Verfahren zur Genexpressionsanalyse

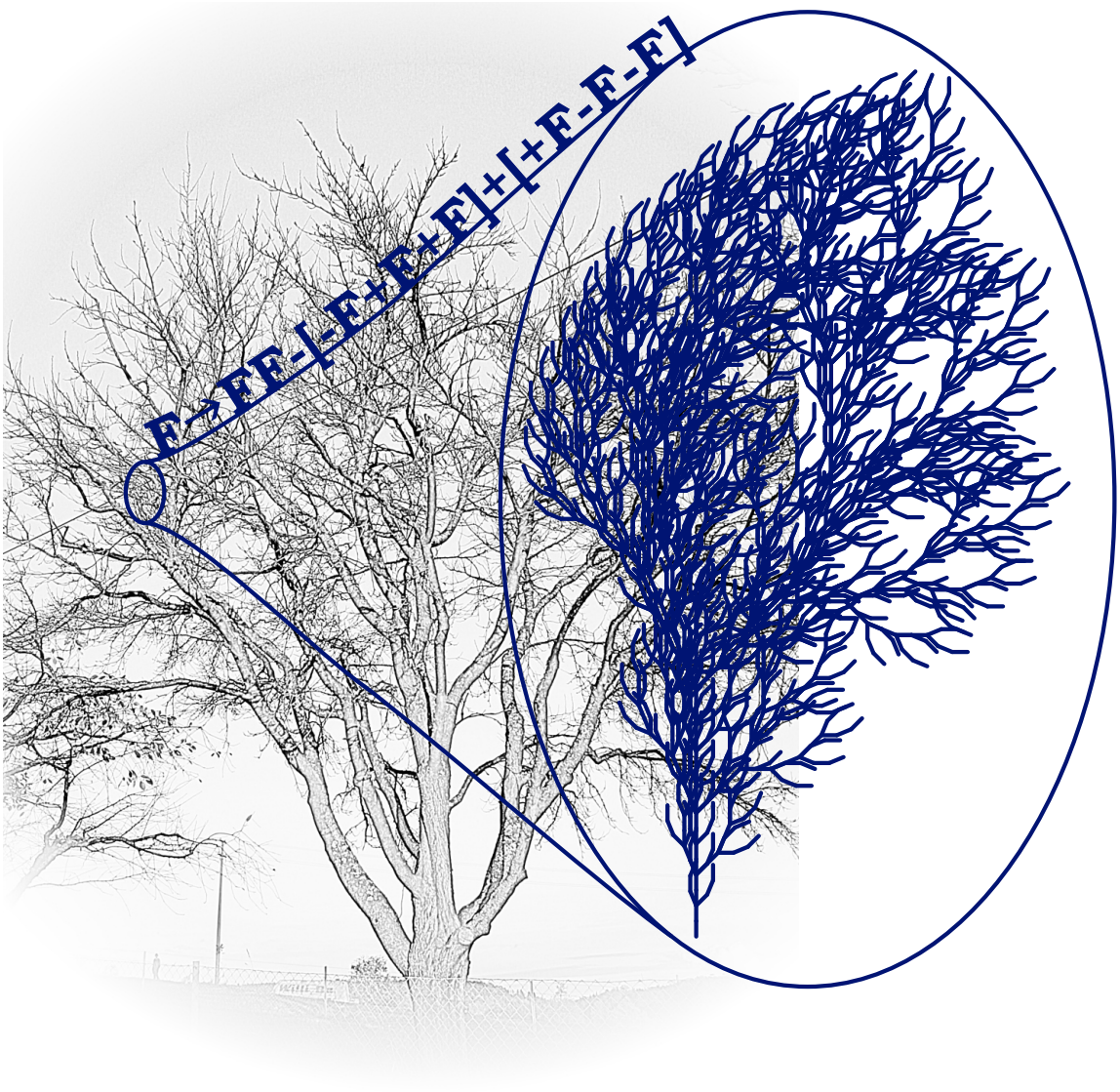
Dissertation

der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Inform. der Medizin Janko Dietzsch
aus Chemnitz

Tübingen
2009

Tag der mündlichen Qualifikation:	27.05.2009
Dekan:	Prof. Dr. Oliver Kohlbacher
1. Berichterstatter:	Prof. Dr. Daniel Huson
2. Berichterstatter:	Prof. Dr. Dirk Bartz (Universität Leipzig)



The aim of science is to seek the simplest explanation of complex facts ... Seek simplicity and distrust it.

A. N. Whitehead

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die im Wortlaut oder dem Sinn nach anderen Werken entnommen worden sind, durch die Angabe der zugehörigen Quelle gekennzeichnet sind.

Tübingen, den 28. September 2009 Janko Dietzsch

Zusammenfassung der Arbeit

Die vorliegende Arbeit befaßt sich mit verschiedenen Methoden der Auswertung von Microarraydaten für die Genexpressionsanalyse. Das Gebiet der Array-basierten Genexpressionsanalyse ist ein sehr umfangreiches und sich schnell entwickelndes Feld. Da es sich um komplexe Hochdurchsatzverfahren handelt, erfordert einerseits die Größe der erzeugten Datenmengen und andererseits die Art der Daten, sowie das Ziel ihrer Auswertung eine Verknüpfung einer Vielzahl von Methoden aus den Bereichen der Informatik, der Mathematik und der Biologie. Der Schwerpunkt dieser Arbeit liegt dabei auf einer geeigneten Verknüpfung informatischer, graphischer und mathematisch-statistischer Methoden.
Die Arbeit untergliedert sich in drei Teile:

- **Teil I** gibt einen Über- und Einblick zu verschiedensten Arraytechnologien, sowohl derzeitigen Stand der Genexpressionsanalyse mit Arrays (Kapitel 1), wie auch zu anderweitig eingesetzten und zukünftigen Arrayanwendungen (Kapitel 2). Ziel dieses Teils ist die Darstellung der Grundlage auf der die nachfolgend auszuwertenden Daten entstehen und das Aufzeigen der vorhandenen technologischen Vielfalt und Breite der Arraytechnologie in den Lebenswissenschaften.

- **Teil II** entwickelt zwei wesentliche, grundlegende Leitlinien, die sich aus den Anforderungen ergeben, wie sie die Auswertung derart großer und komplexer Datensätze stellen. Folgerichtig entprechen diesen Leitlinien zwei im Rahmen dieser Arbeit entstandenen größeren Projekte im Grundsatz.

  Mayday als eine generelle, große, vielfältige Methoden und Verfahren unterschiedlicher Bereiche integrierende Anwendungsplattform (Kapitel 3 und 4) entspricht der einen Linie. Die komplexe Natur der Daten erfordert den Zugriff auf eine große Menge von Auswertungsverfahren unter einer gemeinsamen Plattform und über ein konsistente, einheitliche Nutzerschnittstelle. Kapitel 5 beschreibt dies beispielhaft für die Integration statistischer Verfahren, sowie die dabei notwendigerweise zu beachtenden Randbedingungen innerhalb von Mayday im spezielleren und Java im allgemeineren.

  SpRay steht als Vertreter spezialisierter, kleinerer, auf hohe Leistung und effizienten Umgang mit Speicherplatz hin optimierten Anwendungen für die zweite Linie. Vordringlicher Anwendungsbereich von SpRay ist die explorative, stark graphisch gestützte Exploration von Daten. Dabei stellt es auch eine prototypische Anwendung des sich neu etablierenden Gebiets der Visual Analytics dar, die eine sehr enge direkte Verzahnung von Visualisierung, Statistik und Datenorganisation auszeichnet.

  Insgesamt wurde in beiden Projekten sehr hoher Stellenwert auf eine geignete graphische und interaktive Aufbereitung und Darstellung der Daten bzw. ihrer Analyseresultate gelegt. Für eine weitergehende Zusammenfassung sei hier auf die Kurzübersicht zu Teil II auf Seite 29 verwiesen.

- **Teil III** stellt schließlich die Anwendung der Arraytechnologie innerhalb der medizinischen Forschung in den Vordergrund. Anhand der in Kooperation mit der Transfusionsmedizin Tübingen durchgeführten Entwicklung einer kompletten Microarray-Plattform für die Genexpressionsanaylse von inflammatorischen und mit der Streßantwort assoziierten Genen werden die dabei auftretenden Anforderungen und ihre Lösung erläutert. Ziel war die Etablierung einer vollständigen Pipeline vom Entwurf des Arrays, über die Durchführung des Experiments, bis hin zur Auswertung und deren Validierung. Dabei lag die statistische Versuchsplanung, die Datenorganisation und -verwaltung, sowie die Entwicklung, die Anwendung und Etablierung passender Auswertungsverfahren im Verantwortungsbereich des Autors dieser Arbeit. Neben den Vorversuchen und deren Validierung (Kapitel 7) erfolgt auch eine Darstellung des erfolgreichen Einsatzes im Rahmen von zwei Studien aus der Sportmedizin (Kapitel 8) und der Psychologie (Kapitel 9). Dabei konnte die Funktionsfähigkeit und Brauchbarkeit der entwickelten Array-Plattform und ihrer Anwendungspipline sehr gut untermauert werden.

  Auch hier sei für eine weitergehende Zusammenfassung auf die Kurzübersicht zu Teil III auf Seite 153 verwiesen.

Erfahrungen, die während der praktischen Arbeit mit der Array-Plattform gemacht wurden, flossen auch in die Entwicklung von in Teil II erwähnten Verfahren und Anwendungen bzw. Module ein. Insgesamt ergaben sich zwischen allen Teilen enge Wechselbeziehungen, bei denen Erkenntnise in abgewandelter bzw. angepasster Form in den anderen Projekten ihren Wiederhall fanden.

In einer abschließenden Gesamtsicht sind die zentralen Projekte der Teile II und III gute Ausgangspunkte für weitergehende Anstrengungen. Es konnte jeweils ihre Nützlichkeit, ihre Funktionsfähigkeit und ihr Potential demonstriert werden.

Danksagung

In erster Linie gebührt mein Dank den Menschen, die mich auf dem Weg, der mit dieser Arbeit sein Ende findet, am engsten begleitet haben und deshalb auftretende Höhen und Tiefen besonders mit mir fühlen mussten und meine vollständige Anwesenheit des Öfteren schmerzlich vermissten. Sie gaben dabei nie auf und standen mir in allem hilfreich, ausdauernd und ermutigend zur Seite und wie ein ausgelassenes Kinderlachen so manchen dunklen Tag aufhellen kann wird nur der wirklich verstehen, der es selbst kennt. Mein Dank gilt in erster Linie euch beiden – Irina und Adrian.

Danken möchte ich auch ganz besonders meiner Betreuerin Frau Dr. Kay Nieselt für ihr Vertrauen in meine Arbeit, das Ermöglichen dieser Promotion und ihre Unterstützung über die Jahre hinweg. Herrn Prof. Dr. Daniel Huson danke ich für die Begutachtung der Arbeit und die Wahrnehmung weiterer organisatorischer Tätigkeiten, die er für diese Arbeit übernehmen musste. Mein Dank gilt Herrn Prof. Dr. Dirk Bartz nicht nur für die Übernahme des Zweitgutachtens, sondern auch für eine erfolgreiche und sehr angenehme Zusammenarbeit.

An dieser Stelle gilt mein Dank natürlich auch meinen Eltern und meinem Bruder. Für vieles was ich bin und was mich ausmacht habt ihr den Grundstein gelegt. Ich bin der „Apfelbaum“, den ihr gepflanzt habt. Ich bin dankbar für eine glückliche und eindrucksreiche Kindheit, die mich gelehrt hat, die Dinge von möglichst allen Seiten aus betrachten zu wollen.

Ebenso möchte ich mich ganz herzlich bei den Kooperationspartnern bedanken, die zum Gelingen der gemeinsamen Projekte ganz wesentlich beigetragen haben, wie Dr. Derek Zieker, Dr. Judith Zieker und Dr. Elvira Fehrenbach.

Danken möchte ich auch einer ganzen Schar von Kollegen und Mitarbeitern, sowohl für anregende, hilfreiche, fachliche und persönliche Gespräche sowie mitunter auch ihrer Hilfe zu einer angenehmeren Freizeitgestaltung. Namentlich möchte ich dabei meine Zimmerkollegen Stephan Steigele, Stephan Symons und Florian Battke, unsere Sekretärin Marie-Annick Pacquier, sowie die Kollegen von „oben-drüber“ – Christian Rausch, Tobias Dezulian, Sandra Gesing, Alexander Auch, Daniel Richter, Marine Gaudefroy-Bergmann, Badreddin Abolmaali, Christian Spieth und Nora Speer erwähnen.

Außerdem gebührt ein nicht unerheblicher Teil meines Dankes den Studenten, die durch ihre Arbeit in Form von Studien- oder Diplomarbeiten ganz maßgeblichen Anteil an den Erfolgen der vorgestellten Projekte haben. Hervorheben möchte ich dabei Nils Gehlenborg, Matthias Zschunke, Julian Heinrich, Markus Riester, Marc Röttig, Hannes Planatscher und Daniela Eggle. Diese Liste ist sehr unvollständig und mein Dank gilt einem viel größeren Kreis. Ich empfand die Arbeit mit jungen, talentierten und interessierten Menschen immer als eine persönliche Bereicherung.

Nicht unerwähnt bleiben darf an dieser Stelle das hervorragende Angebot des Universitätssports in Tübingen und die netten Sportskollegen, die der sportlichen auch noch eine soziale Komponente verliehen haben. Ohne diesen Ausgleich wäre ich sicher ein regelmäßiger Gast in den verschiedensten Arztpraxen.

Nicht zuletzt gehört mein Dank auch der großen und unermüdlichen Gemeinde an Entwicklern von freier Software, ohne die diese Arbeit so nicht hätte entstehen können. Schon allein dieses Schriftstück ist unter Linux in Kile geschrieben, mit LaTeX 2_ε und KOMA-Script gesetzt und mit SVK und Subversion versioniert worden – nur um einige Beispiele zu nennen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

Listings

Teil I

Kurze Einführung in die Grundlagen der Microarraytechnologie

1 Etablierte Microarray-Technologien für die Genexpressionsanalyse

> Far better an approximate answer to the right question, than the exact answer to the wrong question, which can always be made precise.
>
> *(J. W. Tukey)*

In den letzten Jahren haben Chip- bzw. Array-basierte Analyseverfahren die Forschung in der Biologie und den Lebenswissenschaften zu einem wesentlichen Teil mitgeprägt. Weiter vorangetrieben wird die Entwicklung durch Fortschritte bei der Genomsequenzierung auf der einen Seite und der immer weiteren Miniaturisierung der Chiptechnologie auf der anderen Seite. Dabei vergrößert sich nicht nur der mögliche Einsatzbereich der Microarrays, sondern es werden auch ganz neue Array-basierte Technologien jenseits der Genexpressionsanalyse entwickelt, wie zum Beispiel Tiling Arrays für das ChIP-on-Chip-Verfahren oder die Resequenzierung (siehe Kapitel 2).

Ihre große Bedeutung erlangen die Microarray-Verfahren durch die Möglichkeit, simultan innerhalb eines Experiments hunderte bis hunderttausende von Messungen der Anwesenheit bestimmter *Ribonukleinsäure* (RNA)- oder *Desoxyribonukleinsäure* (DNA)-Fragmente auf einem Chip durchzuführen. Damit stellen sie ein relativ kostengünstiges Hochdurchsatzverfahren für die moderne Molekularbiologie dar.

Andererseits machen sie durch die Fülle an produzierten Daten, ihrer Komplexität und der Vielzahl an potentiellen Fehlerquellen des Prozesses eine strikte Anwendung mathematisch-statistischer und informationstechnischer Verfahren und Methoden notwendig [Cob06]. Sie bilden damit im Schnittfeld von Biologie, Statistik und Informatik einen Kernanwendungsbereich der Bioinformatik.

1.1 Grundlagen

Zwei wichtige Grundsätze der Molekularbiologie bilden die Grundpfeiler der Genexpressionsanalyse mit Hilfe von Microarrays. Dies ist einerseits die spezifische Basenpaarung der Nukleotide Adenin `A`, Thymin/Uracil `T`/`U`, Guanin `G` und Cytosin

`C` – der sogenannten Watson-Crick-Paarung (siehe Abb. 1.1), die für eine mehr oder weniger exakte Detektion bestimmter Sequenzen in der zu untersuchenden Probe notwendig ist. Neben den verbreiteten Watson-Crick-Paarungen kommen in geringerem Umfang auch ungewöhnliche Paarungen vor, wie `G-U`, `G-T` und `A-C`.

Den anderen Grundsatz bildet das zentrale Dogma der Molekularbiologie, wonach Sequenzinformationen in lebenden Zellen im Wesentlichen von der im Kern befindlichen DNA auf mRNA umgeschrieben (Transkription) und dann zu den Ribosomen transportiert wird, um dort von diesen mit Hilfe von tRNA letztendlich in Proteine übersetzt (Translation) zu werden (siehe Abb. 1.2). Abbildung 1.3 stellt alle bisher im Labor oder in der Natur gefundenen Möglichkeiten der Umwandlungen dar.

Aus dem zentralen Dogma der Molekularbiologie folgt eine Proportionalitätsrelation zwischen der Anzahl an vorliegenden mRNA-Transkripten q_{mRNA} und der Entstehungsrate q_{Pro} der kodierten Proteine:

$$q_{\mathrm{mRNA}} \sim q_{\mathrm{Pro}} \tag{1.1}$$

Somit kann die Bestimmung von q_{mRNA} als indirektes Maß für die Expressionsrate eines Gens in der Zelle dienen.

Als Ausgangspunkt und historisch als eines der ersten Genexpressionsarrays kann der 1975 von E.W. Southern entwickelte *Southern-Blot* (S. 498 in [AJL+02], [Sou75]) angesehen werden [Lan99a], bei dem eine *Gelelektrophorese* zur Auftrennung von DNA-Fragmenten mit der spezifischen Hybridisierung farbstoffmarkierter, komplementärer DNA-Einzelstränge kombiniert wird. Verfahrenstechnisch werden dabei die elektrophoretisch aufgetrennten Targetfragmente – DNA beim Southern-Blot oder RNA beim *Northern-Blot* – direkt vom Gel auf eine Nitrozellulose-Membran übertragen („blotting“). Anschließend erfolgt dort die Hybridisierung mit radioaktiv oder Fluoreszenzfarbstoff markierten, komplementären DNA-Sonden, deren genaue Position dann im Elektrophorese-Abdruck auf der Nitrocellulose-Membran visualisiert werden kann.

Traditionelle Methoden zur Messung der Genexpression, wie sie auch der Northern-Blot darstellt, können nur einen sehr begrenzten Umfang an Gensequenzen gleichzeitig untersuchen. Die Weiterentwicklung des technologischen Grundgedankens bot jedoch das Potential die Untersuchungspalette sehr zu erweitern. Es führte über das Hybridisieren von mRNA gegen ganze Klon- bzw. *komplementäre DNA* (cDNA)-Bibliotheken mittels auf Nylonfiltern fixierten cDNA-Sondenmolekülen zur modernen Microarraytechnologie in ihren zwei wesentlichen Ausprägungen, den gespotteten Arrays (Abschn. 1.3) und den In Situ-Arrays (Abschn. 1.4). Wesentliches Merkmal und damit Grundlage für eine Art einfacher Definition stellt damit die Fixierung der Sondenmoleküle auf einer Oberfläche dar:

Definition 1 (Microarray) *Unter Microarrays versteht man eine hochdichte Fixierung von Nukleotid-Sonden an einer festen Oberfläche, von deren regulärer Anordnung auf die jeweilige Sonde geschlossen werden kann. Detektionsmechanismus ist die spezifische, komplementäre Watson-Crick-Basenpaarung bei der Hybridisierung der Sondenmoleküle mit den zugehörigen Target-Nukleotidsequenzen des zu untersuchenden Stoffgemisches.*

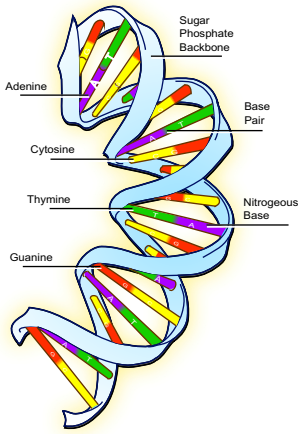


Abbildung 1.1: DNA-Doppelhelix mit Watson-Crick-Paarungen (`A-T`, `C-G`) (Quelle [Wan08])

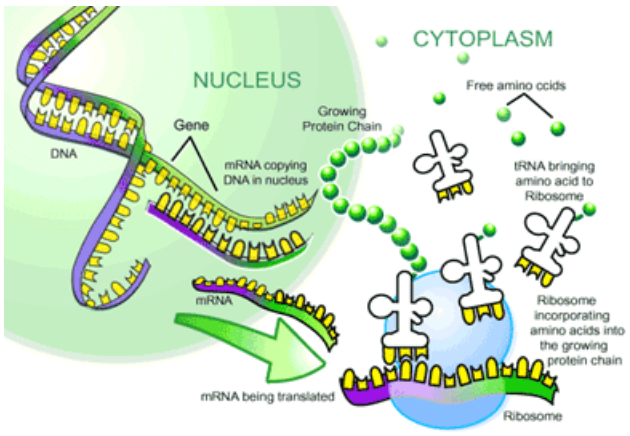


Abbildung 1.2: Transkription und Translation in der Zelle (Quelle [Wan08])

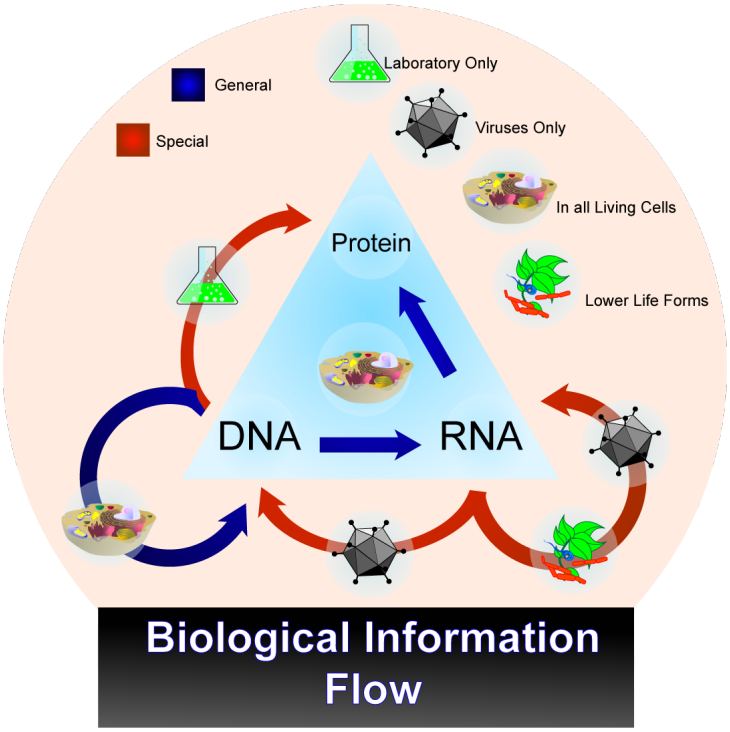


Abbildung 1.3: Darstellung des zentralen Dogmas der Molekularbiologie (Quelle [Jon06]). Dargestellt sind die allgemein zutreffenden Wege in Blau und spezielle, bei bestimmten Organismen oder im Labor vorkommende bzw. induzierbare in Rot.

Diese Definition schließt auch die besondere Form der Informationskodierung ein, wie sie bei den Bead-Arrays von Abschnitt 1.5 vorgenommen wird.

1.2 Prinzipieller Ablauf eines Microarrayexperiments zur Genexpressionsanalyse

Unter der in Definition 1 verallgemeinerten Grundannahme für Microarrays ergibt sich auch ein allgemeiner Ablauf bzw. ein verallgemeinertes Protokoll für ein Genexpressionsexperiment mit Hilfe von Microarrays. Spezifische Besonderheiten, die der jeweils verwendeten Array-Technologie geschuldet sind, werden erst in den zugehörigen Abschnitten 1.3, 1.4 und 1.5 angesprochen.

1. **Festlegung der durch die Genexpressionsanalyse zu beantwortenden biologischen oder medizinischen Fragestellung**

2. **Auswahl oder Entwurf des verwendeten Microarrays**
   Für die Planung eines Genexpressionsexperiments ist, nach der Festlegung der durch dieses Experiment zu beantwortenden Fragestellung, die Auswahl bzw. der Entwurf des zu verwendenden Arrays der erste Planungsschritt. Insbesondere ist dabei die Auswahl der für das Array genutzten Technologie wichtig, werden beispielsweise vorgefertigte Arrays von Anbietern wie Affymetrix [Aff08b] oder Illumina [Ill08b] genutzt, für die bereits ein komplettes Auswertungsprotokoll vorliegt, oder werden sogenannte *Customized-Arrays* verwendet. Customized-Arrays werden in Zusammenarbeit mit den Anbietern entsprechend den Wünschen des Experimentators entworfen. Eine weitere Möglichkeit besteht in der Herstellung des Arrays gänzlich in Eigenregie des Experimentators, z.B. für ein Spotted Array.

