Palakal u. a. 2002] PALAKAL, Mathew ; MUKHOPADHYAY, Snehasis ; MOSTAFA, Javed: An intelligent biological information management system. In: *Proceedings of the 2002 ACM symposium on Applied computing*, 2002, S. 159 – 163

[Peltz 2005] *Kapitel* Computational Biology: Are We There Yet. In: PELTZ, Gary: *Computational Genetics and Genomics*. New Jersey: Humana Press Inc., 2005, S. 3–32

[Perez u. a. 2004] PEREZ, Antonio J. ; PEREZ-IRATXETA, Carolina ; BORK, Peer ; THODE, Guillermo ; ANDRADE, Miguel A.: Gene Annotation from Scientific Literature using Mappings between Keyword Systems. In: *Bioinformatics* 20 (2004), Nr. 13, S. 2084–2091

[Petersohn 2005] PETERSOHN, H.: *Data Mining*. München: Oldenbourg, 2005

[Potamias 2006] POTAMIAS, George: *State of the Art on Systems for Data Analysis, Information Retrieval and Decision Support*. January 2006

[Puerta u. Eisenstein 2002] PUERTA, Angel ; EISENSTEIN, Jacob: XIML: A Universal Language for User Interfaces. In: *Computer-Aided Design of User Interfaces III, Proceedings of the Fourth International Conference on Computer-Aided*

*Design of User Interfaces*, Dordrecht: Kluwer Academic Publishers, 2002, S. 275–282

[Purcell u. a. 2007] PURCELL, S ; NEALE, B ; TODD-BROWN, K ; THOMAS, L ; FERREIRA, MAR ; BENDER, D ; MALLER, J ; SKLAR, P ; BAKKER, PIW de ; MJ, MJ D. ; SHAM, PC: PLINK: a toolset for whole-genome association and population-based linkage analysis. In: *American Journal of Human Genetics* (2007), S. 559—575

[Ravat u. a. 1999] RAVAT, Franck ; TESTE, Olivier ; ZURFLUH, Giles: Towards data warehouse design. In: *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*. New York, NY, USA : ACM Press, 1999, S. 359–366

[Richter 2002] RICHTER, Kai: Generic Interface Descriptions using XML. In: KOLSKI, Christophe (Hrsg.) ; VANDERDONCKT, Jean (Hrsg.): *Computer-Aided Design of User Interfaces III, Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces*, 2002, S. 275–282

[Richter 2007] RICHTER, Kai: *Methoden zur Unterstützung bei der Entwicklung plattformübergreifender Benutzerschnittstellen*, Technische Universität Darmstadt, Diss., 2007

[Roizes 2000] *Kapitel* Identification of Microsatellite Markers: Screening for Repeat Sequences and Mapping Polymorphisms. In: ROIZES, Gerard: *SNP and Microsatellite Genotyping: Markers for Genetic Analysis*. Natick, MA: Eaton Publishing, 2000, S. 35–48

[Sammon u. Finnegan 2000] SAMMON, David ; FINNEGAN, Pat: The ten commandments of data warehousing. In: *SIGMIS Database* 31 (2000), Nr. 4, S. 82–91

[Sayers u. a. 2000] *Kapitel* MADGE and Other Single Nucleotide Polymorphisms Analysis Methods: Application to the Molecular Genetic Epidemology of Asthme and Cardivascular Disease. In: SAYERS, Ian ; CHEN, Xiao he ; YE, Shu ; BEGHE, Bianca ; DAY, Ian N. M.: *SNP and Microsatellite Genotyping: Markers for Genetic Analysis*. Natick, MA: Eaton Publishing, 2000, S. 13 – 34

[Schanze 1992] *Kapitel* Gesamtkonzeption zur Informationsverarbeitung im Krankenhaus. In: SCHANZE, M.: *Traumatologie aktuell - Dokumentation und Archivierung im Krankenhaus*. Bd. 7. New York: Thieme, 1992, S. 154 – 162

[Scherf u. a. 2005] SCHERF, M ; EPPLE, A ; WERNER, T: The next Generation of Literature Analysis: Integration of Genomic Analysis into Text Mining. In: *Briefings in Bioinformatics* 6 (2005), Nr. 3, S. 287 – 297