   Der eigene Arrayentwurf stellt hohe Ansprüche an das Vorhandensein von Erfahrung und Wissen. Unbedingt müssen potentielle Fehlerquellen beachtet und unter statistischem Gesichtspunkt unter Kontrolle gehalten werden. Beispiele dafür sind vorgesehene technische Replikate (Spot-Replikate) auf dem Chip und deren zufällige Verteilung über die Chipoberfläche, sowie verschiedene Kontroll- und Leerspots.

3. **Entwurf des Genexpressionsexperiments**
   Ein weiterer wichtiger Planungsschritt im Vorfeld des eigentlichen Experiments ist die konkrete Versuchsplanung unter den Randbedingungen der ersten beiden genannten Schritte. Hierbei ist bereits die Beachtung der anzuwendenden Analyseverfahren für die Versuchauswertung nötig. Typische Fragestellungen dieser Phase wären beispielsweise:

   - Zellen welchen Zustandes, Gewebes oder Zeitpunktes sind zu untersuchen?
   - Wie gewinne ich die entprechenden Zellen am günstigsten?

- Gibt es bereits Protokolle für die Extraktion und Behandlung der mRNA?
- Wie vergleiche ich die Genexpression zwischen den einzelnen Proben in der günstigsten Art und Weise? Dabei werden die Auswahlmöglichkeiten durch die biologische Fragestellung und die verwendete Technologie eingeengt. In einem zweikanaligen Experiment lassen sich binäre Vergleiche direkt durch konkurrierende Hybridisierung auf dem gleichen Array durchführen. Dadurch lassen sich mehrere technische Fehlerquellen besser kontrolliert (Blockbildung). Sind aus technischen Gründen oder der Notwendigkeit höherer Flexibilität bei der Auswertung indirekte Vergleiche notwendig, ist der allgemeine Standard (Baseline) für alle Vergleiche festzulegen.

4. **Produktion oder Beschaffung des Arrays**

5. **Gewinnung der Proben**
   Dieser Schritt umfasst die Züchtung, Entnahme oder Behandlung von Zellen zur Gewinnung biologischen Probenmaterials, das in der Lage ist, die dem Experiment zugrundeliegende Frage zu beantworten.

6. **Vorbereitung der Targetfragmente (cDNA/cRNA)**
   - Extraktion der mRNA aus den Proben
   - Aufreinigung
   - Umschreibung in cDNA und cRNA
   - Markierung mit Fluoreszenzfarbstoffen

7. **Hybridisierung und Reinigung**
   Für die Hybridisierung wird die in Schritt 6 gewonnene Lösung zusammen mit den Microarrays unter definierten Bedingungen inkubiert. Anschließend werden noch ungebundene cDNA- und cRNA-Fragmente abgewaschen.

8. **Auslesen der Intensitätswerte des Arrays im Scanner**
   In diesem Schritt erfolgt das Auslesen der Intensitätswerte für jedes einzelne Pixel des Arrayabbildes mit Hilfe eines Laserscanners. Der Laser wird dabei auf das jeweilige „Pixel“ positioniert und regt dort – falls vorhanden – den Farbstoffanteil der gebundenen, markierten Targetfragmente zur Fluoreszenz auf der charakteristischen Spektrallinie an. Dies entspricht dem jeweiligen (Farb-)Kanal. Die ausgelesenen Intensitätswerte für das „Arraypixel“ bilden den Ausgangspunkt für die weitere Analyse und werden gespeichert.

9. **Bildanalyse**
   Im Rahmen der Bildanalyse wird ein Schätzer $\hat{I}_{\text{cDNA},Ch_x}$ für den Intensitätswert eines Spots (entspricht einem bestimmten cDNA- bzw. cRNA-Fragment) im Farbkanal Ch_x ermittelt. Dazu notwendig sind die Detektion der Spots, das

Gridding und die Anwendung statistischer Schätzverfahren, um aus der Menge der gefundenen Pixel eines Spots $\hat{I}_{\text{cDNA},Ch_x}$ des jeweiligen Farbkanals Ch_x zu berechnen.

10. **Grundlegende Expressionsanalyse**
    $\hat{I}_{\text{cDNA},Ch_x}$ steht in direkter Relation zu q_{mRNA} (Formel 1.1) des Zelltyps C_x, die mit dem Farbstoff des Kanals Ch_x markiert wurde:

$$\hat{I}_{\text{cDNA},Ch_x} \sim q_{\text{mRNA},C_x} \tag{1.2}$$

    Die $\hat{I}_{\text{cDNA},Ch_x}$ sind deshalb als Entsprechung für r_{mRNA,C_x} verwendbar und können insbesondere in der Form relativer Verhältnisse r_{cDNA} (Ratio) zwischen verschiedenen Zelltypen C_i und C_j

$$r_{\text{cDNA}} = \frac{\hat{I}_{\text{cDNA},Ch_i}}{\hat{I}_{\text{cDNA},Ch_j}} \sim \frac{q_{\text{mRNA},C_i}}{q_{\text{mRNA},C_j}} \tag{1.3}$$

    bzw. als logarithmiertes Verhältnis l_{cDNA} (Logratio)

$$\begin{aligned} l_{\text{cDNA}} &= \log\left(\frac{\hat{I}_{\text{cDNA},Ch_i}}{\hat{I}_{\text{cDNA},Ch_j}}\right) = \log \hat{I}_{\text{cDNA},Ch_i} - \log \hat{I}_{\text{cDNA},Ch_j} \\ &\sim \log\left(\frac{q_{\text{mRNA},C_i}}{q_{\text{mRNA},C_j}}\right) = \log q_{\text{mRNA},C_i} - \log q_{\text{mRNA},C_j} \end{aligned} \tag{1.4}$$

    Aussagen über deren differentielle Expression von Genen ermöglichen.

    Voraus geht weiteren statistische Analyseverfahren eine *Normalisierung* (vgl. [GCH+05, BIÅS03, Cle79, YBDS02] und Abschnitt 7.2.5 auf Seite 160), die die gemessenen Rohwerte bezüglich bekannter bzw. erwarteter systematischer Fehlerquellen rechnerisch korrigiert, beispielsweise eine unterschiedliche Inkorporationsgeschwindigkeit der genutzten Farbstoffe oder ortsabhängige Intensitätsabweichungen.

11. **Expressionsanalyse auf höherer Ebene**
    Mit den in Schritt 10 gewonnenen Daten lassen sich umfangreiche Analysen durchführen. Auf dieser Ebene geht es darum, geeignete Verfahren einzusetzen, die eine valide Beantwortung der in Schritt 1 formulierten Fragestellung erlauben. Oft geht es dabei um die Entdeckung von Abhängigkeiten zwischen der Aktivität einzelner Gene und deren Bezug zum beobachteten Phänotyp.

    Die angewendeten Verfahren sind sehr vielfältig und stammen aus Gebieten wie der Statistik, dem Machine Learning und der Visualisierung.

1.3 Spotted Arrays

Spotted Microarrays stellen eine der beiden am weitesten verbreiteten technologischen Varianten der Array-Technologie dar. Als ein wesentlicher Meilenstein ihrer

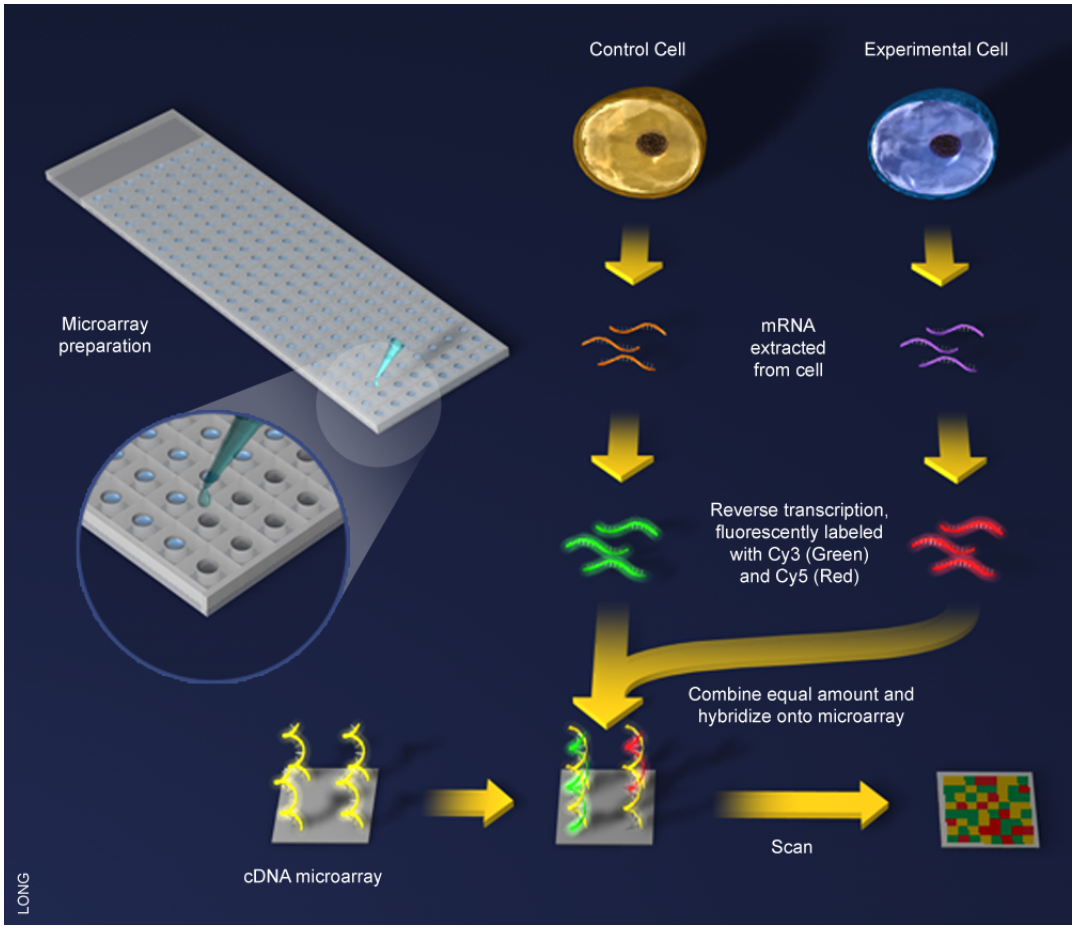


Abbildung 1.4: Typischer Ablauf eines zweikanaligen Genexpressionsexperimentes mit Hilfe eines Spotted Array. (Quelle [Lon08])

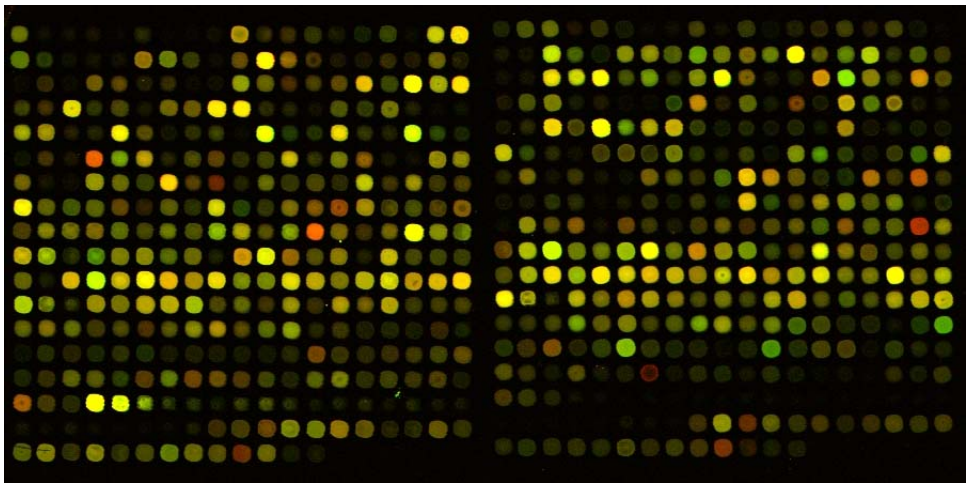


Abbildung 1.5: Rot-Gründarstellung des Laserscans eines gespotteten Arrays. (Quelle [Man08])

Anwendung gilt allgemein die Veröffentlichung von Schena et al. im Jahr 1995 [SSDB95]. Dort untersuchten die Autoren 45 cDNA-Sequenzen (14 komplette Gene und 31 *Expressed Sequence Tag*s (EST)) aus *Arabidopsis thaliana*.

Abbildung 1.4 zeigt einen typischen Abblauf eines zweikanaligen Experiments mit einem Spotted Array, wie es auch in [SSDB95] präsentiert wird. Vor Beginn des Experiments wird das Array gefertigt. Hierzu wird mit einem automatischen Spotroboter das Sondenmaterial auf den Glasträger aufgebracht und auf seiner Oberfläche fixiert. Das Sondenmaterial bestand früher hauptsächlich aus cDNA einer Basenlänge von mehreren hundert bis tausend Nukleotiden und wurde oft im versuchsdurchführenden Labor selbst gewonnen. In den letzten Jahren überwiegt die Verwendung sogenannter „Long-Oligos“ mit Basenlängen bis zu 80 Nukleotiden, die, im Falle eines eigenen Array-Entwurfs, häufig in enger Kooperation mit kommerziellen Anbietern entworfen und dann von diesen bezogen werden.

Entsprechend dem in Abschnitt 1.2 dargestellten allgemeinen Versuchsablaufs wird das mRNA-Material zweier zu vergleichender Zelltypen C_i und C_j aufbereitet, in cDNA umgeschrieben und beispielsweise jeweils mit den sehr gebräuchlichen Farbstoffen Cy3 (besitzt als Substanz eine rosa Färbung) und Cy5 (besitzt als Substanz eine blaue Färbung) markiert, die in unterschiedlichen Spektralbereichen fluoreszieren – Cy3 im grünen bis hellen roten und Cy5 im roten bis dunklen roten. Danach erfolgt die kompetetive Hybridisierung auf dem Array. Blendet man die Bilder der beiden gemessenen Farbkanäle in den Farben grün (Cy3-markiert) und rot (Cy5-markiert) übereinander, so erscheinen die Spots mit gleichmäßig hoher Expressionsstärke in den Zelltypen C_i und C_j als gelb - d.h. $r_{\text{cDNA}} = 1 \Leftrightarrow l_{\text{cDNA}} = 0$ (vgl. Gleichung 1.4), während sich die überwiegende Expression in nur einem Kanal in der entsprechend intensiven farblichen Tönung des Spots bemerkbar macht (siehe Abbildung 1.5).

1.4 In Situ-Arrays

Die andere weit verbreitete technologische Variante der Microarrays verwendet eine Adaption des photolithographischen Prozesses, der in der Halbleiterindustrie zur Herstellung der sehr kleinen Halbleiterstrukturen auf einem elektronischen Chip genutzt wird. Eng verbunden ist die Entwicklung dieses Verfahrens mit den Arbeiten von S. Fodor und seinen Kollegen [FRP+91] und der von Fodor mitgegründeten Firma Affymetrix [Aff08b]. Erste erfolgreiche Anwendungen [LDB+96] demonstrierten das große Potential der auf diese Art produzierten hochdichten Microarrays.

Ein großer Vorteil dieses Verfahrens ist eine weitaus höhere Miniaturisierbarkeit der verwendeten Strukturen als bei den gespotteten Arrays, so dass eine sehr hohe Dichte der Sonden erreicht werden kann, zum Beispiel 1.8 Millionen auf dem Affymetrix Genome-Wide Human SNP Array 6.0 [Aff07]. Als Sonden werden Oligonukleotide einer Länge von 25 (Affymetrix - siehe Bild 1.6) bis ca. 60 Nukleotiden (NimbleGen [Nim08a]) verwendet.

Die Oligonukleotide werden in dem in Abbildung 1.6 dargestellten photolitogra-

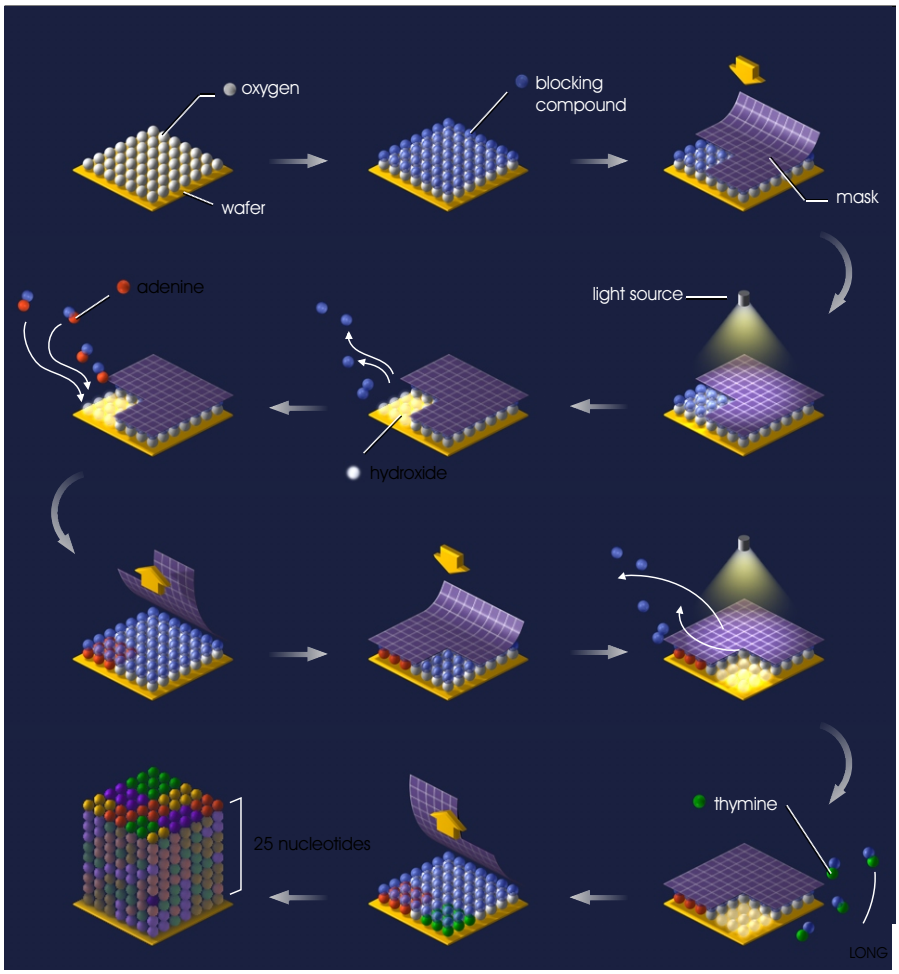


Abbildung 1.6: Schematisierung des photolithographischen Produktionsprozesses eines In Situ-Arrays der Firma Affymetrix. (Quelle [Lon08])

Abbildung 1.7: In Situ-Microarrays der Firma Affymetrix. (Quelle [Sch06])

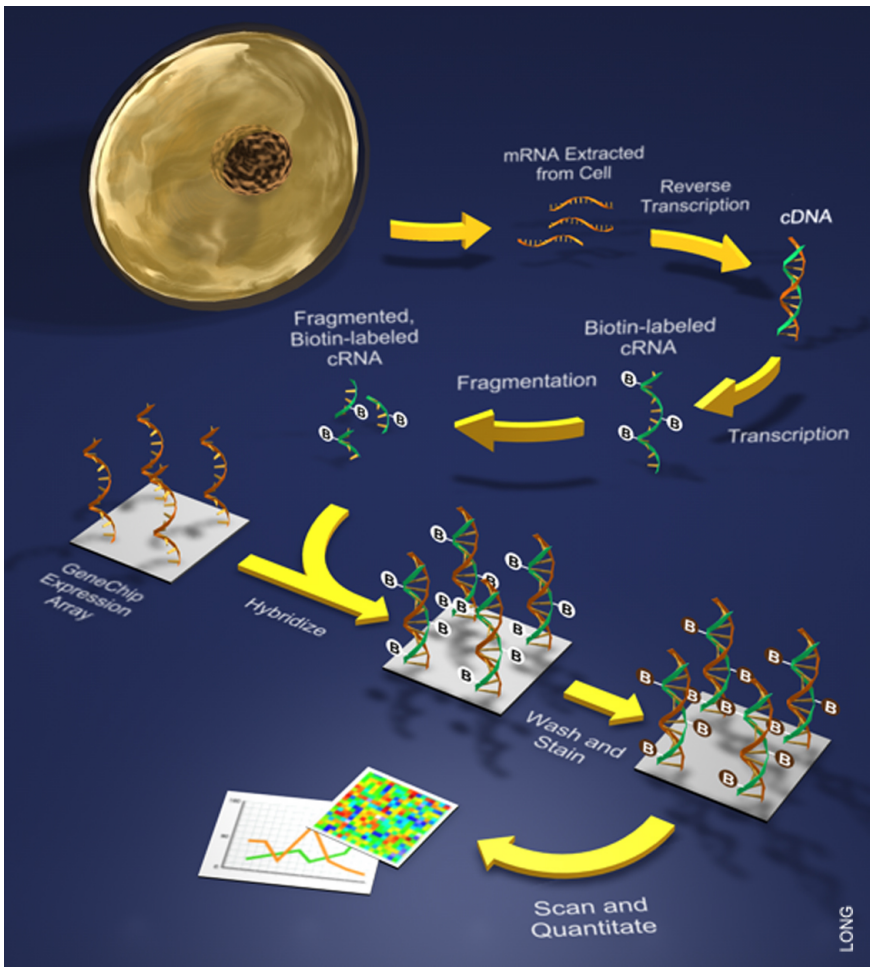


Abbildung 1.8: Typischer Ablauf eines einkanaligen Genexpressionsexperiments mit einem In Situ-Array. (Quelle [Lon08])

phischen Prozess in einzelnen Schritten auf der Chipoberfläche synthetisiert. Dabei muss für jede Oligoposition jeweils viermal ein lichtempfindlicher Schutzlack aufgetragen, dieser dann mit einer strukturabbildenden Maske beleuchtet, gewaschen und an die dadurch zugänglich gewordene Stellen die jeweilige Base `A`, `T`, `C` oder `G` kovalent gebunden werden. Das bedeutet, dass für eine Sondenlänge der Oligos von N mindestens $4 \times N$ technologische Schritte notwendig sind. Bei nicht so hochdichten Arrays ist die Verwendung des kostengünstigeren und flexibleren maskenlosen Prozesses von NimbleGen [Aff08a] möglich, der die Strukturbelichtung statt mit festen Masken mit Hilfe eines *Digital Micromirror Device* (DMD) durchführt.

Nach der Fertigstellung der Arrays werden diese, ähnlich ihren elektronischen Pendants, in passende Gehäuse verpackt. Diese weisen eine Reaktionskammer für die Hybridisierung und ein Fenster zum Auslesen durch den Laserscanner auf. Abbildung 1.7 zeigt zwei Beispiele für GeneChip-Arrays der Firma Affymetrix.

Abbildung 1.8 stellt den Ablauf eines typischen einkanaligen Genexpressionsex-

periments mit einem In Situ-Array dar. Im Unterschied zum zweikanaligen Experiment von Abschnitt 1.3 müssen für einen Vergleich zwischen zwei unterschiedlichen Zelltypen bzw. Zuständen von Zellentypen C_i und C_j zwei getrennte Hybridisierungen auf jeweils einem Array durchgeführt werden. Die Ermittlung von Differenzen muss dann als rechnerischer Vergleich der beiden Arrayscans erfolgen.

1.5 Bead-Arrays

Eine weitere Form eines Microarrays stellt das Bead-Array dar, das in den letzten Jahren unter der Marke *BeadChip* von der Firma Illumina entwickelt und auf den Markt gebracht wurde [KBC+04]. Hierbei handelt es sich um einen dritten technologischen Weg, der Methoden der anderen beiden Technologien in alternativer Form verbindet. Kennzeichnend für die Bead-Arrays ist die Synthese von Oligonu-

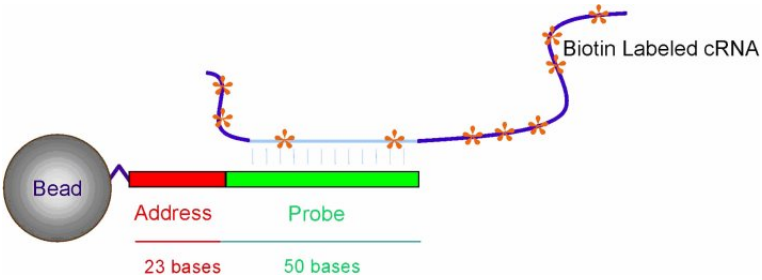


Abbildung 1.9: Schematische Darstellung eines an einem Bead fixierten Oligonukleotides (Quelle [DKF07]). Das Oligonukleotid besteht aus einem Adressteil, der zur Identifikation des zweiten, eigentlichen Sondenanteil des Oligos dient [GKG+04]. Aktuelle BeadChips weisen einen Adressteil von 29 Nukleotiden Länge auf. In der dargestellten Abbildung ist noch eine frühere Version mit nur 23 Nukleotiden Länge zu sehen.

kleotiden als Sondenmaterial, die an einzelnen Partikeln (Beads) befestigt werden [SST+04] - ähnlich den Spotted Arrays. Die Beads wiederum werden zufällig über die Oberfläche eines Trägermaterials, z.B. ein Glasträger oder ein Siliziumchip (vgl. Abbildung 1.10), verteilt und in photolithographisch hergestellten Ausbuchtungen auf ihm fixiert.

Abbildung 1.9 stellt ein solches an einem Bead fixiertes Oligo dar. Jedes Bead enthält viele Kopien ein und desselben Oligos. Ein Bead stellt somit die Entsprechung zu einem Spot auf einem herkömmlichen Array dar. Das Oligo besteht aus einem bei aktuellen Chips 29 Nukleotiden langen Adressteil, der zur Identifikation der Sequenz des eigentlichen Sondenanteils des Oligos benutzt wird. Nur der 50 Nukleotide lange Sondenanteil des Oligos interagiert bzw. hybridisiert mit dem markierten Target aus der Probe. Der Adressteil ist im Hinblick auf eine eindeutige, unverwechselbare Sequenz entworfen worden und bestimmt mit seiner Länge wie viele Oligos überhaupt identifizierbar sind.

Da die Beads zufällig über den Träger verteilt werden, muss der Adressteil der Oligos ausgewertet bzw. dekodiert werden [GKG+04], um eine Zuordnung von der

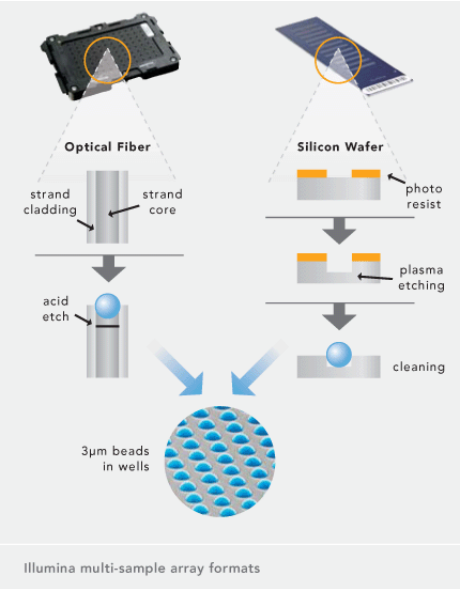


Abbildung 1.10: Technologische Varianten der Bead-Arrays von Illumina: Links als *Sentrix Array Matrix* (SAM) bei der die Beads auf einer Glasfaserstruktur befestigt sind und rechts als BeadChip (Quelle [Ill08b])

Position auf dem Array zur konkreten Sonde herzustellen. Diese Zuordnung ist für jedes Array einmalig und wird nach dem Dekodieren in einer dem Array zugehörigen Datei abgespeichert. Diese Datei, d.h. die in ihr enthaltene Dekodierung, benötigt man später beim Auslesen der Arrays mit dem Laserscanner.

Die Beads eines Oligotyps kommen auf einem Array mit einer hohen Replikationsquote von 30 vor [Ill08a], was die Validität der Messung steigert. Ein weiterer aus statistischer Sicht vorteilhafter Aspekt der Bead-Arrays ist die vollkommen zufällige, von Array zu Array variierende Verteilung der Beads über das Array. Dies sorgt, in Kombination mit der hohen Rate an Bead-Replikaten und replizierten Arrays, für ein Herausmitteln möglicher ortsabhängiger Verzerrungen bei den Messungen. Zusätzlich steigt so auch die Robustheit gegenüber etwaigen zufälligen Beschädigungen in bestimmten Oberflächenbereichen des Arrays während des Ex-

Abbildung 1.11: Die beiden aktuellen BeadChips für die Untersuchung der Genexpression beim Menschen, Human6-WG v3.0 und HumanRef-8 v3.0 (Quelle [Ill08b])

periments.

Die beiden aktuellen BeadChips (siehe Abbildung 1.11) für die Analyse der Genexpression beim Menschen, Human6-WG v3.0 und HumanRef-8 v3.0, untersuchen 48804 bzw. 24526 Features [Ill08a]. Berücksichtigt man hierbei noch die große Anzahl an internen Replikaten, die pro Feature verwendet werden, so stellt die Bead-Array-Technologie in Bezug auf die hohe Dichte eine der In Situ-Technologie vergleichbare Technik dar.

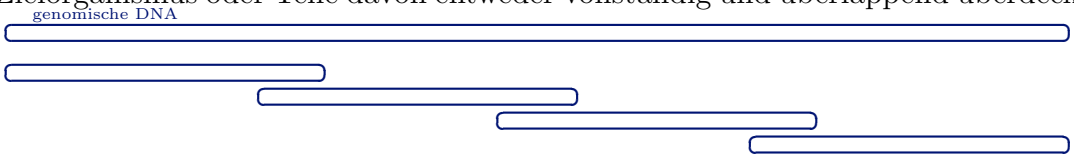
2 Microarray-basierte Verfahren jenseits der Genexpressionsanalyse

> Natural selection is a mechanism for generating an exceedingly high degree of improbability.
>
> *(R. A. Fisher)*

Die bereits eingangs des Kapitels 1 genannten großen Vorteile von Microarrays als kostengünstiges Hochdurchsatzverfahren für den gesamten Life-Science-Bereich forcierte die Entwicklungen einer ganzen Reihe weiterer chipbasierter Verfahren neben der etablierten Genexpressionsanalyse [Hoh06], so zum Beispiel für das Genotyping (*Single-Nucleotide Polymorphism* (SNP)-Arrays), für die genomweite epigenetische Analyse (ChIP-on-Chip) oder die Analyse von Splice-Varianten. Ziel der Analyse bleibt dabei immer die Verknüpfung der genomweiten Variation auf verschiedenen molekularbiologischen Ebenen mit den beobachtbaren Variationen im Phänotyp herzustellen. Gegenstand dieses Kapitels soll die kurze Vorstellung einiger dieser noch nicht so breit eingesetzten neueren Verfahren sein.

2.1 Tiling Arrays

Viele der weiteren Anwendungen von Microarrays beruhen auf dem Einsatz eines sogenannten Tiling Arrays. Bei Tiling Arrays wird das Sondenmaterial der einzelnen Features aus Sequenzabschnitten gebildet, die die gesamte genomische DNA des Zielorganismus oder Teile davon entweder vollständig und überlappend überdecken:



oder nicht vollständig:

Arrays des zweiten Typs werden häufig für Untersuchungen des gesamten Genoms benutzt, zum Beispiel in Form des im nächsten Abschnitt 2.1.2 vorgestellten SNP-Arrays. Die Variante des ersten Typs findet aufgrund der dafür nötigen hohen

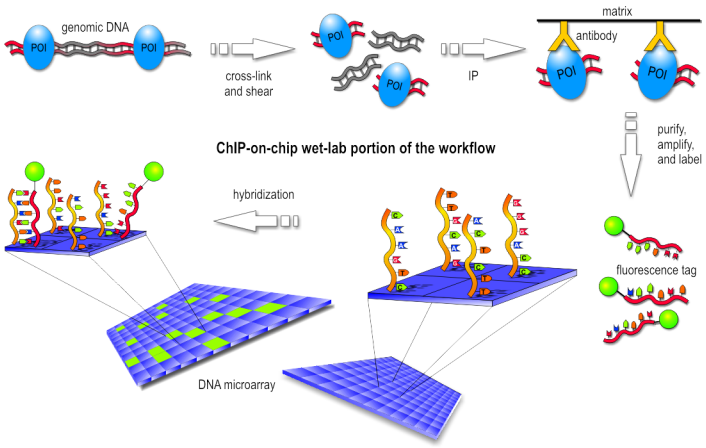


Abbildung 2.1: Schematischer Überblick über den experimentellen Ablauf eines ChIP-on-Chip Experiments für ein interessierendes Protein – *Protein of Interest* (POI) (Quelle [Hen07])

Sondendichte häufig bei konkreteren Follow-Up-Untersuchungen von Teilabschnitten Anwendung, zum Beispiel zur Identifikation chromosomaler Breakpoints.

Neben den in diesem Abschnitt beschriebenen wichtigen Formen von Tiling Arrays gibt es noch weitere Varianten, wofür aber auf die weitergehende Literatur verwiesen wird [Hoh06, Liu07].

2.1.1 ChIP-on-Chip

Eine sehr gute Methode zur Untersuchung genregulatorischer Proteine, wie Transkriptionsfaktoren oder Proteine für die Regulierung der DNA-Replikation, stellt die *Chromatin-Immunopräzipitation* (ChIP) dar. Das Besondere der Methode ist, dass sie die empirische Bestimmung der aktuellen DNA-Bindestellen eines Proteins für eine lebende Zelle unter definierbaren Bedingungen gestattet (vgl. S. 394f in [AJL+02]). Das heißt, mit Hilfe der ChIP erhält man einen sehr direkten „In-Vivo-Einblick“ in das genregulatorische Geschehen dieses Proteins.

Der Ablauf einer ChIP ist im ersten Teil der Abbildung 2.1 dargestellt. Die der DNA angelagerten, interessierenden Proteine werden innerhalb der lebenden Zellen mit kovalenten Bindungen fest an die Chromatin-Basis gebunden (*Cross-Linking*). Nach dem Aufschließen der Zellen erfolgt die Zerlegung der DNA-Protein-Komplexe in kleinere Fragmente ($\sim$ 300 Nukleotide). Die gesuchten DNA-Protein-Fragmente werden aus dieser Mischung mit Hilfe von oberflächenfixierten, spezifischen Antikörpern für das interessierende Protein gewonnen (*Immunopräzipitation*). Am Ende des nächsten Schrittes, der die Auflösung der kovalenten Bindungen zwischen der DNA und dem Protein umfasst, liegen die speziellen Bindestellen des Proteins als DNA-Fragmente vor. Nach einer Amplifizierung können diese Sequenzen bestimmt werden. Für diesen letzten Schritt besteht die Möglichkeit, eine PCR mit spezifischen DNA-Primern durchzuführen oder ein entsprechendes Tiling Array zu

benutzen. Letzteres stellt das ChIP-on-Chip-Verfahren dar [HL04]. Aus den verschiedenen identifizierten Bindungsstellen lassen sich dann mit bioinformatischen Werkzeugen die für Biologen sehr interessanten sequenz-spezifischen Bindungsmotive des interessierenden Proteins berechnen.

Arrays für die ChIP-on-Chip-Analyse sind bei allen führenden kommerziellen Anbietern beziehbar [Aff08b, Nim08b, Agi08].

Eine spezielle Form des ChIP-on-Chip stellt MeDIP-Chip dar, eine Methyl-DNA-Immunopräzipitation mit einer anschließenden Microarrayhybridisierung. Verwendet wird hierbei ein spezifischer Antikörper gegen 5-Methylcytosin, um direkt methylierte DNA zu präzipieren. Der Methylierungszustand von DNA ist sehr interessant, weil er die Transkription von Genen regulieren kann.

2.1.2 SNP-Arrays

Bei SNPs handelt es sich um die Variation einer einzelnen Base an einer bestimmten Stelle des Genoms einer Spezies, die in einem nennenswerten Anteil der Population vorkommt (vgl. S. 464 in [AJL+02]). Sie sind für den größten Teil des Polymorphismus einer Art, d.h. ihrer genetischen Varianz, verantwortlich – mit ca. 83.6 % der gesamten genetischen Variation in der Genexpression [SFD+07]. Für eine Aufklärung des Zusammenhangs zwischen genetischen Varianten und auftretenden phänotypischen Merkmalen, wie der Entwicklung von Krankheiten, der Verstoffwechselung bzw. Wirkung von Giften und Medikamenten oder die Reaktion auf bestimmte Umweltbedingungen, ist eine genaue Kenntnis der vorliegenden SNPs notwendig (*Genotyping*). Darauf aufbauend lassen sich aus dem Vorhandensein charakteristischer SNPs im Einzelfall diagnostische Aussagen gewinnen, womit diesem Werkzeug ein großes Potential für den Einsatz in der klinischen Praxis zuzurechnen ist.

Das internationale HapMap Konsortium identifizierte über 3.1 Millionen verschiedene SNPs in 270 Individuen aus 4 geographischen Regionen [CFB+07] und lieferte damit eine Grundlage für die Fertigung von SNP-Arrays. Bei den Arrays selbst handelt es sich um die oben erwähnte zweite Form des Tiling Arrays, bei dem die potentiellen SNP-Bereiche als Arraysonden verwendet werden. Alle großen kommerziellen Anbieter haben SNP-Arrays in ihrem Produktportfolio [Aff08b, Ill08b, Nim08b], wie zum Beispiel das Affymetrix Genome-Wide Human SNP Array 6.0 [Aff07].

2.1.3 Array-CGH

Neben den SNPs ist die *Copy Number Variation* (CNV) für den nächstgrößeren Anteil der genetischen Varianz zwischen den verschiedenen Individuen einer Art verantwortlich – mit ca. 17.7 % der gesamten genetischen Variation in der Genexpression [SFD+07]. Ähnlich den Ausführungen zu SNPs in Abschnitt 2.1.2 ist die CNV für ein einzelnes Individuum sehr charakteristisch und liefert einen Hinweis auf dessen Prädisposition bezüglich bestimmter Krankheiten bzw. phänotypischer Ausprägungen. Somit besitzt die einzelfallbezogene, konkrete Bestimmung der CNV ein

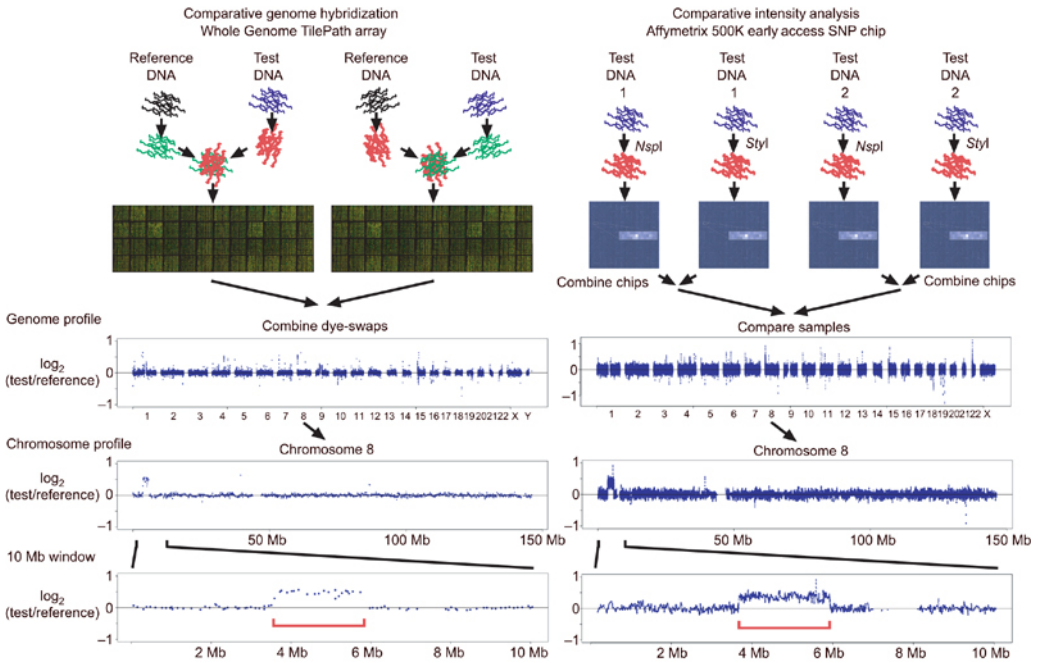


Abbildung 2.2: Array-CGH mit zwei verschiedenen Microarray-Plattformen - einem zweikanaligen Spotted Array, links, und einem einkanaligen In Situ-Array, rechts (Quelle [RIF+06]). Für die Hybridisierung auf Spotted Arrays wurde ein zusätzliches Replikat in Form eines *Dye-Swaps* vorgenommen, um unterschiedlichen Eigenschaften der verwendeten Farbstoffe Rechnung zu tragen. Mit beiden Technologien lässt sich am Ende sehr gut ein großer Bereich mit CNV identifizieren (rote Klammer).

hochinteressantes diagnostisches Potential, das bei geeignet kostengünstigen Verfahren – ähnlich den SNP-Arrays – für den klinischen Alltag sehr wichtig sein kann. So identifizierten Redon et al. 1447 CNV-Regionen im menschlichen Genom [RIF+06].

Technisch lässt sich die Bestimmung der CNV sehr gut mit Hilfe der Microarraytechnologie durchführen. Dabei wird eine *Comparative Genomic Hybridization* (CGH) auf einem Microarray ausgeführt. Bei der CGH wird die genomische DNA der Zellen des zu untersuchenden Kollektivs – Träger eines bestimmten phänotypischen Merkmals oder Erkrankte – und der Kontrollgruppe – Nichtmerkmalsträger oder Gesunde – mit verschiedenen Fluoreszenzfarbstoffen markiert. Anschließend erfolgt die konkurrierende und damit komparative Hybridisierung an einer speziell präparierten Sonden-DNA – im traditionellen Fall einem Präparat der Chromosomen in der Metaphase und bei Array-CGH den Sonden auf einem Microarray.

Abbildung 2.2 zeigt eine Array-CGH für die in [RIF+06] verwendeten zwei Plattformen und ihre anschließende Auswertung. Bei den zweikanaligen Spotted Arrays kann die CGH direkt auf einem einzigen Array ausgeführt werden, während bei den einkanaligen In Situ-Chips das Test- und das Kontroll-Genom jeweils auf einem eigenen Chip hybridisiert und erst danach rechnerisch verglichen wird. Das Ergebnis einer Analyse ist im unteren Teil von Abbildung 2.2 zu sehen. Beide Technologien

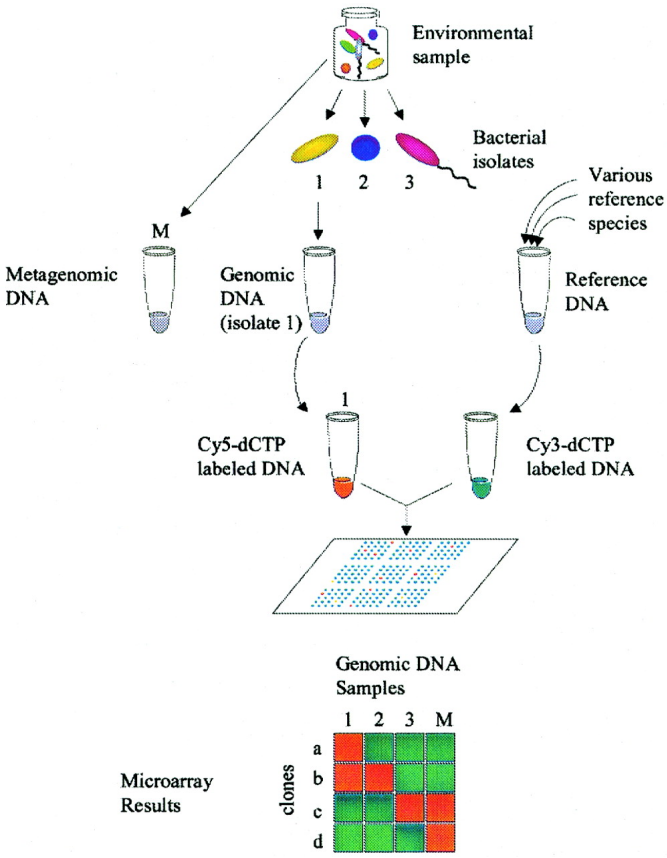


Abbildung 2.3: Metagenomische Analyse (Metagenomic Profiling) einer Umweltprobe mit Hilfe einer Mischung von Vergleichssequenzen (Klon-Bibliothek laborbekannter Stämme) und deren komparativer Hybridisierung auf einem Microarray (Quelle [SCC03]). Jede Hybridisierung vergleicht die DNA erfolgreich kultivierter Stämme der Probe mit den Sequenzen der Klon-Bibliothek – Klon **a** korrespondiert mit Stamm **1**, Klon **b** mit den Stämmen **1** und **2**, Klon **c** mit Stamm **3** und dem Metagenom **M** und schließlich Klon **d** nur mit dem Metagenom **M**.

erlauben eine übereinstimmende Detektion eines $\sim$ 2 Mb großen Bereichs mit einer CNV zwischen den untersuchten Genomen.

Arrays für die Array-CGH finden sich im Angebot aller größeren kommerziellen Hersteller von Microarrays [Aff08b, Ill08b, Agi08, Nim08b].

2.2 (Re)Sequenzierungsarray

Obwohl die Genexpressionsanalyse derzeit den größten Anwendungsbereich der Microarraytechnologie darstellt, wurde es früher vor allem im Zusammenhang mit der Sequenzierung als *Sequencing-By-Hybridization* (SBH) entwickelt [BS88]. Sie galt als eine kostengünstige und aussichtsreiche Sequenzierungsalternative. In Anbetracht der dabei auftretenden technischen und rechnerischen Probleme und den

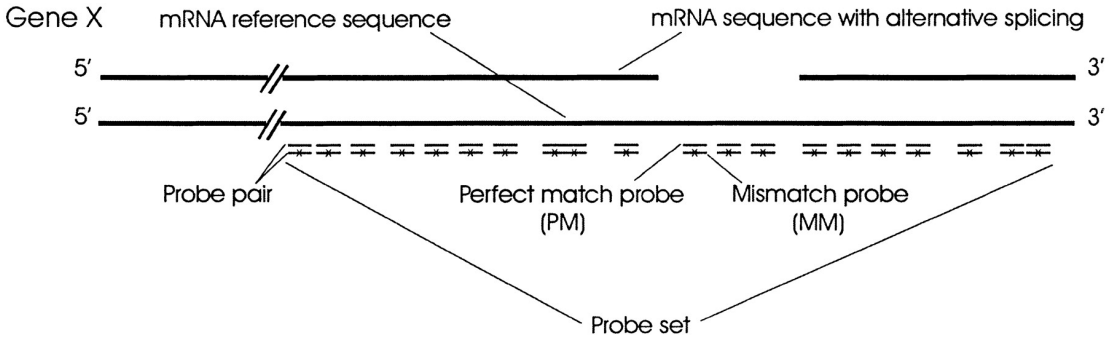


Abbildung 2.4: Prinzipielle Vorgehensweise bei der genomweiten Detektion von Splice-Varianten mit Hilfe von hoch-dichten In Situ-Arrays (Quelle [HMM+01]). Zur Anwendung kommen hier die auf den GeneChips von Affymetrix eingesetzten Paare von *Perfect Match* (PM) und *Mismatch* (MM) Oligomer-Sonden. PMs sind exakt komplementär zur Zielsequenz, während MMs in der zentralen Position eine nicht passende Base besitzen - im Bild durch ein `x` gekennzeichnet. Im oberen Teil der Abbildung befinden sich die beiden alternativen Splice-Varianten der mRNA. Anhand der Auswertung der unterschiedlichen Signalstärken auf der Ebene der einzelnen PM-MM-Paare lässt sich das Vorhandensein verschiedener Splice-Varianten ermitteln.

Fortschritten anderer Sequenzierungstechniken besitzt SBH für die Sequenzierung von Genomen heute praktisch keine große Bedeutung.

Interessant ist die konzeptionell nahegelegene Idee, Microarrays kostengünstig im Rahmen der Metagenomik einzusetzen [SCC03]. Abbildung 2.3 zeigt den Ablauf und die Art einer Array-basierten Profilerstellung für eine Probe aus einem Ökosystem. Der Vorteil metagenomischer Analysen besteht unter anderem darin, auch nichtkultivierbare Arten untersuchen bzw. berücksichtigen zu können und aus den genomischen Daten Charakteristika des Ökosystems abschätzen zu können [TvMK+05].

2.3 Detektion von Splice-Varianten

Eine weitere nützliche Anwendung von Microarrays besteht in der genomweiten Detektion von Splice-Varianten. Alternatives Splicing der prä-mRNA stellt einen weiteren wichtigen Regulationsmechanismus für die Genexpression bei Eukaryonten dar. Diese Regulation kann in Abhängigkeit der Gewebeart oder bestimmter Konditionen variieren, sowie Krankheiten oder phänotypische Merkmale induzieren.

Hierfür eignen sich hochdichte In Situ-Arrays mit entsprechend entworfenen Oligomer-Sonden sehr gut, insbesondere die Tiling Arrays. Abbildung 2.4 zeigt das Prinzip der arraybasierten Detektion von alternativen Splice-Varianten mit Hilfe eines kundenspezifischen GeneChips der Firma Affymetrix [HMM+01]. Die Oligomer-Sonden werden günstigstenfalls entlang der Sequenz der prä-mRNA plaziert. Die Auswertung zur Detektion von alternativen Splice-Varianten muss auf der Ebene der einzelnen Oligomer-Sonden erfolgen, die alternatives Splicing durch entspre-

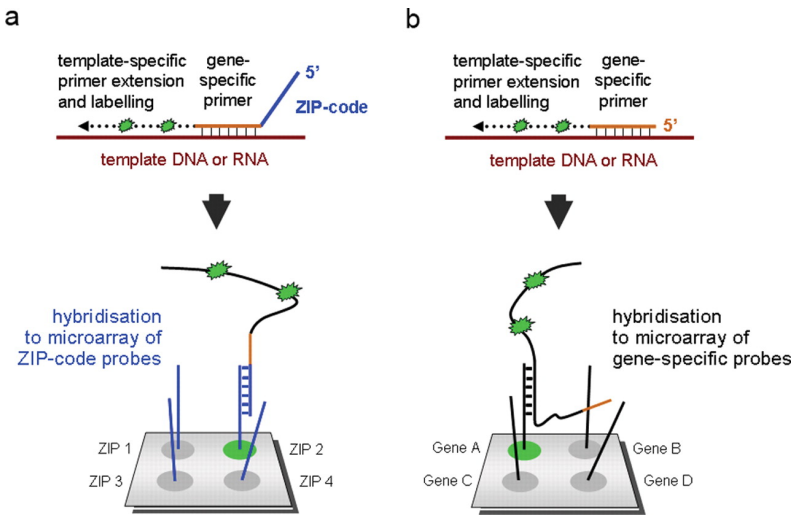


Abbildung 2.5: Vergleich zwischen einem ZIP-Code-Array **(a)** und einem normalen Microarray **(b)** (Quelle [HMJ+06]). Während beim ZIP-Code-Array die eigentliche Detektions- mit der Dekodierungsstufe über das ZIP-Code-Sequenzstück verknüpft und damit recht flexibel ist, muss bei der herkömmlichen Microarray-Plattform das Array jeweils auf die spezifische Fragestellung (Detektion von Gen A bis Gen D) zugeschnitten werden.

chend stark variierende Signalstärken anzeigen.