[Schlicker u. a. 2006] SCHLICKER, Andreas ; DOMINGUES, Francisco S. ; RAHNENFUHRER, Joerg ; LENGAUER, Thomas: A New Measure for Functional Similarity of Gene Products based on Gene Ontology. In: *BMC Bioinformatics* 7 (2006), S. 302

[Schnetzer 1999] SCHNETZER, Ronald: *Workflow-Management kompakt und verständlich*. Braunschweig: Vieweg Verlag, 1999

[Scognamillo u. a. 1999] SCOGNAMILLO, Roberto ; STROZZI, Carlo ; VINCENZI, Beatrice ; RECCHIA, Guiseppe: Clinical Trial Management and Remote Data Entry on the Internet. In: *Drug Information Journal* 33 (1999), S. 1061–1065

[Sevon 2004] SEVON, Petteri: *Algorithms for Association-Based Gene Mapping*, University of Helsinki, Diss., 2004

[Shamir u. Sharan 2000] SHAMIR, Ron ; SHARAN, Roded: CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis. In: *ISMB '00*, Menlo Park: AAAI Press, 2000, S. 307–316

[Shamir u. Sharan 2001] *Kapitel* Algorithmic Approaches to Clustering Gene Expression Data. In: SHAMIR, Ron ; SHARAN, Roded: *Current Topics in Computational Biology*. MIT Press, 2001, S. 269–300

[Shannon 1948] SHANNON, C. E.: A Mathematical Theory of Communication. In: *The Bell System Technical Journal* 27 (1948), S. 379–423

[Slonim 2002] SLONIM, Donna K.: From patterns to pathways: gene expression data analysis comes of age. In: *Nature Genetics* 32 (2002), S. 502–508

[Speed u. Zhao 2003] *Kapitel* Chromosome Maps. In: SPEED, T. P. ; ZHAO, H.: *Handbook of Statistical Genetics*. West Sussex: John Wiley and Sons Ltd., 2003, S. 3 – 38

[Steinbach 2000] STEINBACH, Michael: *An introduction to Cluster Analysis for Data Mining*. http://www.cs.umn.edu/ han/dmclass/cluster$_s urvey_1 0_0 2_0 0.pdf$, 2000

[Stevens u. a. 2000] STEVENS, Robert ; GOBLE, Carole A. ; BECHHOFER, Sean: Ontology-based Knowledge Representation for Bioinformatics. In: *Briefings in Bioinformatics* 1 (2000), Nr. 4, S. 398–414

[Susol u. a. 2000] *Kapitel* High-Throughput Genotyping of Microsatellite Markers. In: SUSOL, Elene ; EYRE, Stephen ; JOHN, Sally: *SNP and Microsatellite Genotyping: Markers for Genetic Analysis*. Natick, MA: Eaton Publishing, 2000, S. 49 – 66

[Tamayo u. a. 1999] TAMAYO, Pablo ; SLONIM, Donna ; MESIROV, Jill ; ZHU, Qing ; KITAREEWAN, Sutisak ; DMITROVSKY, Ethan ; LANDER, Eric S. ; GOLUB, Todd R.: Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. In: *Proc Natl Acad Sci* 96 (1999), S. 2907 – 2912

[Tan u. a. 2005] TAN, Pang-Ning ; STEINBACH, Michael ; KUMAR, Vipin: *Introduction to Data Mining*. Addison Wesley, 2005

[Trewin u. a. 2003] TREWIN, Shari ; ZIMMERMANN, Gottfried ; VANDERHEIDEN, Gregg: Abstract user interface representations: how well do they support universal access? In: *CUU '03: Proceedings of the 2003 conference on Universal usability*, ACM, 2003, S. 77–84

[Tveit u. a. 2004] TVEIT, Henrik ; MOLLESTAD, Torulf ; LAEGREID, Astrid: The Alignment of the Medical Subject Headings to the Gene Ontology and Its Application in Gene Annotation. In: *Rough Sets and Current Trends in Computing*, Berlin, Heidelberg, New York: Springer Verlag, 2004, S. 798–804

[Tyrelle u. King 2003] TYRELLE, Greg D. ; KING, Garry C.: A platform for the description, distribution and analysis of genetic polymorphism data. In: *Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003* Bd. 19, Australian Computer Society, Inc., 2003, S. 173 – 180