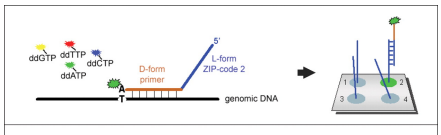
2.4 Ausblick auf universelle Arrays

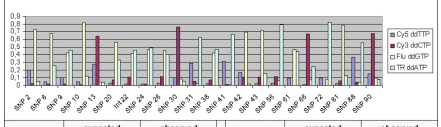
Die bisher vorgestellten verschiedenen Microarrays sind ganz spezifisch für die entsprechende molekularbiologische Untersuchung entworfen und produziert worden. Daneben muss noch zusätzlich der auf den zu untersuchenden Organismus abgestimmte Chip benutzt werden. Sinnvoll wäre es hier, eine universell einsetzbare Array-Plattform zu besitzen, die leicht auf die jeweilige Aufgabenstellung und den Zielorganismus hin angepasst werden kann.

Einen weiteren Nachteil der Array-Plattform stellt die Störung der chemischen Detektionsprozesse durch die Festkörpergrenzfläche des Arrays dar. Viele dieser Prozesse könnten innerhalb einer homogenen Lösung besser und ungestörter ablaufen.

Eine mögliche Lösung für beides stellen *ZIP-Code-Arrays* dar [GWD+99, Hoh06]. Dieser Ansatz ähnelt den in Abschnitt 1.5 erwähnten Bead-Arrays. Auch hier wird der eigentliche Sondenanteil mit einem speziellen Sequenzstück (ZIP-Code) verbunden, das sich aber im Unterschied zu den Bead-Arrays für die „Dekodierung“ der Sondeninformation mit einem komplementären Gegenstück auf der Arrayoberfläche verbindet (siehe Abbildung 2.5).

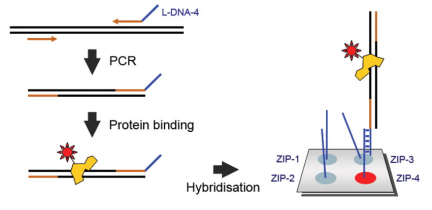
Sehr wichtig ist dabei, dass das „organisatorische“ ZIP-Sequenzstück nicht mit der Ziel-DNA wechselwirkt und es auch nicht zu Kreuzhybridisierungseffekten kommt.

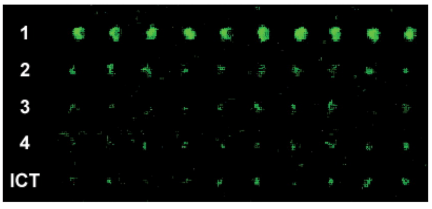




SNP-id.	expected sequence	observed sequence	SNP-id.	expected sequence	observed sequence
SNP 2	G	G	SNP 38	G and A	G and A
SNP 8	G	G	SNP 41	A	A
SNP 9	G and A	G and A	SNP 42	G	G
SNP 10	G	G	SNP 43	G	G
SNP 13	C	C	SNP 56	A	A
SNP 20	G	G	SNP 61	G and A	G and A
Int 22	G and A	G and A	SNP 66	C	C
SNP 24	G and A	G and A	SNP 72	A	A
SNP 26	G and A	G and A	SNP 81	A	A
SNP 30	C	C	SNP 88	A	A
SNP 31	A	A	SNP 90	C	C

Abbildung 2.6: SNP-Genotyping mit Hilfe eines ZIP-Code-Arrays (Quelle [HMJ+06]). Im oberen Teil ist der prinzipielle Ablauf dargestellt und das Ergebnis der Auswertung für 21 SNP-Stellen im Genom eines Menschen. Prinzipiell erfolgt die Detektion der vorliegenden SNP-Form mittels SNP-spezifischen Primern und fluoreszenzmarkierten Dideoxynukleotiden, die an der SNP-Position einen Kettenabruch induzieren.





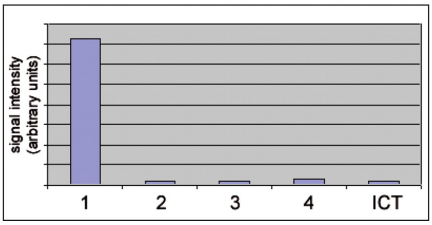


Abbildung 2.7: Untersuchung von DNA-Protein-Wechselwirkungen mit Hilfe eines ZIP-Code-Arrays (Quelle [HMJ+06]). Der obere Teil des Bildes zeigt den grundlegenden Ablauf des Experimentes. Dabei werden aus L-DNA-markierten Primern doppelsträngige DNA-Fragmente synthetisiert und mit dem interessierten fluoreszenfarbstoffmarkierten Protein vermengt. Anschließend erfolgt die kovalente Bindung an die DNA und die Hybridisierung auf dem ZIP-Code-Array. Im unteren Teil ist das Ergebnis einer solchen Analyse zu sehen, in der für das Fragment an Stelle 1 eine Bindung nachgewiesen wird.

Eine sehr gute Möglichkeit dies zu verhindern besteht in der Verwendung von L-DNA, dem Enantiomer (Spiegelbildisomer) der in Lebewesen normalerweise vorzufindenden D-DNA [HMJ+06], da sie nicht mit der D-Form reagiert.

Eine universelle Plattform auf Basis eines ZIP-Code-Arrays lässt sich leicht für viele Experimentformen adaptieren, wie zum Beispiel für das Genotyping (siehe Abbildung 2.6), für die Untersuchung von Splice-Varianten, für Genexpressionsanalysen, für CGH-Experimente und für epigenetische Studien (siehe Abbildung 2.7).

Teil II

Verfahrens- und Software-technische Entwicklungen für die Auswertung von Microarrays

Kurzübersicht zu Teil II

Die im Einführungsteil I umrissenen Eigenschaften der Microarray-Technologie weisen auf die größten Herausforderungen hin, die bei der Auswertung von Microarraydaten zu meistern sind. Neben dem sehr, sehr großen Datenumfang ist der gesuchte Effekt oft sehr klein und durch die hohe Anzahl an möglichen Fehlerquellen zusätzlich von Störsignalen maskiert. Diese Herausforderungen sind nur mit Hilfe einer angepaßten und guten informationstechnischen Grundlage zu bewältigen.

Zentrale Aufgabe dieser Arbeit war es deshalb, sowohl neue geeignete Verfahren zu finden und zu evaluieren, wie auch die notwendigen Sotware-technischen Grundlagen für eine praktische Analysearbeit zu legen. Dabei galt das Hauptaugenmerk der Arbeit den beiden wichtigsten Aspekten der mathematisch-statistischen und visuell explorativen Verfahren bzw. ihrer günstigsten Kombination. Als Ergebnis entstanden dabei vor allem zwei größere Projekte – Mayday und SpRay:

- **Mayday**
  Mayday ist ein Akronym, das für **M**icroarr**ay** **D**ata **A**nal**y**sis steht und eine vielseitig erweiterbare Plattform für die Auswertung von Microarrays bezeichnet [DGN06, DGS+08]. Mayday stellt für die Microarray-Analyse eine breite, Software-technische Werkzeugpalette und eine flexible informationstechnische Infrastruktur zur Verfügung. Es ist somit sowohl für den direkten Einsatz bereits vorhandener und implementierter Methoden durch Anwender im Labor oder bei der Auswertung von Studien gedacht, wie auch für die Entwicklung und Evaluation neuer Algorithmen. Dabei hat sich gezeigt, dass Mayday aufgrund seiner Flexibilität auch in der Lage ist, Verfahren für neue Array-Technologien, wie beispielsweise die ChIP-on-Chip-Technologie, zu integrieren.
  Kapitel 3, 4 und 5

- **SpRay**
  SpRay verfolgt demgegnüber einen spezialisierteren, auf hohe Leistung hin optimierten Ansatz. Es stellt eine Anwendung [DHNB08, DHNB06] für die visuell explorative Analyse von Microarray-Daten dar, wobei eine sehr enge Integration in die Statistik-Umgebung R [R D07] für die Verfügbarkeit einer Vielzahl mathematisch-statistischer Routinen sorgt. Zentral bei diesem Projekt war der Entwurf einer sehr reaktiven Visualisierungsanwendung für die Kombination aus Rohdaten und abgeleiteten statistischen Werten, um eine stark nutzergestütze Exploration der Daten überhaupt erst zu ermöglichen. SpRay verkörpert damit prototypisch eine Anwendung des neu entstandenen

Gebietes der Visual Analytics.
Kapitel 6

Mayday und SpRay sind andauernde, evolvierende Projekte und der erreichte Stand, ist insofern nur als Anfang einer weiteren Entwicklung zu sehen. Jedoch zeigt die Anzahl der Rückmeldungen von Nutzern, wie auch die Verleihung des *bwcon Sonderpreis IT & Life Sciences 2006* des Landes Baden Württemberg [bwc06], dass Mayday bereits den nötigen Reifegrad für einen allgemeinen, praktischen Einsatz erreicht hat.

3 Ein Softwareframework für die Genexpressionsanalyse

> It is a capital mistake to
> theorize before one has data.
>
> *(Sir A. Conan Doyle)*

3.1 Motivation

Die Motivation zum Start des Projektes Mayday entstammt ganz wesentlich eigenen Erfahrungen, die im Zusammenhang mit Projekten gemacht wurden. Insbesondere Erfahrungen aus dem Kooperationsprojekt [KMS+04] mit dem Leipziger Max-Planck-Institut für Evolutionäre Anthropologie (*MPI EVA*) bildeten den Ausgangspunkt für die grundlegende Konzeption und Ausrichtung von Mayday.

So stellten sich bei der praktischen Arbeit mit der Auswertung von Genexpressionsstudien die folgenden Unzulänglichkeiten heraus, die einen nicht unerheblichen zusätzlichen Aufwand verursachen:

1. Meistens ist die Nutzung einer Vielzahl verschiedener Programme für das Bewältigen einer kompletten Analyse notwendig. Hierbei kommt es notgedrungen an den jeweiligen „Systemgrenzen“ der verschiedenen Programme immer wieder zu einem Mehraufwand durch das Umwandeln von einem Datenformat in das andere. Oftmals müssen dafür eigene Skripte oder Programme geschrieben werden.

2. Eine bisher nicht ausreichend umgesetzte Forderung ergibt sich aus der großen Dynamik und dem Umfang des Feldes der statistischen Analysemethoden von Genexpressionsdaten, weshalb neueste Entwicklungen schnell und relativ einfach in Form von Tools für die Anwendung in der Praxis bereitgestellt werden müssen.

3. Es besteht ein Mangel an visuell anspruchsvollen Routinen bzw. Implementationen, deren Exportformate die Qualitätskriterien erfüllen, die für eine Veröffentlichung der Ergebnisse in Fachjournalen erforderlich sind.

4. Die bei der visuellen Datenexploration erzeugten Bilder sind mitunter nicht ausreichend detailliert auflösbar, um wichtige, interessante Fragestellungen beantworten zu können.

5. Die Abhängigkeit der Anwendungen von bestimmten Betriebssystemen beschränkt deren breite Einsatzfähigkeit in Labor und Klinik.

6. Das Vorliegen von mangelhafter und veralteter Dokumentation und Beschreibung der vorhandenen Features erschwert deren Benutzung.

7. Insgesamt ist eine geringe Benutzerfreundlichkeit und unzureichende Beachtung der Usability, insbesondere im Hinblick auf einen „Standardnutzer“ festzustellen.

Gerade auf Programme aus dem Bereich Public Domain und Open Source treffen die genannten Probleme besonders zu, da diese oft von den jeweiligen Wissenschaftlern bzw. Spezialisten eines Anwendungsgebietes im Hinblick auf die Demonstration der Einsetzbarkeit ihrer Ideen geschrieben wurden und nicht auf einen breiten Einsatz durch Anwender ohne nähere Fachkenntnisse.

3.2 Überblick über die existierende Analysesoftware

Neben einer ganzen Reihe von Tools, die auf ein bestimmtes Anwendungsgebiet spezialisiert sind, wie beispielsweise BRB Array [SL07] und Expression Profiler [KKC+04], gibt es Anwendungen, die einen mehr oder weniger integrierenden Ansatz verfolgen. In diesem Abschnitt soll ein kurzer Überblick über die wichtigsten und wesentlichsten Anwendungen dieser Art und ihrer jeweiligen Stärken und Schwächen gegeben werden. Da die erwähnten „Spezialanwendungen“ das in Abschnitt 3.1 Punkt 1.) aufgeführte Problem verschärfen und damit keine Lösung darstellen, wird auf sie hier nicht weiter eingegangen.

3.2.1 Public Domain oder Open Source Anwendungen

Im Zusammenhang mit der Planung von Mayday ist dieser Bereich am interessantesten, weil für Projekte im wissenschaftlichen Umfeld vorzugsweise auf Open-Source-Lösungen zurückgegriffen werden sollte. Zum einen ist das aus finanzieller Sicht auf das Projektbudget ein Vorteil, aber es trägt durch die Offenheit des Quellcodes auch zur Transparenz und Nachvollziehbarkeit bei, einer Forderung, die eigentlich grundsätzlich an jedes wissenschaftliche Projekt oder jede Studie zu stellen ist (vergleiche hierzu [SMZ01]).

Für die praktische Anwendung im Labor oder der Arbeitsgruppe spielt neben dem freien Zugriff und der Manipulierbarkeit der Quellen die einfache Benutzung der Anwendung und das weitestgehende Verbergen der systemimmanenten Komplexität, sowohl vor dem direkten Anwender wie auch dem Entwickler von Erweiterungen, eine enorme Rolle. Für eine breite Verwendung von Open Source-Software wird der Aspekt der Usability immer wichtiger.

Bioconductor

Bioconductor [GCB+04], [GCH+05], [Bio08a] stellt die wichtigste und wesentlichste Plattform für die Genexpressionsanalyse dar. Basierend auf R [R D07], einer kom-

pletten Programmiersprache und -umgebung für statistische Aufgabenstellungen, besteht Bioconductor aus einer umfangreichen Sammlung von R-Paketen. Damit besitzt es zugleich eine sehr leistungsfähige Grundlage für eine schnelle Entwicklung und Erweiterung durch neue statistische Routinen und Methoden mittels des Package-Mechanismus von R. Allerdings ist ein genereller Nachteil von R und somit auch von Bioconductor die fehlende Benutzerfreundlichkeit und mangelnde direkte graphische Interaktivität über *Graphical User Interfaces* (GUI).

Eine nutzbringende Anwendung von Bioconductor setzt die Einarbeitung in die Sprache R und die Erstellung eigener R-Skripte voraus. Bioinformatiker und Statistiker bilden deshalb die eigentliche Zielgruppe von Bioconductor und weniger Biologen und Mediziner in Laboren oder Kliniken. Bioconductor ist äußerst mächtig, flexibel und mittels R-Packages erweiterbar, aber es setzt ein hohes Maß an Spezialwissen voraus.

TigerTM4

Das vorwiegend Java-basierte TigerTM4 [SSW+03, J.C08] vom *J. Craig Venter Institute* (JCVI) (ehemals *The Institute for Genomic Research* (TIGR)) ist eine Suite, die im Wesentlichen aus vier verschiedenen Anwendungen besteht:

- *Microarray Data Manager* (MADAM) mit einer MySQL-basierten *Minimal Information About a Microarray Experiment* (MIAME)-kompatible Datenbank [BHQ+01] im Hintergrund für die Speicherung und Verwaltung der Microarray-Daten
- TIGR_Spotfinder zur Bildanalyse der vom Microarray-Scanner gelieferten TIFF-Dateien (C/C++-basiert)
- *Microarray Data Analysis System* (MIDAS) für die Normalisierung und Qualitätsfilterung der Rohdaten, implementiert sind beispielsweise Methoden für die Auswertung der Replikate, die *Locally Weighted Scatterplot Smoother* (LOWESS)-Normalisierung, die Dye-Swap-Korrektur und die z-Score-Filterung der Intensitäten
- *Multiexperiment Viewer* (MeV) für die statistische und graphische Analyse bzw. das Finden von differentiell exprimierten Genen oder bestimmter Genexpressionsmustern, implementiert sind verschiedene Verfahren für das Clustering, die Berechnung von Distanzen, die visuelle Datenexploration und mehrere Klassifikationsverfahren des maschinellen und statistischen Lernens

MeV kann durch Module um neue Algorithmen oder Verfahren ergänzt bzw. erweitert werden.

BASE

BASE [STVC+02], [TVCS06], [BAS07] ist ein webbasiertes System, das von der Lund Universität entwickelt wurde. Es liegt derzeit noch in zwei technisch ver-

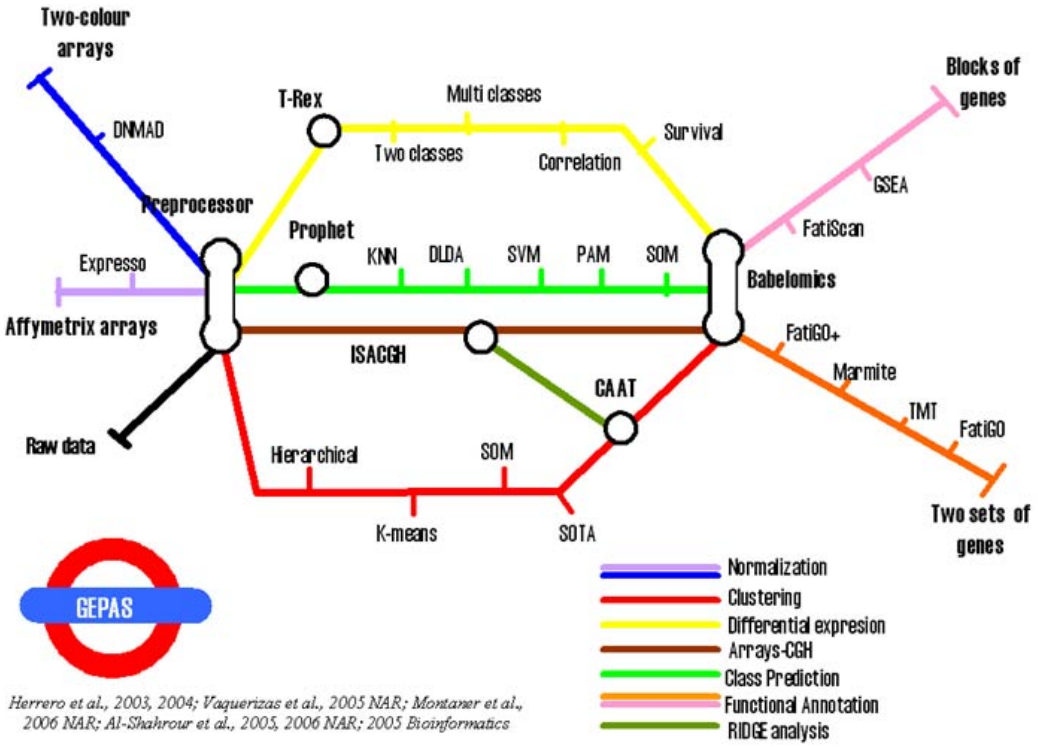


Abbildung 3.1: Übersichtsdarstellung der GEPAS-Komponenten. Entnommen der Tools-Seite von [GEP08].

schiedenen Varianten vor, der Version 1.2.x und dem sogenannten BASE 2. BASE 1.2.x setzt vor allem auf den Webserver Apache [The07a], PHP-Scripten [php07], C/C++-Programmen, einem SQL-basierten *Datenbankmanagementsystem* (DBMS), wie PostgreSQL [Pos07] oder MySQL [MyS07] und einer ganzen Reihe weiterer Programme und Bibliotheken auf (siehe `http://base1.thep.lu.se/documentation/install.html`). Diese Abhängigkeiten stellen eine nicht zu unterschätzende Hürde beim Aufsetzen und Selbstadministrieren eines *Inhouse-Systems* mit BASE dar.

BASE 2 stellt eine komplette Überarbeitung der Version 1.2.x in Java dar, der die Entwicklung auf eine einheitlichere Basis stellt und die Komplexität der in der alten Version angesammelten Abhängigkeiten reduziert.

GEPAS

Auch GEPAS („**G**ene **E**xpression **P**rofile **A**nalysis **S**uite“) [MTHC+06, GEP08] ist eine webbasierte Plattform, die mehrere verschiedene Funktionsmodule gebündelt zur Benutzung anbietet (vgl. Bild 3.1). Dabei erfolgt der Zugriff auf die Funktionalität der Suite über die Benutzung einfacher HTML-Formulare. Mit ihrer Hilfe werden Datenfiles auf den Server geladen oder vorhandene ausgewählt, falls nach dem Präprozessieren Daten innerhalb von GEPAS weiter bearbeitet werden sollen, und die zugehörigen Parameter für die einzelnen Verfahren eingestellt.

GEPAS besitzt ein umfangreiches Angebot an Methoden, die in eigenen Implementierungen von Algorithmen und Bioconductor-Routinen realisiert sind. Abbildung 3.1 gibt sowohl einen Überblick über die vorhandenen Funktionsmodule von GEPAS als auch über die möglichen Verarbeitungspfade in Abhängigkeit von der verwendeten Array-Technologie und dem angestrebten Ziel der Auswertung.

GEPAS ist ausschließlich für die Nutzung über das Internet gedacht und bietet keine Möglichkeit für eine gruppeninterne, webbasierte Installation als Inhouse-System. Als ein wichtiger Vorteil dieses Aspektes für den Endnutzer wird in der Veröffentlichung [MTHC+06] die gehobene Hardware-Ausstattung des Clusters, auf dem GEPAS betrieben wird, herausgestellt - vor allem die Hauptspeichergröße.

EMMA2

Auch EMMA2 stellt, genauso wie BASE und GEPAS, eine webbasierte Plattform für die Analyse von Expressionsdaten dar [Don07, DGB+03]. Anwender müssen sich am EMMA-Webserver anmelden und können dann unter ihrem Zugang eigene Daten hochladen und davon ausgehend weitere Analysen durchführen. Eine eigene Installation als Inhouse-System ist nicht vorgesehen, allerdings kann auf die Funktionalität von EMMA über Webservices zugegriffen werden.

Wesentliches und vordringliches Ziel, das sich die Entwickler von EMMA gestellt haben, ist eine möglichst umfangreiche Implementierung gängiger offener Standards wie *Minimal Information About a Microarray Experiment* (MIAME), das *Microarray Genexpression - Object Model* (MAGE-OM) und die *Microarray Genexpression - Markup Language* (MAGE-ML). Als grundlegende Implementierungssprache wählten die Entwickler, infolge der starken Orientierung auf die Anwendung über das Netz, Perl. Für statistische und numerische Routinen wird auf R und Bioconductor zurückgegriffen. Module, die eine größere Interaktivität zur Verfügung stellen sollen, sind als Java-Applets realisiert. Die objektrelationale Abbildung der Datenobjekte von EMMA (MAGE-OM-Objekte) auf *Structured Query Language* (SQL)-Tabellen des genutzen relationalen DBMS und allgemeinen *Extensible Markup Language* (XML)-Transformationen wird *XSL Transformation* (XSLT) verwendet.

EMMA läßt sich durch Plugins erweitern, die vorwiegend in R oder Perl zu implementieren sind.

3.2.2 Kommerzielle Anwendungen

Da sich Mayday vor allem als eine Analyse-Plattform für Microarraydaten im akademischen und Public-Domain- bzw. Open Source-Umfeld platziert, sollen hier nur kurz die wichtigsten kommerziellen Applikationen erwähnt werden.

GeneSpring

GeneSpring GX [Agi07] stellt in seiner aktuellen Form eine sehr umfassende und weithin verbreitete Applikation für die Auswertung von Microarray-Daten dar. Die Auswertung der Systeme industriell führender Anbieter von Expressionstechnologie, Affymetrix [Aff08b], Illumina [Ill08b] und Agilent [Agi08], sind integriert und werden sehr gut unterstützt.

Beispielhaft für die in GeneSpring angebotene Funktionalität seien die folgenden Punkte genannt:

- Unterstützung des Datenaustauschformates MAGE-ML der *Object Managment Group* (OMG) für Microarraydaten
- Verschiedene Verfahren zur Qualitätseinschätzung und Vorverarbeitung der Chip-Daten, z.B.:
  - Intensitätsabhängige Normalisierungsmethoden, wie *Locally Weighted Scatterplot Smoother* (LOWESS) [Cle79, Cle81]
  - chip- oder genbezogene Normalisierung
  - Normalisierung mit Hilfe von Kontrollen
- Mehrere Methoden für die Datenvisualisierung:
  - 2D- und 3D-Scatterplots
  - Box-Plots
  - Volcano-Plot
  - Dendrogramme (2D)
  - Chromosomen-Maps
  - Pathway-Diagramme
- Fortgeschrittene statistische Methoden zur Auswertung:
  - Hypothesentests, wie z.B. t-Test
  - *Analysis of Variance* (ANOVA) (einfaktorielle und zweifaktorielle)
  - verschiedene Post-Hoc Tests
  - mehrere Korrekturen für multiples Testen
  - *False Discovery Rate* (FDR)
- Methoden zur Mustererkennung und des Machine Learnings:
  - Verfahren zur hierarchischen Clusterung, sowohl von Genen wie auch von Experimenten
  - *Self Organizing Maps* (SOM)
  - k-Means Clustering
  - *QT Clustering* [HKY99]
  - *Principal Components Analysis* (PCA)

  - *k-Nearest-Neighbor Klassifikation* (kNN) Klassifikation
  - Klassifikation mit *Support Vector Machine* (SVM)

- Zugriff auf GCOS- (Affymetrix) und BASE-Datenbanken innerhalb von GeneSpring

- Zugriff auf die Funktionalität von GeneSpring über ein *Application Programming Interface* (API) für Programmierer

- Mögliche Integration von Methoden aus R [R D07], BioConductor [Bio08a], SAS [SAS08] oder MATLAB [Mat08]

- Unterstützung von Groovy-Scripts

Bei eigenen testweisen Anwendungen und der Befragungen von Biologen, die das System im routinemäßigen Laborbetrieb einsetzen, stellten sich vor allem die folgenden Kritikpunkte heraus:

- Einschränkung bei der Interaktivität durch den Eindruck eines trägen Reaktionsverhaltens der Anwendungsoberfläche

- Verbesserungsfähiges Sitzungsmanagement (Abspeichern von Arbeitsständen, Undo-Redo-Funktionalität, Versionierung)

- Management eines Zugriffs durch mehrere Nutzer für eine gemeinschaftliche Auswertung von Daten (Versionierung mit Nutzerkennung)

Spotfire DecisionSite

Spotfire DecisionSite ist eine umfangreiche Anwendung für die Datenexploration und -analyse in verschiedenen Anwendungsbereichen, wie z.B. dem Ingenieurwesen. Speziell für die Auswertung von Genexpressionsdaten ist „Spotfire DecisionSite for Microarray Analysis“ [Spo08b] und „Spotfire DecisionSite for Functional Genomics“ [Spo08a] gedacht. Spotfire bietet, ähnlich GeneSpring, umfangreiche statistische und visuelle Methoden an. Für eine genauere Betrachtung sei neben den oben angegebenen *Uniform Resource Locator*s (URL) auf die sehr aufschlussreiche Anwendungsstudie [SNLD06] verwiesen. Hier findet man auch eine Einschätzung von Spotfire, das neben anderen zusätzlichen Softwarewerkzeugen bei der Analyse eines realen Datensatzes von erfahrene Bioinformatiker eingesetzt werden sollte.

Dabei zeigten sich die sehr guten interaktiven, explorativen Visualisierungen in Verbindung mit ausgezeichneten Filtermechanismen (*Dynamic Queries* (DQ)) in Spotfire als eine große Stärke. Gerade bei der aufwendigen ersten explorativen Analyse der Rohdaten ist eine schnelle, zeitsparende Anwendung des Visualisierungsdreiklangs „Overview, Zoom and Filter“ sehr wichtig. Der anfänglich 45001×72 große Datensatz konnte im Verlauf der ersten Exploration auf die Dimension

30000×72 reduziert werden. Eingesetzt wurden dabei mehrere verschiedene Visualisierungen, wie Streudiagramme (Scatterplots), Profilplots, verschiedene Clusterdarstellungen und dies in Kombination mit verschiedenen statistischen Routinen, zum Beispiel dem t-Test. Sogar für Präsentationen vor kooperierenden Arbeitsgruppen wurde Spotfire während der Diskussionen eingesetzt, was für gute interaktive und schnelle Abläufe in Spotfire spricht.

Auf der anderen Seite zeigt [SNLD06] aber auch einige Defizite auf, wie das Fehlen einer History, durch die sich kompliziert herzustellende Anordnungen von Daten leicht wieder rekonstruieren lassen würden. Außerdem wechselten die auswertenden Bioinformatiker auf andere, „bioinformatischere“ Werkzeuge, wie beispielsweise Programme für die Abbildung der Daten auf Stoffwechselpfade, um die biologische Relevanz gefundener interessanter Zusammenhänge besser abschätzen zu können. Auch für die Erzeugung von publikationsreifen Grafiken wurde von den Auswertenden eine andere Software zu Hilfe genommen.

3.3 Definition der Entwurfsziele von Mayday

Die im Abschnitt 3.1 aufgezählten Probleme und die Eigenschaften der in Abschnitt 3.2 vorgestellten wichtigsten integrierenden Anwendungen definieren die „ökologische Nische“ von Mayday und legen damit die wesentlichen Eckpunkte des Zielentwurfs fest.

Dabei muss eine Vielzahl an verschiedensten organisatorischen und inhaltlichen Aspekten berücksichtigt werden (vgl. Abbildung 3.2).

3.3.1 Flexibilität und Erweiterbarkeit

Sowohl das Feld der Genexpressionsanalyse, wie auch allgemein der Bearbeitung und Auswertung von biologischen Daten ist von einer so starken Dynamik und Veränderung geprägt, dass sich über nennenswerte Zeiträume lediglich offene und erweiterbare Architekturen behaupten können.

Die ständige Entwicklung sowohl neuer technologischer Verfahren, wie z.B. neuer Arraytypen (Tiling-Arrays, ChIP-on-Chip), als auch neuer Analysemethoden für bestehende Technologien verlangt die Fähigkeit zur schnellen Integration neuer Funktionalität in die vorliegende Applikation. Dies legt als Rahmen für Mayday eine Plattform - eine flexible Plugin-basierte Struktur - zugrunde, auf der eine *Service Oriented Architecture* (SOA) abgebildet werden kann.

Ein zentraler Kern stellt lediglich die allgemeinen, infrastrukturellen und kommunikativen Mechanismen sowie die einfachsten Datenstrukturen für die Interaktion von Modulen bzw. Plugins zur Verfügung.

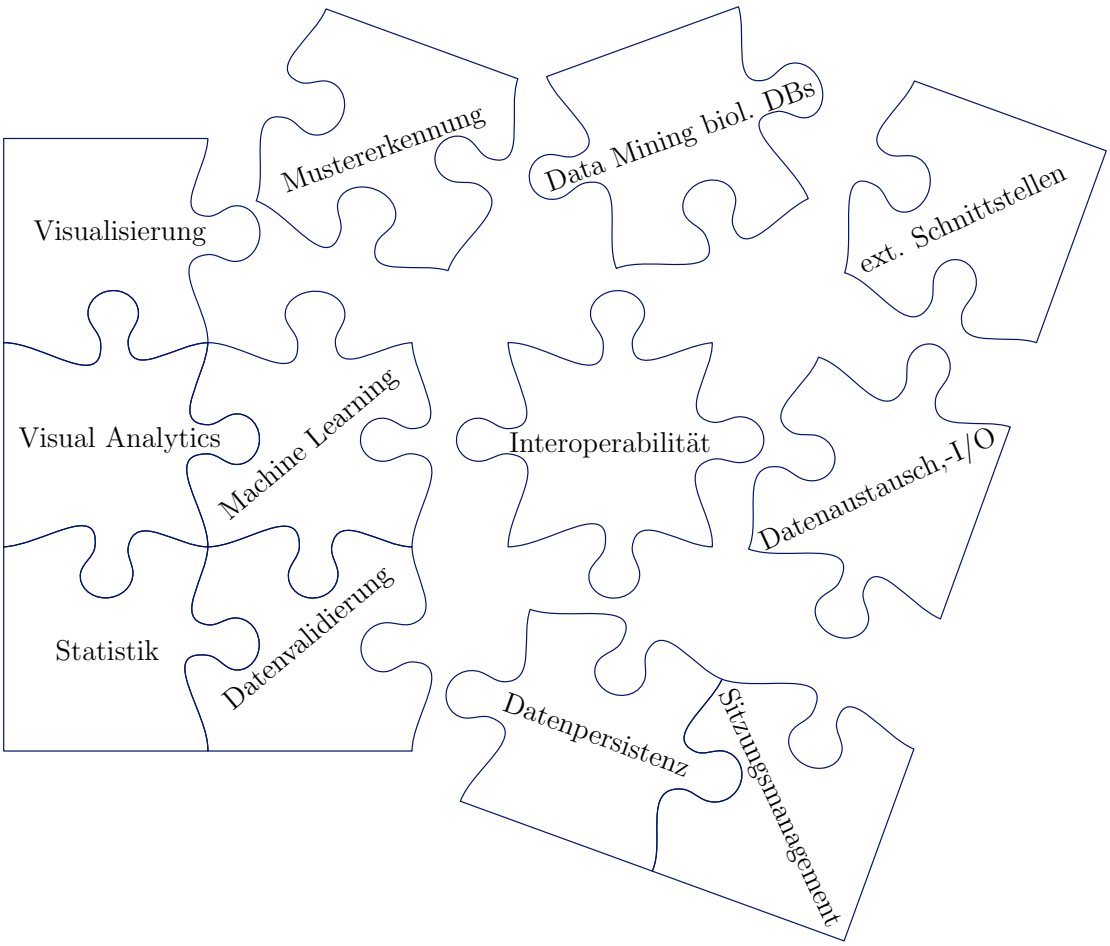


Abbildung 3.2: Der Entwurf einer Plattform zur Analyse von Microarrays weist eine Vielzahl an verschiedenen Aspekten auf - sowohl inhaltlich fachlicher und organisatorischer Natur. Die in der Abbildung angegebene Übersicht ist keineswegs vollständig und verdeutlicht die modulare Struktur, die diese Plattform auf Grund ihres breiten Einsatzes haben muß. Ein weiterer wichtiger Aspekt, der allerdings die ganze Anwendung global betrifft ist deren Benutzbarkeit.

3.3.2 Kommunikation, Datenaustausch und Datenmanagement

Die im vorhergehenden Abschnitt 3.3.1 genannte rasche Technologieentwicklung verlangt zugleich nach der Fähigkeit zur raschen Integration neuer Datenarten und Datenformate. Für viele biologische Fragestellungen ist gerade eine geeignet kombinierte Sichtweise auf Daten aus verschiedenen Datenquellen unabdingbar. Dabei können diese Daten von unterschiedlicher Struktur sein, zum Beispiel numerische Expressionsdaten und textbasierte Annotationen der zugehörigen Daten.

Es besteht die Notwendigkeit eines einfachen Zugriffs auf verschiedene Datenquellen, beispielsweise über das Internet, wie auch für eine einfache Konvertierung zwischen verschiedenen Datenformaten und damit für eine Implementation grundlegender Datenaustauschformate.

Neben diesem Aspekt ist auch der große Umfang der anfallenden Daten ein wesentliches Merkmal der Bioinformatik. Es muss deshalb ein leistungsfähiges Datenmanagement vorhanden sein, günstigstenfalls duch die Verwendung eines bewährten DBMS als Backend-Datenspeicher.

Eine weitere Anforderung an das Datenmanagement stellt die übliche Arbeitsweise bei einer Analyse dar. Aufgrund der Datenmenge wie auch der Vielzahl an potentiellen Auswertungsmethoden ist das Vorgehen sehr explorativ, d.h. mehrere verschiedene Methoden werden für den jeweiligen Analyseschritt probiert und eventuell wieder verworfen. Die Wiederherstellung des Zustandes vor einem Analyseschritt ist deshalb wichtig und damit eine Art von Versionierung der Analyse hilfreich bzw. nötig.

Die Analyse eines Genexpressionsexperiments oder auch eines biologischen Experiments ganz allgemein ist oft geprägt durch seinen interdisziplinären Ansatz. Das heißt, Fachleute verschiedener Disziplinen, wie der durchführende Biologe und der auswertende Statistiker bzw. Bioinformatiker sollten auf den gemeinsamen Datenstamm mit ihren jeweiligen fachbezogenen Sichten (Visualisierungen oder Analyseansätze) zugreifen können und das möglichst auch unabhängig vom geographischen Standort (verteilter versionierter Zugriff).

3.3.3 Leistungsfähige Methoden für die visuelle Datenexploration

Kennzeichnend für viele Fragestellungen in der biologischen Forschung ist das häufige Nichtvorhandensein von A-Priori-Modellen, weshalb die explorative Datenanalyse [Tuk77] im Rahmen der Analyse eines Experiments einen sehr hohen Stellenwert einnimmt. Da jedoch zusätzlich das Datenaufkommen sehr hoch ist, sind geeignete visuelle Verfahren zur Datenexploration unverzichtbar. Das in Abschnitt 3.2.2 erwähnte Spotfire gibt hier in Bezug auf die schnelle, interaktive visuelle Exploration und deren Kombination mit DQs ein gutes Beispiel.

Der Umfang der Datenmenge erzeugt zugleich einen hohen Druck in Richtung einer guten Geschwindigkeitsoptimierung der grafischen Routinen, denn visuelle Datenexploration muss in hohem Maße interaktiv durchführbar sein, um sinnvoll angewendet zu werden.

3.3.4 Leistungsfähige Statistik- und Machine Learning-Module

Messungen an biologischen Systemen sind von einer Vielzahl an potentiellen Fehlerquellen und Variationen begleitet. Um die davon erwünschte biologische Variation von der unerwünschten technischen zu trennen, bedarf es geeigneter statistischer Methoden und Modelle. Sowohl für die Qualitätsanalyse der einzelnen Messung wie auch der eigentlichen Hypothesen- und Wissensgewinnung ist die Verwendung von Verfahren aus Statistik und Machine Learning zentral.

Auch auf diesem Gebiet erzeugt die Menge an Daten einen hohen Druck in Rich-

tung auf eine Optimierung der Routinen in Bezug auf Geschwindigkeit und Speicherplatz.

Gerade in den letzten Jahren ist ein Zusammenwachsen der Bereiche Visualisierung und Statistik in Form der Visual Analytics zu beobachten, um eine bessere Analyse sehr großer Datenmengen realisieren zu können [TC05, Tho06, SNLD06, WAG06].

3.3.5 Integration von explorativer prototypischer und schematischer Benutzung

Ein weiterer wichtiger Aspekt wird bereits in Abschnitt 3.3.2 mit der Nutzung durch unterschiedliche Anwendergruppen angesprochen. Einerseits soll die Analyse-Software dem Statistiker und Bioinformatiker das Implementieren und Testen neuer Methoden und Modelle ermöglichen, aber andererseits dem Biologen im Labor die einfache Benutzung etablierter Verfahren mit günstigen Standardparametern gestatten. Dies gibt dem, das Experiment durchführenden, Biologen ein schnelles, einfach zu erlangendes Feedback und gewährt so bessere Möglichkeiten zur Qualitätssicherung und Prozessoptimierung.

Mayday muß somit einem versierten Experten für die Auswertung die Möglichkeit bieten, neue Ideen ohne großen Einarbeitungsaufwand zu implementieren und diese seinen Anwendern, Biologen im Labor oder Medizinern in der Klinik, in einfach ausführbarer Form (in der Art einer Batch-Datei) zur Verfügung zu stellen.

3.3.6 Breite Unterstützung gängiger Betriebssysteme und heterogener Umgebungen

Biologische oder klinische Labore wie auch Krankenhäuser sind von einer großen Heterogenität in Bezug auf die verwendeten Software- und Hardwareplattformen geprägt. Da dies die Zielumgebung für Mayday ist, müssen alle gängigen Systeme unterstützt werden und die Implementierung auch eine leichte und gute Integration in diese Form von Netzwerkumgebung ermöglichen.

Im Wesentlichen lassen sich die in Abschnitt 3.2 aufgeführten Systeme in Bezug auf ihren Einsatz in verschiedenen Umgebungen in zwei Modelle einteilen:

- installierbare Inhouse-Systeme, z.B. Bioconductor, TigerTM4 und GeneSpring
- Internet- bzw. webbasierte Systeme, z.B. GEPAS und BASE

Die Grenzen zwischen beiden Kategorien sind nicht strikt, so ließe sich BASE auch im hausinternen Netz installieren und nutzen. Dabei ist jedoch ein höherer Aufwand aufzubringen als bei einer dafür vorgesehenen Anwendung, wie TigerTM4. GEPAS hingegen sieht diese Möglichkeit gar nicht mehr vor.

Bei entsprechend guter rechentechnischer Ausstattung des Labors können Inhouse-Systeme in Bezug auf die Leistungsfähigkeit und die Kontrolle über die eigenen Daten von Vorteil sein. Webbasierte externe Systeme wiederum mindern den Aufwand

der Systeminstallation und -pflege, erfordern aber eine breitbandige Anbindung an das Internet und ein hohes Vertrauen in die den Dienst bereitstellende Institution oder Organisation.

4 Mayday als flexible Plugin-Plattform

> Absolute certainty is a privilege of uneducated minds-and fanatics. It is, for scientific folk, an unattainable ideal.
>
> *(C. J. Keyser)*

4.1 Übergreifende Systemanforderungen

Direkt aus den in Abschnitt 3.3 formulierten Designzielen und Randbedingungen für eine Analyseplattform leiten sich übergreifend architekturielle Grundentscheidungen für den Entwurf von Mayday ab.

4.1.1 Implementierungssprache

Grundlegend sind in den Punkten von Abschnitt 3.3 zwei gegensätzliche Anforderungen auszumachen. Einerseits muss die Plattform in Bezug auf Speicher und Geschwindigkeit sehr leistungsfähig und andererseits sehr einfach zu installieren und zu bedienen sein. Letzteres sollte auch von nicht IT-affinem Personal unter den heterogenen Systembedingungen eines Labors oder einer Klinik zu bewältigen sein.

Die oben genannten hohen Leistungsanforderungen sprechen für eine Implementierung in nativ compilierbarem Quellcode und damit bei einem Projekt dieser Größenordnung für C++. Allerdings ist die zweite Anforderung mit C++, insbesondere der Aspekt der einfachen Lauffähigkeit unter verschiedenen Betriebssystemen und der guten Einbindung in heterogene IT-Landschaften, nur unter sehr hohem Entwicklungsaufwand und einer entsprechend längeren Entwicklungszeit zu gewährleisten.

In Anbetracht der zur Verfügung stehenden Entwicklungskapazitäten und dem Zeitrahmen wurde als Kompromiss zwischen den erwähnten gegensätzlichen Aspekten Java als generelle Implementierungssprache gewählt. Vorteile von Java sind:

1. eine relativ gute Unabhängigkeit vom jeweiligen Betriebssystem durch die Verwendung der *Java Virtual Machine* (JVM) als Laufzeitumgebung

2. eine bessere Leistungsfähigkeit als rein interpretierte Sprachen (Perl, Python, Ruby ... etc.) durch die Verwendung eines *Just-In-Time-Compiler*s (JIT-Compiler)

3. eine gut dokumentierte, umfangreiche Klassenbibliothek ist bereits Teil des Standardumfangs

4. eine Vielzahl zusätzlicher, frei verfügbarer, quelloffener Bibliotheken und Tools

5. eine gute Client-Server- und Internettauglichkeit durch einfach zu programmierende Netzwerkschnittstellen

6. einen in vielen verschiedenen, erfolgreichen Projekten nachgewiesenen Reifegrad als Implementierungsplattform großer bis sehr großer Softwaresysteme

7. eine sehr gute Unterstützung der Entwicklung durch ein breites Angebot an integrierenden Entwicklungswerkzeugen

8. eine installationsfreie Ausführbarkeit von Java-Anwendungen über das Internet hinweg mit Hilfe von Java-Web-Start [Sun08]

Demgegenüber stehen vor allem leistungsbedingte Nachteile:

1. geringere Steuerungs-, Optimierungs- und Leistungsfähigkeit in Bezug auf Speicherbedarf und Laufzeitverhalten gegenüber nativ kompilierenden Sprachen - beispielsweise gibt Brian Kernighan für sein algorithmisches Beispiel auf Seite 8 in [OW07] eine rund 6 mal langsamere Laufzeit der Java-Version im Vergleich zur C-Variante an (in Bezug auf den Speicherbedarf vgl. Abschnitt 5.3.1 ab Seite 75)

2. geringere Reaktivität der Java-basierten GUIs als vollkommen nativ implementierte Bibliotheken

3. geringere Möglichkeit spezielle Vorzüge einer vorhandenen Prozessorarchitektur auszunutzen

Einen möglichen Ansatz, um zur Not die genannten leistungsmäßigen Probleme abzumildern, bietet die Auslagerung sehr rechen- und speicherintensiver Prozesse in nativ kompilierte Routinen. Java bietet hierfür einen standardisierten Zugriff über das sogenannte *Java Native Interface* (JNI) [Sun04a]. Auch für die in Punkt 2 erwähnte langsamere Reaktivität der in Java standardmäßig verwendeten Swing-Oberfläche gibt es mit dem *Standard Widget Toolkit* (SWT) [Ecl08c] der Eclipse Foundation einen konkurrierenden, JNI-basierten Ansatz. Aufgrund des Fehlens mancher Funktionalität in bestimmten Betriebssystem-Widgets und der Notwendigkeit diese dann zu emulieren, erreicht aber auch dieser Ansatz nicht die Reaktionsgeschwindigkeiten rein nativer Oberflächen.

Zu beachten ist, dass ein solches Vorgehen - die Auslagerung in nativ kompilierbare Module - einen Teil der oben genannten Vorteile einer rein Java-basierten

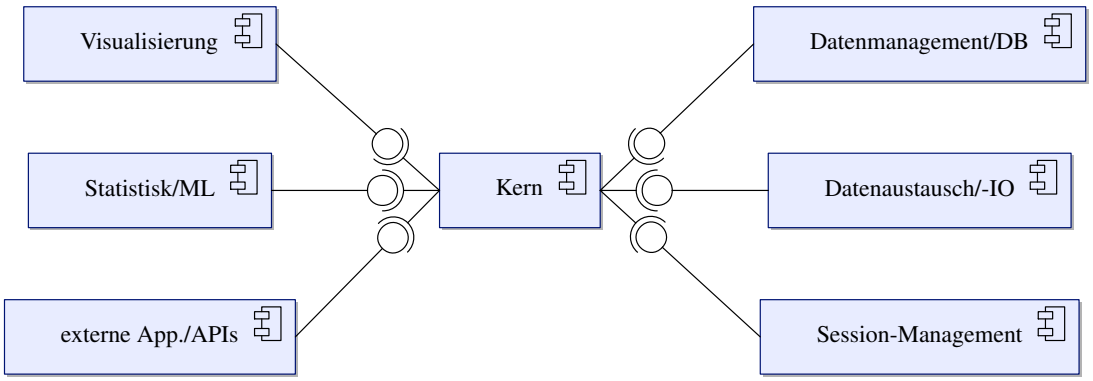


Abbildung 4.1: Übersicht über den groben, inhaltsstrukturierten, modularen Aufbau von Mayday. Der wesentliche Teil der Funktionalität ist einzelnen Modulen vorbehalten, die über ein, von einem leichtgewichtigen Kern bereitgestelltes, Plugin-Interface angekoppelt werden.

Implementierung, wie die Systemunabhängigkeit und die Verfügbarkeit über Java-Web-Start, zunichte macht.

4.1.2 Einsatzmodell von Mayday

In Bezug auf die in Abschnitt 3.3.6 genannten Einsatzmodelle ist Mayday primär für eine Realisierung als Inhouse-System vorgesehen. Als eine Art Kompromiss zwischen den in Abschnitt 3.3.6 erwähnten Kategorien eröffnet die zusätzliche Realisierung von Mayday als Web-Start-Anwendung [DGS+08] den Web-gestützten, installationsfreien Einsatz von Mayday. Dabei wird die Anwendung in einem spezifizierten Bereich auf dem Rechner des Nutzers ausgeführt und keine Nutzdaten werden an den „Web-Start-ausliefernden“ Web-Server übertragen.

Abgesehen davon bietet die Realisierung von Mayday in Java auch die Möglichkeit, die implementierte Funktionalität bzw. Teile davon zukünftig auf eine Java-basierte Webplattform zu portieren.

4.1.3 Das Komponentenmodell von Mayday

Der Anforderungsanalyse von Abschnitt 3.3 sind notwendig zu realisierende Komponenten bzw. Komponentengruppen von Mayday zu entnehmen. Strukturiert und kategorisiert man die Anforderungen sehr grob nach ihren inhaltlichen Schwerpunkten, so lassen sich die folgenden Module identifizieren (vgl. auch Darstellung in Abbildung 4.1):

- ein leichtgewichtiges, zentrales Kernmodul
- eine umfangreiche Visualisierungskomponente
- ein ebenfalls umfangreiches Modul für die Statistik und das Machine Learning

- eine Komponente für den Datenaustausch sowie den Datenim- und -export
- eine Komponente für das Datenmanagement, die Datenspeicherung und den DB-Anschluß
- eine Komponente für Schnittstellen zu externen Programmpaketen, Anwendungen oder Umgebungen
- ein Modul für das Session-Management, einschließlich der Möglichkeit zum Batch- bzw. Skriptbetrieb

4.2 Der Mayday-Kern

Die Anforderungen in Abschnitt 3.3.1 definieren im Wesentlichen die Struktur des Mayday-Kerns. Hauptsächlich stellt der Kern die notwendige Infrastruktur und allgemeine Datenobjekte für das Zusammenspiel der Plugins zur Verfügung. Für die letztendliche Realisierung der Funktionalität sind allein die einzelnen Komponenten bzw. Module zuständig.