[Vogler 1996] *Kapitel* Chancen und Risiken von Workflow-Management. In: VOGLER, Petra: *Praxis des Workflow-Managements*. Braunschweig: Vieweg Verlag, 1996, S. 343 –362

[Vossen 2009] VOSSEN, Gottfried: *Lexikon der Wirtschaftsinformatik*. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Datenmanagement/Datenbanksystem, 2009

[W3C 2002] W3C: *XML Encryption Syntax and Processing*. http://www.w3.org/TR/xmlenc-core/, Dezember 2002

[W3C 2004] W3C: *RDF/XML Syntax Specification (Revised)*. http://www.w3.org/TR/REC-rdf-syntax/, Februar 2004

[W3C 2008] W3C: *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. http://www.w3.org/TR/REC-xml/, November 2008

[W3C 2009] W3C: *XForms 1.1*. http://www.w3.org/TR/xforms11/, Oktober 2009

[Wegrzyn u. a. 2008] WEGRZYN, Jill L. ; LEE, Jennifer M. ; TEARSE, Brandon R. ; NEALE, David B.: TreeGenes: A Forest Tree Genome Database. In: *Int J Plant Genomics* (2008)

[Wehbe u. a. 2009] WEHBE, Firas H. ; BROWN, Steven H. ; MASSION, Pierre P. ; GADD, Cynthia S. ; MASYS, Daniel R. ; ALIFERIS, Constantin F.: A novel information retrieval model for high-throughput molecular medicine modalities. In: *Cancer Informatics* 9 (2009), S. 1–17

[Weijters u. a. 1997] WEIJTERS, Ton ; HERIK, H. Jaap Van D. ; BOSCH, Antal Van D. ; POSTMA, Eric: Avoiding overfitting with BP-SOM. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI'97*, San Francisco: Morgan Kaufmann, 1997, S. 1140–1145

[Winter u. a. 2002] *Kapitel* Krankenhausinformationssysteme. In: WINTER, Alfred ; AMMENWERTH, Elske ; BRIGL, Birgit ; HAUX, Reinhold: *Handbuch der Medizinischen Informatik*. München, Wien: Carl Hanser Verlag, 2002, S. 473–552

[Xiao u. a. 2003] XIAO, Xiang ; DOW, Ernst R. ; EBERHART, Russel ; MILED, Zina B. ; OPPELT, Robert J.: Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization. In: *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, Washington, DC: IEEE Computer Society, 2003, S. 154.2

[Xiong 2006] XIONG, Jin: *Essential Bioinformatics*. Cambridge: Cambridge University Press, 2006

[Yakir u. a. 2005] *Kapitel* Statistical Theory in QTL Mapping. In: YAKIR, Benjamin ; PISANTE, Anne ; DARVASI, Ariel: *Computational Genetics and Genomics*. New Jersey: Humana Press Inc., 2005, S. 33–50

[Yeung u. Haynor 2001] YEUNG, K Y. ; HAYNOR, D R.: Validating Clustering for Gene Expression Data. In: *Bioinformatics* 17 (2001), S. 309–318

[Zaiss u. a. 2002] *Kapitel* Medizinische Dokumentation, Terminologie und Linguistik. In: ZAISS, Albrecht ; GRAUBNER, Bernd ; INGENERF, Josef ; LEINER, Florian ; LOCHMANN, Ulrich ; SCHOPEN, Michael ; SCHRADER, Ulrich ; SCHULZ, Stefan: *Handbuch der Medizinischen Informatik*. München, Wien: Carl Hanser Verlag, 2002, S. 45–102

[Zhao u. a. 2005] ZHAO, Jinying ; BOERWINKLE, Eric ; XIONG, Momiao: An Entropy-Based Statistic for Genomewide Association Studies. In: *American Journal of Human Genetics* 77 (2005), S. 27–40

[Ziegler u. Koenig 2006] ZIEGLER, Andreas ; KOENIG, Inke: *A Statistical Approach to Genetic Epidemiology*. Weinheim: Wiley-VCH, 2006

# Tabellenverzeichnis

# Abbildungsverzeichnis

# Listings