4.2.1 Grundlegende Datenstrukturen

Aufgrund der sehr allgemeinen, flexiblen und teilweise unvorhersehbaren zukünftigen Anforderungen an den Kern, kann nur der vorstellbar „kleinste gemeinsame Nenner“ zentral implementiert werden. Dies bedeutet einerseits die Verwendung der einfachsten Datentypen sowie bei notwendigerweise komplexeren Datentypen ein entsprechendes Datenstruktur-Framework:

- Grundlage der meisten Array- bzw. Assay-bezogenen Analyseprozesse bildet die Manipulation und Transformation einer Ansammlung von m Objekten, z.B. Array-Features oder Genen, eines n-dimensionalen reellen Zahlenraums. Hierbei entspricht die Zahl n der Dimensionen den durchgeführten Untersuchungen.

  Abbilden lässt sich diese Struktur programmintern am besten und am allgemeinsten auf eine $m \times n$ dimensionale Matrix von Gleitkommazahlen. Die m Zeilenvektoren entsprechen den oben genannten Features und erhalten zusätzlich einen dieses Feature identifizierenden Bezeichner. Dies entspricht einem eindeutigen Namen für die Zeilen, während die Spalten ebenfalls über die zugehörige Kondition (Replikatnummer, Zeitpunkt der Zeitreihe oder eine bestimmte Gewebeart) eindeutig identifiziert werden können (vgl. Abbildung 4.2).

  Im Zusammenhang mit Expressionsexperimenten wird diese Matrix auch als Expressionsmatrix bezeichnet. Das Konzept selbst trägt jedoch viel weiter

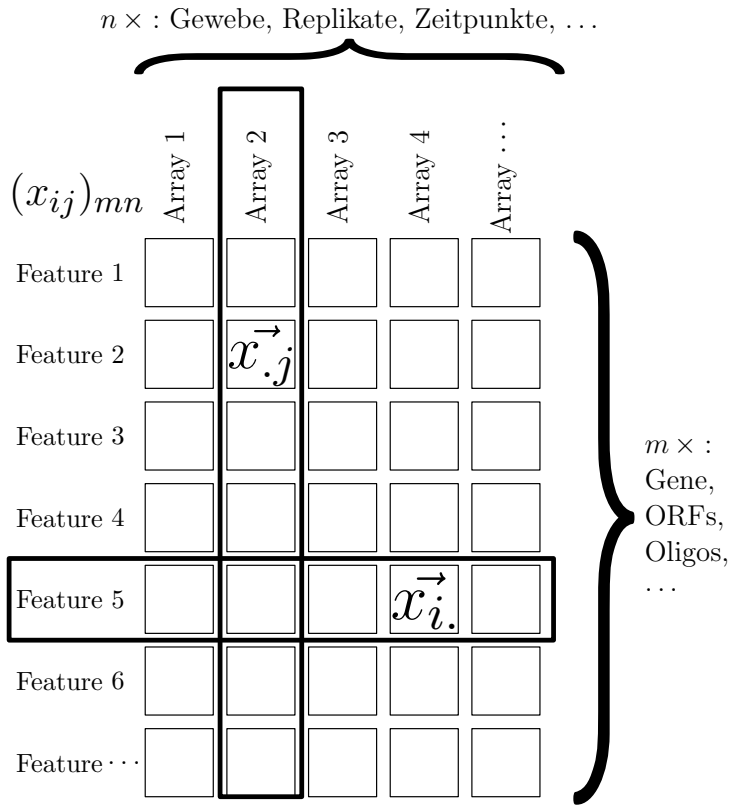


Abbildung 4.2: Logisches Modell einer Array-basierten Messung. Die einzelnen auf das Array aufgebrachten, eindeutig identifizierbaren Features lassen sich in allen Messkonditionen als $m \times n$ Matrix $M = (x_{ij})_{mn}$ darstellen. Der Zeilenvektor $\vec{x_{i.}}$ bildet dabei das Profil eines Features über alle verschiedenen Konditionen bzw. Arrays hinweg. Der Spaltenvektor $\vec{x_{.j}}$ repräsentiert ein komplettes Array und damit eine Messung für eine bestimmte Messkondition.

und erlaubt die Behandlung einer Vielzahl von arraybezogenen Problemstellungen. Auch der Erfolg von Systemen, deren Datenstrukturen im Wesentlichen nur auf Vektoren, Matrizen und Listen beruhen, wie R [R D07] und Matlab [Mat08] zeigen die große Tragweite und Verallgemeinerungsfähigkeit dieses Konzepts.

Im Kern von Mayday erfolgt die Abbildung dieser Struktur zentral auf die Klassen `MasterTable`, `DataSet`, `Probe` und `ProbeList` (siehe Abb. 4.3). Die einzelnen, einem Feature des Arrays zugeordneten, Messprofile werden Mayday-intern in einer `Probe` gehalten. Das Array selbst findet seine programmtechnische Entsprechung in der Klasse `MaterTable`. Mit Hilfe von `ProbeList`-Objekten ist es möglich, einzelne Features zu strukturieren und zu klassifizieren (vgl. hierzu auch [Geh03]). Die Verwaltung der `ProbeList`-Objekte obliegt Objekten der Klasse `ProbeListManager`.

- Andererseits ist der zukünftige Bedarf an komplexeren Datentypen abzuse-

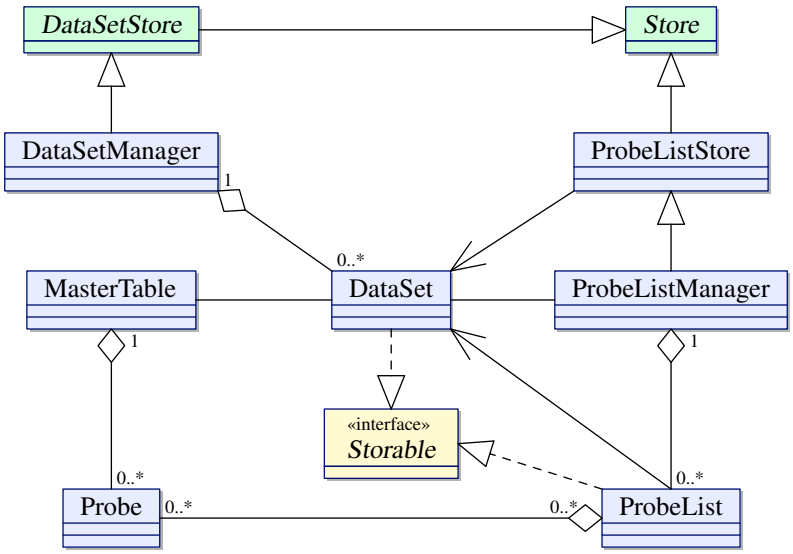


Abbildung 4.3: Das UML-Diagramm zeigt die wesentlichen und zentralen Strukturen für die Speicherung der Daten des logischen Modells von Abb. 4.2 in Mayday. Hier wie auch in den folgenden Diagrammen dieses Abschnitts sind Schnittstellen gelb, abstrakte Klassen grün und normale Klassen blau hervorgehoben.

hen, die eng mit den jeweiligen Features eines Arrays verbunden sind. Hierbei handelt es sich zum Beispiel um Annotation, wie weitergehende Beschreibungen, Angaben zu Stoffwechselpfaden oder Klassifikatoren, die aus anderen Berechnungen heraus oder Datenbanken gewonnen worden sind. Da deren genauer Typ nicht vorherzusehen ist, muss die Möglichkeit zum bedarfsgerechten Zusammenbau der notwendigen Datenstrukturen aus einfachen Datentypen gegeben sein.

Hierfür enthält Mayday das Meta-Information-Framework, das Mechanismen zur Definition und Speicherung sogenannter *Meta Information Object*s (MIO) bietet (vgl. auch [GDN05]). Der Einsatz von MIOs ist in den Plugins sehr vielfältig und zahlreich.

Abbildung 4.4 zeigt einen Ausschnitt aus den Strukturen des MIO-Frameworks von Mayday. Der Übersichtlichkeit halber wird jedoch nur die Kerngruppe um `MIO` und `MIOType` gezeigt. Ein Beispiel für eine weitere, nicht dargestellte, wichtige Gruppe von Klassen und Schnittstellen im Kern ist die um `MIOTypeParser` zum Austausch, sowie zum Im- und Exportieren von MIOs.

Zentral für das MIO-Framework sind die abstrakte Klasse `MIO` und die beiden Schnittstellen `MIStructure` und `MIOType`. Letztere definieren Funktionen, die allgemein von jedem MIO zu erfüllen sind, während die abstrakte Klasse `MIO` bereits eine grundlegende Implementation eines Teils dieser Funktionalität vorhält. Alle realen, weiter spezialisierten MIO-Datentypen, wie `StringMIO`, `IntegerMIO`, `DoubleMIO`, `StringListMIO` (ein MIO, das eine Liste von Strings speichern kann) oder `StringMapMIO` (ein MIO, das eine Map von Strings

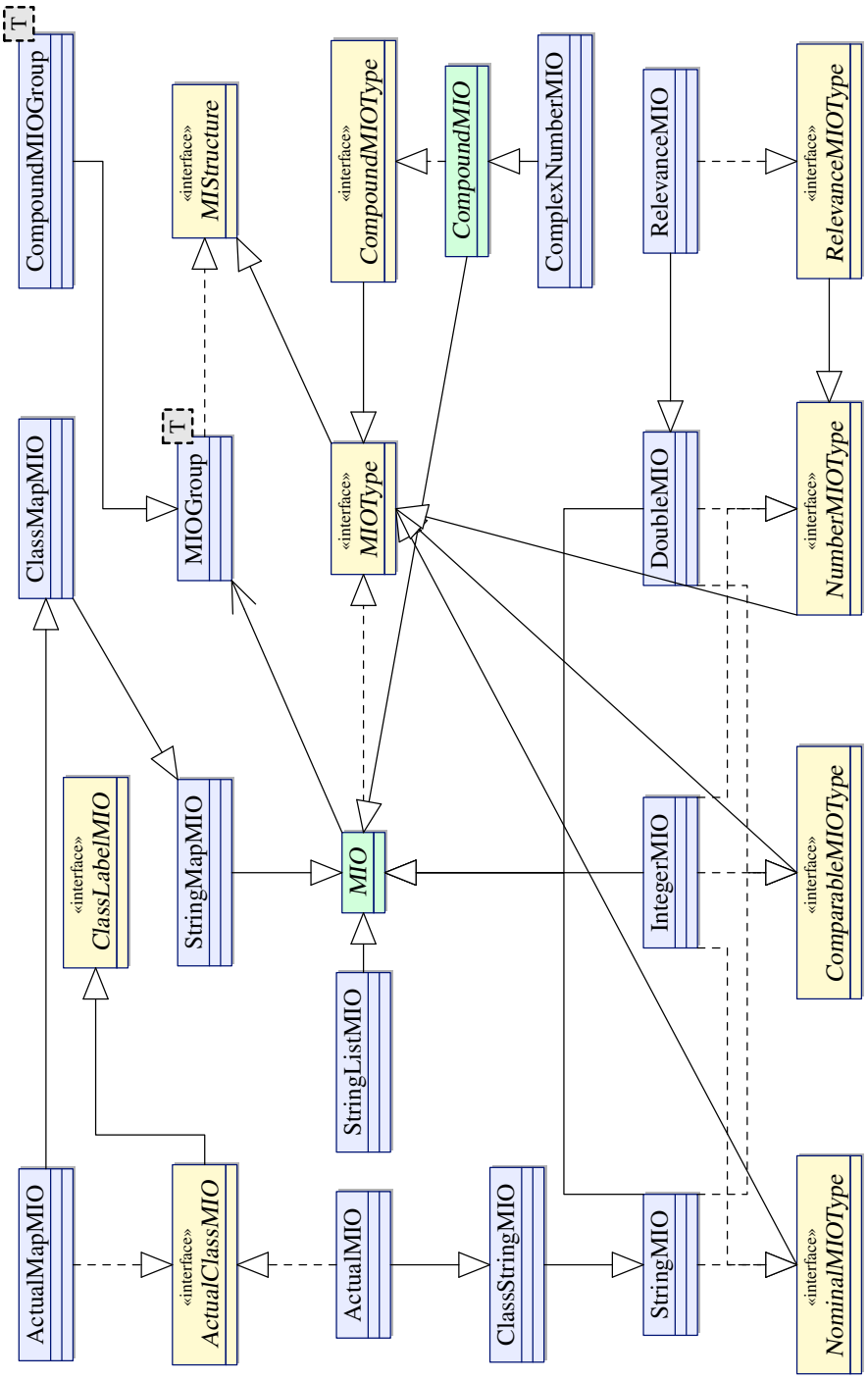


Abbildung 4.4: Das UML-Diagramm zeigt einen Ausschnitt aus dem MIO-Framework, dabei sind wesentliche, zentrale Strukturen, wie `MIO` und `MIOType`, abgebildet, aber auch abgeleitete, auf bestimmte Aspekte hin spezialisierte Klassen und Schnittstellen, z.B. `RelevanceMIO` und `RelevanceMIOType`. Eine genauere Erläuterung der dargestellten Klassen und Schnittstellen ist dem Text von Abschnitt 4.2.1 zu entnehmen.

speichert) erweitern `MIO` und überschreiben gegebenenfalls dessen Methoden in geeigneter Weise.

Spezielle Eigenschaften der Datentypen werden durch zusätzliche Schnittstellendefinitionen charakterisiert, beispielsweise `ComparableMIOType` für die Definition von Ordnungsrelationen, `NumberMIOType` für die Konvertierbarkeit des Typs in eine Zahl und `NominalMIOType` als reines Marker-Interface, und müssen von den MIO-Typen implementiert werden.

Zusammengesetzte, aus einfachen MIOs aufgebaute Datentypen werden durch das Marker-Interface `CompoundMIOType` gekennzeichnet. Ein Beispiel eines solchen MIOs stellt `ComplexNumberMIO` dar.

Abbildung 4.4 zeigt zwei Beispiele für eine sinnvolle Erweiterung bzw. Ergänzung vorhandener Datenstrukturen im Core. So dienen die Definitionen der Schnittstelle `RelevanceMIOType` und der Klasse `RelevanceMIO` der Konstruktion eines MIOs für einen Relevanzwert (vgl. [GDN05]), auf der Grundlage der Fließkommazahlen-Definition von `NumberMIOType` und `DoubleMIO`. Intensiven Gebrauch dieses Relevanz-MIOs machen die in [GDN05] veröffentlichten Visualisierungen.

Die Strukturen `ClassStringMIO`, `ActualMIO`, `ActualClassMIO`, `ClassLabelMIO`, `ActualMapMIO` und `ClassMapMIO` des Package `mayday.core.mi.classlabel` definieren, ausgehend von den String-basierten MIOs `StringMIO` und `StringMapMIO` (ebenfalls in `mayday.core.mi.classlabel`), Datenstrukturen für die Kennzeichnung von Datengruppen (*Class Labels*). Diese sind für das Data Mining und Machine Learning notwendig und spielen eine wichtige Rolle im zugehörigen Mayday-Plugin [Sym06]. Die beiden Schnittstellen `ClassLabelMIO` und `ActualClassMIO` sind leer und stellen ein Marker-Interface dar.

4.2.2 Die Pluginschnittstelle

Mayday verwendet in der aktuellen Ausbaustufe ein sehr einfaches Pluginkonzept (siehe Abb. 4.5), das vor allem durch die Implementierung der Methoden des Interfaces `Pluggable` realisiert wird. Die abstrakte Klasse `AbstractPlugin` bietet bereits eine Standardimplementierung allgemeingültiger Methoden aller Plugins. So ist die Funktionalität für die Kommunikation mit der in der Anwendung vorhandenen Instanz des `PluginManager`'s bereits ausprogrammiert, was das Anlegen eines passenden Menü-Eintrags mit der zugehörigen `RunPluginAction` umfasst. Daneben ist eine Struktur von `PluginAnnotation` vorgesehen, die auf der allgemeinen Klasse für Annotationen `Annotation` basiert und das Plugin näher beschreibt.

Gültige Plugins für Mayday implementieren somit notwendigerweise die Schnittstelle `Pluggable` und erweitern optional die Klasse `AbstractPlugin`. Die Abbildung 4.6 zeigt als erläuterndes Beispiel die Plugin-Klassen des Mayday Clustering-Plugins (Package `mayday.clustering`) mit ihren Verknüpfungen zu den zentralen Klassen des Pluginmechanismus in `mayday.core`.

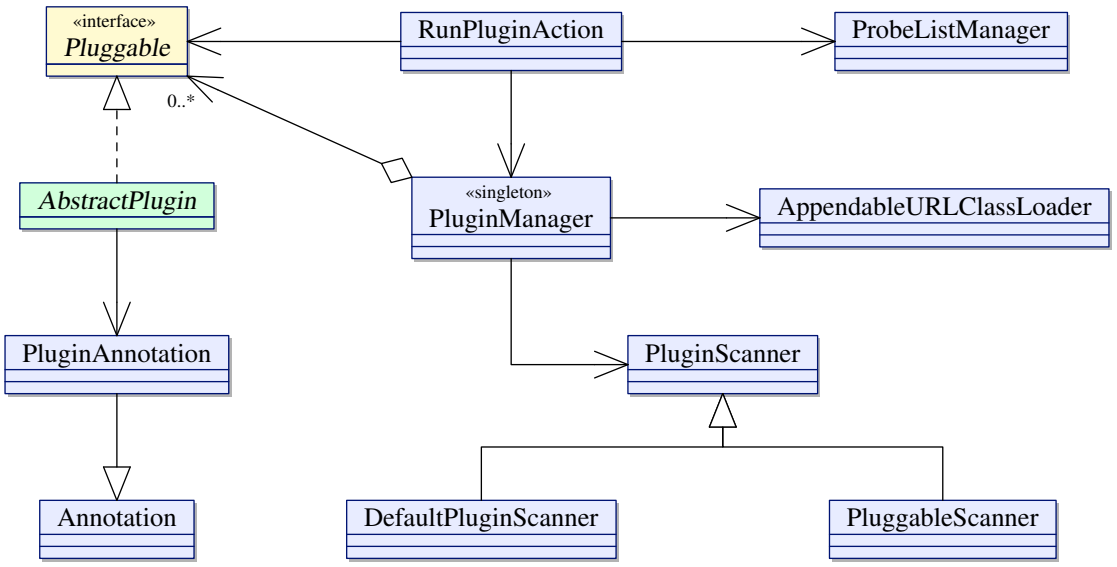


Abbildung 4.5: Übersicht über die an der Pluginschnittstelle von Mayday beteiligten wichtigsten Datenstrukturen. Die Klassen `ProbeListManager` und `Annotation` sind auch im Kontext anderer Konstrukte, z.B. Abb. 4.3, von Bedeutung und deuten hier den Bezug zum restlichen Framework an. Die in Mayday angelegte Instanz von `PluginManager` stellt ein Singleton für das komplette Framework dar.

Die einzelnen Plugins werden von der einzigen, Mayday-weit existierenden Instanz des `PluginManagers` (Singleton) mittels eines `PluginScanner`-Objektes gesucht, über Classloader geladen und verwaltet. Die Verwaltung der Plugins erfolgt in einer `Map`, die die Pluginnamen auf die zugehörigen Objekte abbildet.

Objekte von `RunPluginAction` kapseln die Ausführung eines Plugins in einen eigenen, nebenläufigen Thread und organisieren die Rückgabe der Ergebnisse.

`RunPluginAction` implementiert zusätzlich das Action-Interface der *Java Foundation Classes* (JFC) (`AbstractAction`), um sich in die übliche GUI-Infrastruktur von Java einzupassen.

4.2.3 Zusätzliche infrastrukturelle und fachbezogene Funktionalität des Core

Neben den beschriebenen wichtigsten zentralen Diensten der Datenhaltung und des Pluginmanagements bietet der Core von Mayday als weitere, allgemein verfügbare Funktionalität:

- `mayday.core.gui` :
  GUI-Elemente von Plugin-unabhängiger, allgemeiner Bedeutung, wie graphische Menüs und Elemente für die Darstellung und Verwaltung von Daten der Plugins und der Datensätze

- `mayday.core.gudi` :

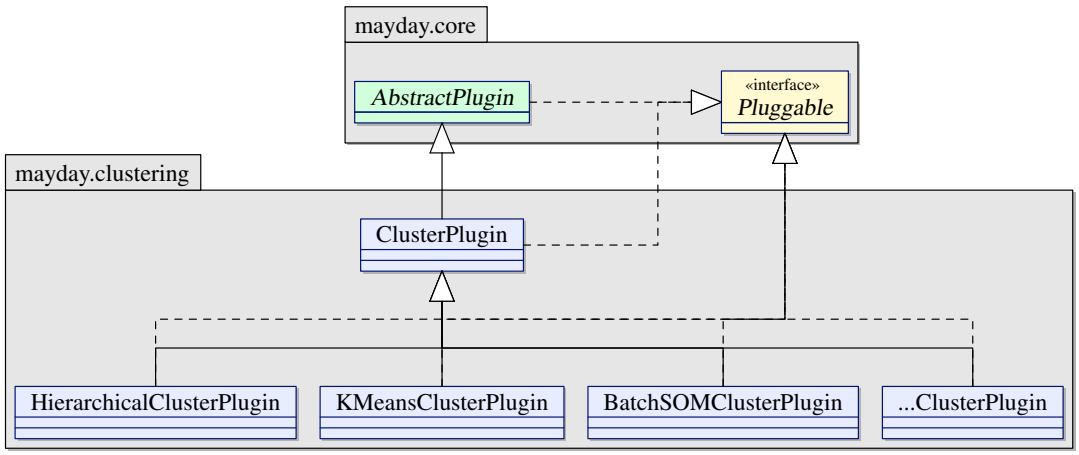


Abbildung 4.6: Überblick über die Organisation der Plugin-Klassen im Clustering-Plugin (Package `mayday.clustering`) für Mayday und deren Beziehungen und Relationen zu den Plugin-Klassen in `mayday.core`. Das Clustering-Plugin sammelt alle allgemein für Clustering-Algorithmen notwendige Funktionalität noch einmal in der Klasse `ClusteringPlugin`, von der dann die weiter spezialisierten Plugin-Klassen `****ClusterPlugin`, z.B. `BatchSOMClusterPlugin`, abgeleitet sind. Alle Klassen, die als Plugin fungieren können, müssen die Schnittstelle `Pluggable` implementieren.

Klassen für den Import und Export von Daten, sowie für das Sitzungsmanagement, wozu unter anderem die Persistenz von Sitzungszuständen gehört (Sessionmanagement) - siehe auch Abschnitt 4.3.11

- `mayday.core.misc.MathObjects` :
  Mathematische Definitionen, Strukturen und Algorithmen, die zentral, Plugin-übergreifend verfügbar sind und so als eine generelle, mathematische Bibliothek des Frameworks fungieren sollen; aktuell sind vor allem Elemente der linearen Algebra und der Statistik vorhanden

- `mayday.core.tasks` :
  Strukturen für die nebenläufige Abarbeitung von umfangreichen und rechenintensiven Arbeitsprozessen mit vorbereiteten GUI-Elementen zur Rückmeldung des Prozessfortschrittes und für die Interaktion mit dem Nutzer

4.3 Die Mayday-Plugins

Die Stärke der Mayday-Plattform liegt im flexiblen Ausbau ihrer Funktionalität durch Plugins, womit der wichtigste Teil der Zielstellung von Abschnitt 3.3.1 verwirklicht wird. Vor allem im Rahmen von studentischen Arbeiten, wie Studien- und Diplomarbeiten, sind bereits eine Reihe von Plugins entstanden. Über eine Auswahl davon soll ein kurzer Überblick und der jeweilige Bezug zu den in den Unterabschnitten von Abschnitt 3.3 genannten Entwurfszielen, sowie den in Abschnitt

4.1.3 aufgeführten Mayday-Komponenten gegeben werden (graphische Übersicht in Abbildung 4.7).

4.3.1 Visualization Plugin, Relevance Functions Plugin

Das Visualisierungsplugin geht auf [Geh03] zurück und implementiert eine Reihe statistischer Visualisierungen für die Datendarstellung, die insbesondere bei der Aus- und Bewertung von Microarraydaten eine wichtige Rolle spielen [DGN03]. Im Plugin sind ein Tabular View (Übersicht über die Zahlenwerte in Form einer Tabelle), eine einfache Heatmap, eine erweiterte Heatmap (*Enhanced Heatmap*, siehe [GDN05]), der PCA- (*Principal Components Analysis* (PCA)), der Box- und der Profilplot enthalten. Box- und Profilplot sind auch in Form eines Multiplots darstellbar, was besonders die vergleichende, explorative Visualisierung von Datenpartitionierungen bzw. Clusterungen sehr gut unterstützt.

Die Anwendung der Enhanced Heatmap greift auf die Unterstützung durch das *Meta Information Object* (MIO)-Framework und die Relevanzfunktionen des Relevance Function Plugins zurück. Über definierte Relevanzfunktionen können bestimmte Aspekte der Daten betont und hervorgehoben werden und stellt in diesem Sinne bereits einen Ansatz zur Visual Analytics dar.
Zielbereiche: Visualisierung, Visual Analytics, (Abschnitt 3.3.3)

4.3.2 Database Plugin

Das Database Plugin von Mayday bindet Mayday an eine dem MIAME-Standard angelehnte Datenbank an [Sym05]. Neben dem eigentlichen Plugin finden sich auf der Online-Quelle [DGS+08] auch SQL-Skripte für das Anlegen der Datenbank. Das zugehörige Datenbankschema stellt eine vereinfachte Form des MAGE-OM-Entwurfs dar. Primäres DBMS des Database Plugins ist PostgreSQL [Pos07].

Die Benutzung eines Datenbank-Backends schafft neben dem schnelleren und selektiveren Zugriff auf interessierende Daten weitere Vorteile, wie z.B. eine vereinheitlichte Datengrundlage und Datenzugriffsschicht über SQL für verschiedene Anwendungen mit SQL-fähiger Datenbankanbindung. Das verhindert unnötige Konvertierungen (Im- und Exportschritte) beim Austausch mit anderen Anwendungen.

Zusätzlich enthält das Database Plugin Routinen für den Import von Microrarray Daten im ImaGene- [Bio08b] und Affymetrix-Format [Aff08b].
Zielbereiche: Datenmanagement/DB, Datenaustausch/-IO, (Abschnitte 3.3.2 und 3.3.6)

4.3.3 Clustering Plugin

Das Clustering Plugin [DGN03] für Mayday bietet eine Reihe von Verfahren zur Partitionierung bzw. Clusterung der zu untersuchenden Daten und deren Visualisierung an. Enthalten sind hierarchisches Clustern, k-Means Clustering, *Self Organi-*

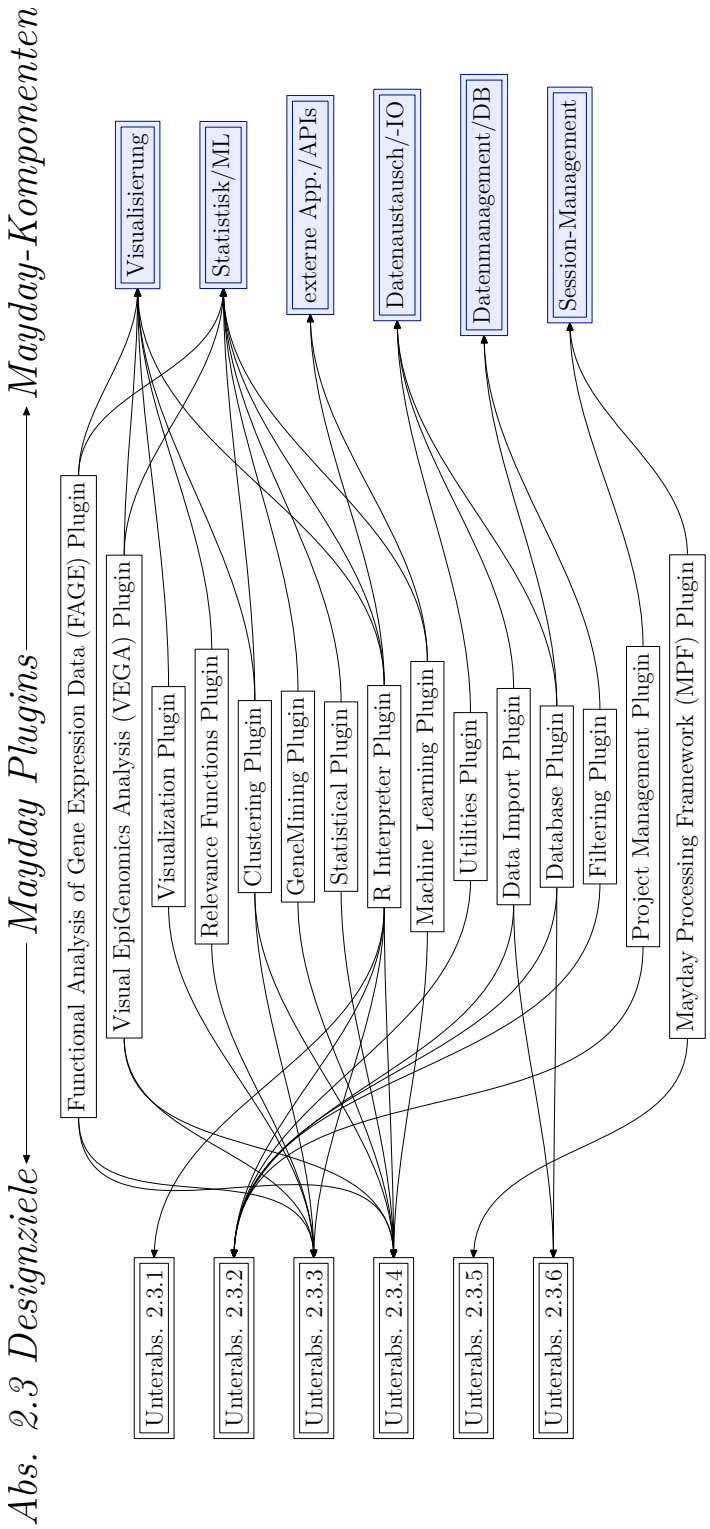


Abbildung 4.7: Graphische Übersicht über die Zuordnung der aufgezählten Plugins zu den einzelnen Entwurfszielen, die in den Unterabschnitten des Abschnitts 3.3 aufgeführt sind und den Komponenten von Mayday, die in Abschnitt 4.1.3 genannt wurden.

zing Maps (SOM) und *Density Based Spatial Clustering of Applications with Noise* (DBSCAN) [EKSX96]. Für das k-Means Clustering ist auch eine Variante für die interaktive, nutzerunterstützte Suche nach einem optimalen Wert für den Parameter k vorhanden. Die SOMs sind in Form des sogenannten Batch-SOM-Algorithmus implementiert [HTF01] und gestatten neben der Bestimmung der Größe und Form der genutzten Karten auch die Auswahl zwischen rektangulärer und hexagonaler Kartentopologie.

Für alle in diesem Plugin vorhandenen Clusterverfahren kann aus den in `mayday.core.misc.MathObjects.DistanceMeasures` implementierten Distanzen, zum Beispiel die euklidische, die Canberra- oder die Minkowski-Distanz, eine geeignete gewählt werden.

Neben der Möglichkeit, Clusterresultate mit dem in Abschnitt 4.3.1 erwähnten Multiplot über das Visualisierungsplugin darzustellen, enthält das Clustering Plugin eigene, verfahrensspezifische Visualisierungen für die Clusterprozeduren bzw. für die Beurteilung der Clusterresultate, beispielsweise die Dendrogrammdarstellungen für das hierarchische Clustern.
Zielbereiche: Statistik/ML, Visualisierung, (Abschnitte 3.3.3 und 3.3.4)

4.3.4 R Interpreter Plugin

Das R Interpreter Plugin [DZN04, Zsc04] erlaubt den Zugriff auf die reichhaltige Funktionalität von R [R D07] und seinen Erweiterungen, wovon im Zusammenhang mit Mayday BioConductor [GCB+04, Bio08a] am wichtigsten ist. Dadurch besteht einerseits die Möglichkeit, die in diesen Systemen bereits vorhandenen Verfahren für Statistik, Machine Learning und Visualisierung in Mayday zu integrieren und andererseits die guten und schnellen Entwicklungsmöglichkeiten neuer Verfahren in R für Domainenexperten innerhalb von Mayday zu erschließen - als eine Art „statistische Konsole“ im weitesten Sinne. Diese in R implementierten Verfahren lassen sich dann dem R-unkundigen, anwendenden, biologischen Labor- bzw. Klinikpersonal über ihre gewohnte, interaktive Mayday-GUI anbieten.

Für die Kommunikation zwischen dem nativ kompilierten R und dem *Java Virtual Machine* (JVM)-basierten Mayday wurden mehrere Möglichkeiten evaluiert und aufgrund dieser Ergebnisse eine einfache File-basierte Variante gewählt [DZN04], die sich als ziemlich robust erwies und die geringsten Anforderungen an die Laufzeitumgebung stellte. Trotzdem hat sich das R Interpreter Plugin als eines, der schwieriger zu installierenden Plugins erwiesen, insbesondere der Betrieb unter verschiedenen Betriebssystemen im Zusammenhang mit neuen R-, Java- und Betriebssystemversionen erfordert immer wieder Nachbesserungen.

Ein weiteres Problem der Verknüpfung zu nativem Code besteht in dessen Inkompatibilität mit Java-Web-Start [Sun08]. Aus diesem Grund und der vorher genannten Problematik besteht, trotz der existierenden und funktionierenden R-Anbindung, weiterhin Bedarf Funktionalität, die R oder BioConductor bereits besitzen, in einer nicht-nativen, Java-basierten Form in Mayday zur Verfügung zu stellen.

Zielbereiche: Visualisierung, Statistik/ML, externe App./*Application Programming Interfaces* (API), (Abschnitte 3.3.1, 3.3.2, 3.3.3 und 3.3.4)

4.3.5 Machine Learning Plugin

Bei Fragestellungen, die mit Hilfe von Microarrays untersucht werden, handelt es sich sehr oft um Klassifikationen zwischen verschiedenen Gruppen, zum Beispiel zwischen `gesund`/`krank` oder `Wildtyp`/ `Mutante1`/ `Mutante2`/`...`, oder die Bestimmung geeigneter Prädiktoren für diese Gruppen. Die Beantwortung solcher Fragen ist ein zentraler Bestandteil des Machine Learnings, das hierfür eine große Anzahl von anwendbaren Techniken entwickelt hat.

Das Machine Learning Plugin für Mayday [Sym06, SBD+06, SDN07] macht hierfür die Funktionalität der WEKA-Library [oW08, WF05] zugänglich, die verschiedene Klassifikations- und Feature-Selection-Verfahren enthält.

Obwohl ein großer Teil dieser Verfahren auch in R und damit über das im vorherigen Abschnitt 4.3.4 erwähnte R-Plugin für Mayday zugänglich sind, bietet das Machine Learning Plugin den Vorteile, dass es sich bei WEKA um eine Java-Library handelt. Sie läßt sich deshalb einfach zusammen mit Mayday distributieren und hält das Plugin zugleich kompatibel zu Java-Web-Start.
Zielbereiche: Statistik/ML, externe App./APIs, (Abschnitt 3.3.4)

4.3.6 Mayday Processing Framework (MPF) Plugin

Vielfach hat man bei der Auswertung von Daten aus Hochdurchsatzverfahren, wie sie Microarray-basierte Experimente geradezu prototypisch darstellen, eine Mischung aus explorativen und schematischen Analyseabschnitten. Erstere stellen die Hauptaufgabe bei der erstmaligen Analyse eines Experiments oder Experimenttyps dar, während letztere nach der Etablierung eines geeigneten und validierten Analyseablaufs für die Alltagsarbeit und die Bewältigung der auftretenden umfangreichen Datenmengen von großer Bedeutung ist.

Softwareseitig spiegeln sich diese beiden Aspekte in einer interaktiven, interpretierten Arbeitsform und einem automatisierten skript- bzw. pipelinefähigen Batchbetrieb eines Systems wieder. Der interaktive, interpretierte Modus ist in Mayday von vornherein über seine interaktive GUI vorhanden. Das MPF Plugin [SBD+06, Bat07b] rüstet den zweiten Aspekt für Mayday nach. Es stellt eine Art von graphischer Skriptsprache dar, in der sich Pipelines aus einzelnen Prozessmodulen zusammenstellen und auf Datensätze anwenden lassen. Der Zusammenbau der Pipeline erfolgt interaktiv innerhalb der GUI des MPF Designers. Die Pipeline selbst wird intern als Graph gehalten und dieser anhand der Randbedingungen des Datenflußes zwischen den Aus- und Eingängen der Prozessmodule validiert.

Die MPF Prozessmodule bilden die grundlegenden Funktionsblöcke (*Building Blocks*) des MPF. Neben den bereits vorgefertigten Prozessmodulen ist es für Experten einerseits möglich neue Prozessmodule in Java zu entwickeln, erstellte R-Skripte mit Hilfe eines kapselnden Prozessmoduls innerhalb des MPFs zugänglich

zu machen und andererseits für Anwender, neue Prozessmodule aus Zusammenstellungen vorhandener zu definieren (für Beispiele siehe [Bat07b]).
Zielbereiche: Session-Management, (Abschnitt 3.3.5)

4.3.7 Visual EpiGenomics Analysis (VEGA) Plugin

Das VEGA Plugin [ZDS+06, Zsc06] trägt der zunehmenden Bedeutung neu entwickelter Array-basierter Technologien Rechnung, wie das in Abschnitt 2.1.1 auf Seite 18 erwähnte ChIP-on-Chip Verfahren [HL04]. Im Besonderen unterstützt es die integrierte Analyse und Darstellung normaler Genexpressionsdaten zusammen mit epigenetischen.

Hierzu bietet VEGA die Möglichkeit Daten unterschiedlicher Quellen auf der Basis eines Distanzmaßes, wie der chromosomalen Lokation, zu alignieren und in einem gemeinsamen Rahmen, einer Trackliste, gestapelt als sogenannte Tracks darzustellen. Tracks bilden einen zweidimensionalen Darstellungsbereich für die Daten eines Experiments. Die vertikalen Relationen der Feature-Lokationen entlang eines Tracks wird mit Hilfe des Trackmaßes (Track Measure) bestimmt, das innerhalb einer Trackliste für alle Tracks gleich sein muß. Das horizontale Maß wird als Probe Measure bezeichnet, charakterisiert die Darstellung des Meßwertes für jedes Feature und gilt damit einheitlich für den kompletten Track.

Um möglichst flexibel verschiedene Darstellungen zu kombinieren, sind Visualisierung und Datenhaltung programmtechnisch getrennt - vergleichbar dem *Model View Controller* (MVC)-Paradigma. Die graphische Darstellung der Tracks erfolgt durch ein Track-Render-Objekt. Aktuell sind zwei verschiedene Varianten davon implementiert, ein Stem-Track-Renderer, der die Featuremeßwerte eines Tracks als eine Art Balkendiagramm visualisiert und ein Heatmap-Track-Renderer, der sie als einzeilige Heatmap darstellt.

Alle angesprochenen Strukturen befinden sich im Package `mayday.tracks` und sind für die Adaption an neue Daten und Darstellungen an einer Ergänzung bzw. Erweiterung der vorhandenen Klassenhierarchien ausgerichtet.

Eine Beispielanalyse von ChIP-on-Chip-Daten für CpG-Inseln und zugehörigen Genexpressionsdaten kann [Zsc06] entnommen werden.
Zielbereiche: Visualisierung, Statistik/ML, (Abschnitte 3.3.3 und 3.3.4)

4.3.8 Functional Analysis of Gene Expression Data (FAGE) Plugin

Eine Schwierigkeit bei der Detektion krankheitsspezifischer Expressionsunterschiede, vor allem im menschlichen Organismus, besteht darin, dass phänotypische Ausprägungen oft durch eine geringe Expressionsänderung von hochvernetzten Regulatorgenen verursacht wird. Diese kleinen Expressionsschwankungen erreichen bei üblichen statistischen, einzel-Gen-basierten Auswertungsverfahren oft nicht die notwendige statistische Evidenz, um als Effekt in einer Studie nachgewiesen zu werden.

Ein Ansatz dem zu begegnen und die Evidenzgrundlage für Expressionsunterschiede dieser hochinteressanten Gene zu erweitern, besteht darin eine statistische Analyse auf einer ganzen Gruppe von Genen durchzuführen. Die *Gene Set Enrichment Analysis* (GSEA) ist ein Beispiel einer solchen Methode [STM+05, SKG+07], die einen sogenannten *Enrichment Score* (*ES*) über der zugehörigen Gruppe an Genen, dem Gene Set, beurteilt. Dabei wird das entsprechende Gene Set je nach Anwendungsfall bzw. Anwendungsbereich aufgrund sekundärer, domainenspezifischer Informationen zusammengestellt, beispielsweise die Zusammenfassung aller Gene die an der Regulation eines Stoffwechselpfades beteiligt sind. Man beurteilt dann die gesamte Evidenz für eine Änderung dieses ganzen Stoffwechselpfades.

Das FAGE Plugin [ZDS+06, Deu06] integriert die Java-Bibliothek `gsea.jar` von [GSE08] und macht damit GSEA für Mayday verfügbar. Außerdem erfolgt eine Aufbereitung und Unterstützung bei der Auswahl passender Parameter, wie zum Beispiel einer geeigneten Metrik, sowie auch eine aufbereitende Unterstützung für die Beurteilung der Ergebnisse. Zusätzlich bietet FAGE die Anwendung der GSEA auf Fälle mit nur einem Phänotyp mit Hilfe des *Rank Product*s (RP) als Metrik.

Ein in FAGE integrierter Pathway-Viewer visualisiert Stoffwechselpfade als Netzwerk und annotiert die Knoten mit den zugehörigen Genexpressionswerten. Zur visuellen Betonung werden die Knoten entsprechend dieser Genexpressionswerte farblich kodiert dargestellt. Die Farbkodierung kann vom Anwender, ähnlich der in der Heatmap des Visualisierungsplugins, angepaßt werden.
Zielbereiche: Visualisierung, Statistik/ML, (Abschnitte 3.3.3 und 3.3.4)

4.3.9 GeneMining Plugin

Da, wie schon im vorherigen Abschnitt 4.3.8 erwähnt, mit Hilfe von Microarrayexperiment der Bezug zwischen zwei phänotypischen Klassen zu bestimmten Gengruppen gefunden werden soll, tritt oft die Frage nach der Gruppe von Genen auf, die die beiden Klassen am besten bzw. signifikantesten trennt. Im Bereich des Machine Learning bezeichnet man das als Feature- oder Merkmalsselektion. Um eine valide, sich auf die biologische Grundgesamtheit bzw. Population verallgemeinerbare Schlußfolgerung zu erzielen, muß diese Merkmalsselektion robust sein. Das bedeutet sie muß recht unempfindlich gegenüber den spezifischen Gegebenheiten einer einzelnen Studie (u. a. dem Störpegel bzw. dem Rauschen) und den spezifischen Eigenheiten des einzelnen, verwendeten Auswertungsverfahrens sein.

Das GeneMining Plugin [SSD+07, Sch07] stellt hierfür eine robuste Auswahlprozedur mit mehreren, verschiedenen Auswahlkriterien zur Verfügung. Neben allgemein bekannten und genutzten Kriterien, wie die t-Statistik, *Information Gain* (IG), *Statistical Analysis of Microarrays* (SAM) oder eindimensionalen *Support Vector Machine*s (SVM), ist auch das neue robuste Kriterium *Quartet Mining* (QM) enthalten. Es stellt eine Adaption des in der Phylogenie bekannten Quartet Mappings [NSvH01] für die robuste Merkmalsselektion bei binären Klassifikationen dar. Ausgangspunkt ist die hierarchische Clusterung aller erhobenen Proben (Organismen, Zustände oder Gewebe) innerhalb eines Dendrogramms anhand ihrer Genex-

pressionsprofile. Unterscheiden sich die beiden Klassen K_1 und K_2 in irgendeiner Weise in ihren Genexpressionsprofilen, so läßt sich im Dendrogramm ein Split zwischen zwei Teilbäumen T_1 und T_2 finden, der die Realisierungen der beiden Klassen trennt, $K_1 \sim T_1$ und $K_2 \sim T_2$. Gegenstand des QM ist die Identifikation der für diesen Split verantwortlichen Expressionsprofile bzw. eines Rankings der Expressionsprofile entsprechend ihrer Bedeutung für diesen Split.

Verknüpft man die gefundenen Features mit biologischem Vorwissen, z.B. Stoffwechselpfaden oder ähnlichem, kann Merkmalsselektion neben der Identifikation geeignter Features für die Konstruktion eines Prädiktors oder Klassifikators, wichtige Hinweise auf die, der phänotypischen Ausprägung zugrunde liegenden, biologischen Mechanismen geben. Als besonders robust und erfolgsversprechend haben sich in [Sch07] die beiden Kriterien QM und SAM erwiesen.
Zielbereiche: Statistik/ML, (Abschnitt 3.3.4)

4.3.10 Statistical Plugin

Das Statistik Plugin von Mayday implementiert im Package `mayday.statistics` Methoden zur Berechnung grundlegender statistischer Kennzahlen eines Featureprofils, wie zum Beispiel den Mittelwert, die Varianz oder die Standardabweichung. Neben diesen finden sich noch Methoden zur Klasseneinteilung numerischer Daten oder zum Berechnen der Pearson-Korrelation zwischen zwei Profilen im Package.

Package `mayday.statistics.inference` ist als Container für statistische Inferenzverfahren gedacht und bietet aktuell mehrere Formen des t-Tests [SH06] an:

- Ein- oder Zweistichprobentest
- einseitig oder zweiseitig
- gepaarter oder ungepaarter Zweistichprobentest
- homoskedastisch (Student'scher t-Test [SH06]) oder heteroskedastisch (Welch-Test [Wel38, Sat46, Wel47])

Die Resultate der Routinen werden als MIOs abgelegt und stehen so für die weitere Verarbeitung bzw. Darstellung zur Verfügung - letzteres speziell über die zusätzliche Gewichtung mit Hilfe der Relevanz-Funktionen des Relevance Function Plugins (Abschnitt 4.3.1).
Zielbereiche: Statistik/ML, (Abschnitt 3.3.4)

4.3.11 Project Management Plugin

Das Project Management Plugin stellt den Ausgangspunkt für ein Session-Management in Mayday dar, d.h. für das Sichern der bearbeiteten Datensätze im aktuellen Bearbeitungszustand, das Abspeichern einer entsprechenden Bearbeitungshistorie und dem Wiederherstellen des Bearbeitungszustandes aus diesen gesicherten Daten. Implementiert ist zur Zeit das Sichern und Wiederherstellen der Daten.

Die eigentliche Funktionalität des Plugins befindet sich momentan jedoch im Wesentlichen im Kern-Package `mayday.core.gudi`. Über die Elemente des Packages ist das Sichern und Laden der bearbeiteten Datensätze in Form von *Mayday Workspace Snapshots* möglich (ZIP-gepackte XML-Serialisierungen der Datenstrukturen), außerdem der Export in das textbasierte *Mayday Tabular Format*, `*.mtf`.
Zielbereiche: Session-Management, (Abschnitt 3.3.2)

4.3.12 Data Import Plugin, Filtering Plugin, Utilities Plugin

Das Data Import Plugin, das Filtering Plugin und das Utilities Plugin bieten diverse Funktionen, um Daten zu importieren oder geeignet zu bearbeiten. So wird der Import von MIOs einfacher Datentypen, beispielsweise Gleitkommazahlen, Ganzzahlen, Zeichenketten und Zeichenkettenlisten unterstützt, aber auch der zusammengesetzter Datentypen, wie komplexer Zahlen.

Das Filtern nach vorgegebenen Indizes oder einfachen Profilen ist möglich, wie auch der Export in das Nexus- oder Phylip-Format, um Daten für die Phylogenieprogramme `PAUP*` und `PHYLIP` zu exportieren und mit Expressionsphylogenien zu experimentieren (vergl. [Rie05]). Daneben existieren Funktionen zur Bildung von Schnitt- und Vereinigungsmengen ausgewählter Array-Features.
Zielbereiche: Datenaustausch/-IO, (Abschnitte 3.3.2 und 3.3.6)

4.4 Organisatorische Aspekte und Randbedingungen bei der akademischen Softwareentwicklung umfangreicher Systeme in der Bioinformatik

Die Problematik der Entwicklung wissenschaftlicher Software hat ihre spezifischen Besonderheiten und muß, gerade weil daraus oft weitreichende Schlüsse gezogen werden, kritisch hinterfragt werden. Ein Aspekt der trotz seiner Offensichtlichkeit und Unumgänglichkeit im wissenschaftlichen „Software-Alltag“ sehr oft zu wenig Beachtung findet, ist die Korrektheit bzw. Genauigkeit der zugrunde liegenden Berechnungen (siehe hierzu die Ausführungen zum Gleitkomma-Standard IEEE 754 in Abschnitt 5.3.1 und [KD98]). Aber neben diesen vergleichsweise „harten“, eingängigen Aspekten gibt es „weichere“, organisatorische, die ab einer bestimmten Anwendungskomplexität unbedingt zu beachten sind.

4.4.1 Software-Engineering im Umfeld akademischer Softwareentwicklung

Vor allem beim Anwachsen des Umfanges eines Software-Projektes, seiner wachsenden inneren Abhängigkeiten und Vernetzungen, sowie des sich vergrößernden entwickelnden Personenkreises (möglicherweise noch externer, über das Internet

entwickelnden Personen) und einer zunehmenden personellen Fluktuation, schlagen die graduellen, organisatorischen Anforderungen in eine neue Qualität um und spielen eine wichtige Rolle auf die methodisch reagiert werden muß.

Language	files	blank	comment	code		scale		3rd gen. equiv
Java	2136	73349	85589	277757	x	1.36	=	377749.52
HTML	361	11085	5850	85203	x	1.90	=	161885.70
XML	139	4732	6381	75564	x	1.90	=	143571.60
SQL	17	294	138	2235	x	2.29	=	5118.15
DTD	10	72	24	232	x	1.90	=	440.80
CSS	2	15	7	38	x	1.00	=	38.00
Bourne Shell	3	4	0	9	x	3.81	=	34.29
DOS Batch	3	2	0	5	x	0.63	=	3.15
SUM:	2671	89553	97989	441043	x	1.56	=	688841.21

Tabelle 4.1: LoC-Metrik des kompletten ausgechekten Mayday-Repositorys (Kern und Plugins) als Ausgabe des Tools CLOC [Al 08]. Angegeben ist die Anzahl an Files, deren Typ und die darin enthaltenen Code-, Kommentar- oder Leerzeilen. Die Umrechnung in die hypothetische Sprache der dritten Generation „3rd gen. equiv“, dient lediglich Vergleichszwecken und setzt eine ähnliche Komplexität wie C++ an - d.h. scale(C++) = 1.00.

Tabelle 4.1 enthält ein grobe Abschätzung des Umfangs von Mayday mit Hilfe der *Lines of Code*s (LoC)-Metrik. Die Abschätzung mittels LoC ist sehr umstritten und diskussionfähig, sie soll an dieser Stelle jedoch lediglich zur groben Einordnung der Größe von Mayday als Softwareprojekt dienen. Die Werte wurden mit Hilfe von CLOC [Al 08] auf einem Snapshot des vollständig ausgecheckten Mayday-Repositorys (Plugins und Kern) gewonnen. Zum Vergleich sind in Tabelle 4.2 die auf [Al 08] angegebenen Summenwerte für MySQL 5.0.24a, PostgreSQL 8.1.4, Perl 5.8.8 und SQLite 3.3.7 aufgeführt.

Language	files	blank	comment	code		scale		3rd gen. equiv
mysql-5.0.24a	3076	214148	276063	1245478	x	1.44	=	1794575.59
postgresql-8.1.4	2482	102369	159109	936275	x	1.43	=	1339742.23
perl-5.8.8	1952	101643	134554	427441	x	2.69	=	1149231.15
sqlite-3.3.7	145	10757	26980	96217	x	1.65	=	159163.19

Tabelle 4.2: Summenwerte der LoC-Metrik für bekannte Open-Source-Projekte, wie sie auf [Al 08] angegeben sind. Zu beachten ist, bei einem Vergleich mit Hilfe der hypothetischen Sprache der dritten Generation, die unterschiedliche Skalierung bei der Umrechnung der verwendeten Programmiersprachen. Skriptsprachen wie Perl erhalten dabei einen sehr hohen Umrechnungsfaktor von scale(Perl) = 4.00.

Im wissenschaftlichen Umfeld der Softwareentwicklung, allen voran in Gebieten

jenseits der Kerninformatik, existiert dieser Problematik gegenüber eine noch immer mangelnde Sensibilität, wie [Wil06] sehr gut darlegt. Insbesondere machen es die schnellen Entwicklungszyklen, der Wechsel der Paradigmen, sowie die Fülle und Verstreutheit an Informationen und methodischen Vorgehensweisen aufwändig und schwer den nötigen Überblick zu erlagen. Infolge der in [Wil06] beschriebenen Erfahrungen hat dessen Autor eine umfangreiche Veröffentlichung [Wil08] mit methodischen Vorgehensweisen und Ratschlägen zugänglich gemacht.

Von Vorteil bei der Softwareentwicklung in der Bioinformatik ist der vorhandene, ausbildungsbedingte leichte Zugang von Bioinformatikern zu modernen Methoden des Software-Engineering. Allerdings entsteht ein großer Teil akademischer Software durch die Mitwirkung von Studenten in Form von Studien-, Diplom-, Bachelor- oder Masterarbeiten. Jedoch selbst die bei der Mayday-Entwicklung gemachten Erfahrungen sind derart, dass die Mehrheit der Bioinformatik-Studenten in diesen Arbeiten erstmals mit den Methoden konfrontiert sind, die bei der Entwicklung großer Systeme auftreten. Speziell Erfahrungen in der Benutzung eines Versionierungswerkzeuges zum *Software Configuration Management* (SCM), wie CVS [CVS08] oder Subversion (SVN) [The08], als ersten Schritt zu einer organisierten Softwareentwicklung ist noch erheblich unterrepräsentiert. Ansätze die Curricula diesbezüglich mit langfristigeren Praktika zu umfangreicheren und komplexeren Projektzielen anzureichern, sollten hier bald Wirkung zeigen.

Dabei stellt sich jedoch die Frage nach der Verwendung eines Versionsmanagements nicht erst ab einer Softwareentwicklung im Team, sondern gehört bei modernen, agilen Entwicklungsprozessen, zum Beispiel dem des *Extreme Programming*s (XP), bereits zur Einzelentwicklung. Aber spätestens ab einer Entwicklung im Kollektiv muß eine entsprechende Unterstützung durch ein Werkzeug erfolgen, das Mehrfachzugriff erlaubt, die Änderungsgeschichte protokolliert und bei der Lösung von Änderungskonflikten hilft.

Neben den Herausforderungen, die sich jedem Projektteam in gleicher Weise stellen - sei es in der industriellen oder in der akademischen Softwareentwicklung, verschärft sich bei größeren und damit langfristigeren Projekten im akademischen Umfeld der Aspekt der personellen Fluktuation von Entwicklern.

Die Examensarbeiten, deren Gegenstand in der Bioinformatik häufig die Implementierung und anschließende Evaluierung eines Verfahrens darstellen, haben Laufzeiten von maximal einem halben bis zu einem Jahr, unter Beachtung aller notwendigen Vor- und Einarbeitungszeiten. In dieser Zeit muß das geplante Verfahren im Rahmen der Examensarbeit programmiert, lauffähig gemacht, angewendet und beurteilt werden. Im Verhältnis zur gesamten Projektlaufzeit einer umfangreichen akademischen Software bedeutet das einen sehr kurzen zeitlichen Rahmen, den der Entwickler dem Projekt zur Verfügung steht und bedingt die hohe personelle Fluktuation.

Als Folge dieser Fluktuation müssen ständig neue Mitglieder des Projektteams eingearbeitet und die Wartung und Weiterentwicklung des Codes ausscheidender Teammitglieder sichergestellt werden. Das fordert die Organisation eines ständigen Wissenstransfers.

Auch in der industriellen Softwareentwicklung steht man diesem Problem gegenüber, jedoch in nicht so starker Form, da der Zugriff einer Firma auf ihre Programmierer im Allgemeinen langfristiger ist. Im Rahmen des Paradigmas der *Agilen Softwareentwicklung* wird dieses Problem mit dem Prinzip des „kollektiven Codebesitzes“ (Collective Code Ownership) beantwortet. Die dort vorgesehenen Lösungsansätze sehen die umfassende Verwirklichung eines *Pair Programming*, d.h. jede Aufgabe wird von zwei Entwicklern gemeinsam bearbeitet, und umfangreiche *Code Reviews* vor. Beides sind adäquate Methoden, sofern man die dafür notwendigen hohen personellen und zeitlichen Ressourcen aufbringen kann. Dies ist im akademischen Umfeld nur schwer umsetzbar.

Insgesamt ist die akademische Softwareentwicklung mehr der Form der üblichen akademischen Arbeiten angepaßt, d.h. sie ist eher explorativ, soll das Funktionieren neuer Verfahren demonstrieren und baldmöglichst eine Veröffentlichung flankieren. Es findet generell eine schnellere, erkundende, prototypenhafte Entwicklung statt, bei der deshalb häufig sowohl die Testabdeckung, die Nutzer- bzw. Anwendungsdokumentation wie auch die Entwicklerdokumentation zu kurz kommen. Letztere stellt einen weiteren wichtigen Punkt dar, der den oben erwähnten Wissenstransfer zwischen alten und neuen Entwicklern fördern würde und erstere ist für eine breite Anwendung der Applikation überaus wichtig. Die beste Funktionalität nutzt nichts, wenn keiner von ihrem Vorhandensein weiß.

Ein weiterer Ansatzpunkt die erwähnte Problematik etwas abzumildern unterstützt Mayday bereits von seiner Architektur her, durch die starke Modularisierung in einzelne Komponenten kann zumindest der Einarbeitungsaufwand neuer Teammitglieder vermindert werden.

4.4.2 Aspekte der Entwicklung im Team

Mayday wurde als ein Teamprojekt von Anfang an mit Hilfe eines zentralen, CVS-verwalteten Repositorys entwickelt. Die bekannten Nachteile von CVS, speziell bezüglich der Versionierung von Verzeichnissen, wurden durch die sehr gute, standardmäßige Unterstützung von CVS in der genutzten Hauptentwicklungsplattform Eclipse [Ecl08a] wettgemacht. Weitere Probleme des CVS-Nachfolgers Subversion beim File-Locking der Berkeley-DB [Ora08], die lange Zeit das einzige Datenbackend von Subversion darstellte, auf *Network File System* (NFS)-Shares, verhinderten zusätzlich einen frühen Übergang auf Subversion.

Ein weiterer wichtiger Punkt bei der verteilten Entwicklung ist die möglichst schnelle Integration der getrennt entwickelten Codeteile, um so auftretende Probleme divergierenden Quellcodes früh und zeitnah zu entdecken. Dieser Prozess muß regelmäßig und automatisiert erfolgen. Für das Mayday-Projekt bedeutet das die Überwachung des Source-Codes aller Plugins und des Kerns und das Anstoßen der entsprechenden Build-Prozesse bei vorgefundenen Änderungen. Die Realisierung erfolgte durch die kooperative Anwendung von CruiseControl [Cru08], einem in Java geschriebenen Werkzeug für die kontinuierliche Integration (*Continuous Integration*) in Erstellungsprozessen von Software, und Ant [Apa08], einem in Java-

implementierten Kommandozeilenprogramm für die Prozessautomatisierung. CruiseControl läuft in einem Serverprozess und überwacht dabei das zentrale, interne Repository für Mayday und seine Plugins alle 6 Stunden auf Änderungen. Treten Änderungen auf, so stößt es über Ant-Build-Skripte die zugehörigen Build-Prozesse an. Treten beim Bauen der geänderten Code-Bestandteile Probleme auf, so wird neben dem Entwickler, der diese Änderungen ins Repository eingepflegt hat, auch der Projektstab über das Fehlschlagen des Build-Prozesses per Mail informiert. Ist die Kompilation erfolgreich, so wird nachfolgend der Neubau der Java-Web-Start-Komponenten und des Installers ausgeführt. Anschließend erfolgt der Upload auf die Mayday-Webseite [DGS+08].

4.4.3 Anwenderaspekte

Wichtig ist es, auch bei der Entwicklung akademischer Software, den anvisierten Anwender der Software im Blick zu behalten. Entscheidend ist dabei die Frage, welche IT-Kenntnisse von einem Anwender dieser Software zu erwarten sind. Handelt es sich um Experten, die nicht nur generell eine hohe Affinität zu Computern haben, sondern auch eine höhere Motivation, sich durch schlechte bzw. mangelhafte Dokumentationen und Kommandozeilenoptionen durchzukämpfen. Oder handelt es sich eher um fachfremdere Anwender mit weniger Bezug zu Rechnern, die möglichst selbsterklärende GUIs erwarten und sich schnell wieder abwenden, wenn sie den Zugang zur gewünschten Funktionalität nicht auf Anhieb finden.

Mayday sieht sich beiden Kollektiven gegenüber und muß beiden gerecht werden, wie auch in Abschnitt 3.3.5 dargelegt wurde. Sowohl der „ambitionierten Bioinformatiker“ wie auch der im Labor arbeitenden Biologe, der schnell mal ein Tool sucht, um seine Routineaufgaben abwickeln zu können, soll in Mayday das geeignete Werkzeug finden.

Damit sind aber schon die Hürden, die vor dem erstmaligen Einsatz einer Software wie Mayday liegen von erheblicher Bedeutung. Mayday bietet hier zwei einfache und bequeme Wege. Einerseits per Java-Web-Start über das Internet ohne Installation oder der Installation auf dem eigenen Rechner per Installer des kompletten Maydays.

Aufgrund der Breite an unterstützten Betriebssystemen fiel die Wahl des Tools zum Bau des Mayday-Installers auf den InstallBuilder von BitRock [Bit08]. Er ist für Open-Source Projekte frei verfügbar, läßt sich per XML konfigurieren und kann aus dieser Basis heraus Installer für verschiedenste Betriebssysteme generieren.

Anhand der Rückmeldungen von Mayday-Nutzern, war insgesamt eine in den letzten Jahren gestiegene Erwartung an die Anwenderfreundlichkeit von Open-Source-Software seitens der Nutzer zu verzeichnen, dem auch Rechnung getragen werden muß. Die Akquirierung eines großen Anwenderkreises hebt nicht nur die Wirkung (Impact), die eine Software erreicht, es sorgt – durch die weitergehende Austestung des Codes – auch für eine besser Qualität der Software. Weiterhin erhöht ein breiter Anwenderstamm die Attraktivität des Softwareprojektes für freie Entwickler und befördert damit die Gewinnung externer Entwickler.

4.5 Diskussion und Ausblick

Mayday hat in der hier beschriebenen Entwicklungsstufe bereits einen Reifegrad erreicht, der den in Abschnitt 3.3 genannten Zielen gerecht wird. Dies belegt nicht nur die Verleihung des *bwcon Sonderpreis IT & Life Sciences 2006* des Landes Baden Württemberg [bwc06], sondern auch die stetig wachsende Zahl an Rückmeldungen von externen Mayday-Nutzern. Dabei erfolgt der Einsatz von Mayday auch jenseits der Bioinformatik in unerwarteten Bereichen, wie beispielsweise in einem Ingenieurbüro der Deutschen Bahn AG.

4.5.1 Der Mayday-Kern

Die in Abschnitt 4.2.2 beschriebene Plugin-Schnittstelle von Mayday erleichtert durch ihre Einfachheit die schnelle Einarbeitung neuer Entwickler und damit das unkomplizierte Schreiben von Mayday-Plugins, beispielhaft sei hier das Schreiben eines Mayday-Plugins als Übungsaufgabe zur Vorlesung „Algorithmen der Bioinformatik“ genannt.

Auf der anderen Seite schränkt diese Einfachheit aber auch die Funktionalität der Schnittstelle stark ein. Gerade das Wachsen der Zahl an Plugins und das immer stärker Auftreten von Abhängigkeiten untereinander, bis hin zu Abhängigkeiten zwischen einzelnen Revisionen von Plugins, erfordern ein umfangreicheres Plugin-Modell. Dieses Modell sollte in der Lage sein diese Abhängigkeiten abzubilden (Revisionsmanagment) und damit das korrekte Funktionieren der Plugins sicherzustellen. Gegebenenfalls sollte auch der Betrieb unterschiedlicher Revisionen ein und desselben Plugins mit korrekten Zuordnungen möglich sein, falls die Plugin-Abhängigkeiten dies verlangen.

Ein typischer Fall für Plugins die Abhängigkeiten, insbesondere auch die von einzelnen Revisionen, implizieren sind reine Bibliotheken-Plugins. Ein Beispiel für ein solches Plugin in Mayday wäre eine mathematische Bibliothek in Form eines Plugins, die alle notwendigen mathematischen Algorithmen für Mayday global zur Verfügung stellt. Eine solche Plugin-Bibliothek würde keine dem Nutzer in der Mayday-GUI zugängliche Funktionalität anbieten, sondern als reines „Helfer-Plugin“ die Funktionsfähigkeit anderer Plugins gewährleisten. Im Kontext des aktuellen Entwicklungsstandes von Mayday stellt dies eine neue Art von Plugins dar. Momentan erfolgt das globale Zurverfügungstellen mathematischer Algorithmen und Strukturen über das im Kern gelegene Package `mayday.core.misc.MathObjects`. Eine Auslagerung in ein eigenes Plugin würde mehr Plugin-Entwicklern das Integrieren der mathematischen Routinen, die ihre Plugins benötigen und verwenden, in diese Bibliothek erleichtern. Sie müßten sich nicht zugleich mit dem Rest des Kerns beschäftigen und die Verwendung wäre ganz analog zu einer normalen Bibliothek. Als Nebeneffekt könnte dadurch der mathematischer Code zentral wartbar und für alle Plugin-Entwickler verfügbar gemacht werden, was unnötige Code-Replikation und damit auch unnötige Fehlerquellen vermeiden hilft. Letzteres wird dadurch zusätzlich unterstützt, dass ein solcher zentraler Code-Bestandteil

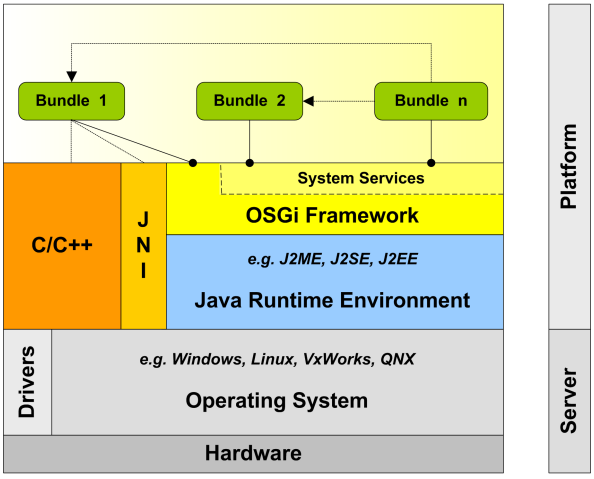


Abbildung 4.8: Schichtenmodell des OSGi-Standards (Quelle [Gra06]).

besser und mit höherer Code-Überdeckung (*Coverage*) testbar ist bzw. durch seine mehrfache Verwendung automatisch mehr getestet wird. Im momentanen Entwicklungszustand ist ein großer Teil der mathematischen, statistischen und Machine Learning-Routinen über die einzelnen Plugins verteilt.

Ein großes Problem bei der Auswertung von Array-basierten Life-Science-Daten stellt deren großer Umfang dar. Besonders Mayday ist als Java-basierte Anwendung in Bezug auf Speicherplatz und Rechengeschwindigkeit mit den Grenzen konfrontiert, die die Entwicklung in dieser Sprache setzt (mehr hierzu und möglichen Optimierungen innerhalb von Java in Kapitel 5). An dieser Stelle wäre es von großem Nutzen für sehr umfangreiche Datenmengen nahtlos auf speziell optimierten, leistungsfähigen nativen Code auszuweichen, wie beispielsweise auf die in Kapitel 6 vorgestellte Anwendung SpRay zur visuellen Datenexploration. Sehr hilfreich wäre es, bereits im Pluginmechanismus eine Möglichkeit zur mühelosen Integration von „Hochdurchsatzplugins “ mit nativem Code vorzusehen. Dabei kommt dann jedoch die konkrete unterliegende Architektur von Betriebssystem und Hardware zum Tragen, was den Mechanismus durch eine weitere Abhängigkeitsebene verkompliziert.

Der Entwurf eines für die obigen Zwecke hinreichend umfangreichen Plugin-Modells, seine technische Realisierung und Testung stellt einen sehr hohen Aufwand dar, weshalb der Rückgriff auf bereits etablierte Standards notwendig ist. Im Java-Umfeld hat sich in den letzten Jahren das dynamische, SOA-orientierte Komponentenmodell der *Open Services Gateway initiative* (OSGi) [OSG08, OSG07] mehr und mehr als generell akzeptierter Standard durchgesetzt. Gerade mit dem Wechsel von Eclipse [Ecl08a] seit Version 3.0 auf die OSGi-Plattform als zugrunde liegendes Komponentenmodell, hat es sich auch für Desktopanwendungen etabliert. Das Eclipse-Projekt Equinox [Ecl08b] stellt eine freie Implementierung des aktuellen Standards OSGi R4.1 dar. Eine weitere freie Implementierungen des OSGi-Standards ist Knopflerfish [Mak08].

Die distributierbaren Komponenten von OSGi werden als Bundle bezeichnet

(vgl. Abbildung 4.8). Neben einem Life-Cycle-Management, einem Sicherheits- und Kommunikationsmodell für Bundles sind grundlegende, administrative Services in der „OSGi Service Platform - Core Specification“ und weitere wichtige in „OSGi Service Platform - Service Compendium“ definiert [OSG07], zum Beispiel die *Log Service Specifiaction* oder die *HTTP Service Specification*. Bundles können anhand einer zentralen Service-Registry zur Laufzeit, d.h. dynamisch, neue Services anmelden, nach Service-Objekten suchen oder sich über Änderungen von Service-Objekten informieren lassen. Eingeschlossen ist auch die Behandlung nativer Code-Bestandteile von Bundles (S. 57 in „OSGi Service Platform - Core Specification“ von [OSG07]), sowie Mechanismen zum Update-Management über das Netz.

OSGi stellt somit eine Lösung für alle mit Mayday auftauchenden infrastrukturellen Herausforderungen und Probleme dar, inklusive einer sehr guten Skalierbarkeit in Bezug auf die Anzahl der eingesetzten Plugins. Auf der anderen Seite stellt der Aufwand für die Einarbeitung in die Spezifikation und deren Komplexität einen nicht zu unterschätzenden Nachteil dar. In den letzten Jahren ist jedoch durch die sehr breitenwirksame Anwendung von OSGi im Eclipse-Umfeld die Unterstützung der Softwareentwickler durch Entwicklungstools innerhalb der Eclipse-*Integrated Development Environment* (IDE) sehr gewachsen, so daß ein Teil der Komplexität durch diese Tools vor dem Entwickler verborgen wird. Auch für Knopflerfish existiert ein Eclipse-Plugin, das die Bundle-Entwicklung für Knopflerfish unterstützt.

Steigt die Anzahl externer Entwickler und die Anzahl an einzusetzenden Plugins und deren (versionsbehafteten) Abhängigkeiten, so nimmt auch die Notwendigkeit zu, immer mehr der im OSGi-Standard bereits spezifizierten - und in dessen vorhandenen Implementationen implementierten - Dienste und Mechanismen auch im Plugin-Mechanismus von Mayday vorzusehen. Insofern erscheint auf lange Sicht eine Portierung von Mayday auf ein ausgereiftes und ausgetestes Komponentenmodell wie OSGi notwendig - möglicherweise über den Zwischenschritt einer vorherigen Konsolidierung der Code-Basis mittels eines eigendefinierten, vereinfachten Plugin-Modells.

So flexibel der Ansatz über das MIO-Framework auch ist, die sehr intensive Nutzung des MIO-Frameworks zum Informationsaustausch innerhalb und zwischen den Plugins zeigen bereits die derzeitigen, inhärenten Leistungsgrenzen auf. Die aktuelle Implementierung bedarf einer kritischen Durchsicht (Code Review) im Hinblick auf weitergehendere Optimierungen. Zu überlegen wäre eine engere Integration einfacher, vor allem numerischer Datentypen in eine flexiblere `MasterTable`, um bei längeren Rechnungen einen schnelleren Zugriff zu haben und den MIO-Mechanismus komplexeren, strukturreicheren Datentypen vorzubehalten.

Weitere Ziele in der Kernentwicklung wäre die oben schon am Beispiel des Packages `mayday.misc.MathObjects` erwähnte weitere Auslagerung von Teilen des Kerns in eigene, austauschbare Plugins. Neben der erwähnten mathematischen Bibliothek als Plugin ist es von Vorteil das Session-Management in einem eigenen Plugin zu entwickeln. Damit wäre es möglich jeweils ein geeignetes Sessionmanagement einzusetzen (pluggable Sessionmanagement), das sich durch die Nutzung unterschiedlicher (Datenbanken-)Backends, die Form von Bearbeitungsprotokollen (Sessionhis-

tory) oder Redo-Undo-Strategien bzw. Transaktionsmodelle unterscheiden kann.

Sogar die Auslagerung der Task- und Thread-bezogenen Klassen `mayday.core.task` in eine Plugin-Library erscheint eine lohnendes Ziel. Hier könnten verschiedene Entwickler Verarbeitungs- und Schedulingstrategien abgestimmt auf bestimmte Anwendungsfälle oder Architekturen ihrer Plugins hinterlegen und sie damit zugleich auch für andere Pluginentwickler in konsistenter Weise zugänglich machen.

Insgesamt läuft die langfristige Strategie in der Entwicklung des Mayday-Kerns auf eine Beschränkung auf die grundsätzliche Funktionalität eines infrastrukturellen Kerns hinaus, was die Komplexität verringert, die Test- und Wartbarkeit erhöht und damit die potentielle Fehleranfälligkeit senkt.

4.5.2 Die Plugins

In Bezug auf die Palette an Plugins ist generell ein weiterer Ausbau an Funktionalität wichtig, d.h. sowohl vorhandene Plugins müssen mehr Methoden und Verfahren implementieren, wie auch die komplette Neuentwicklung von Plugins für aktuell veröffentlichte Verfahren und Technologien ist notwendig. Vor allem für letzteres ist die Gewinnung externer Entwickler von großem Vorteil. Einerseits steigen so die verfügbaren Ressourcen für die Weiterentwicklung von Mayday und andererseits können so andere inhaltliche Sichten bzw. Standpunkte von Gruppen integriert werden, die eine etwas andere Ausrichtung ihres Arbeitsumfeldes haben. Das würde die inhaltliche Vielfalt an Plugins weiter stärken.

Für Plugins, die sich vor allem unter interner Kontrolle befinden, ist eine Durchsicht in Bezug auf ihre inhaltliche, logische Konsistenz notwendig. Zum Teil spiegelt der strukturelle Aufbau historisch gewachsene Strukturen wieder und orientiert sich nicht an inhaltlichen Aspekten. Beispielhaft sei dafür das Statistik Plugin genannt. Wie schon am Anfang des Abschnitts 4.5.1 erwähnt, verteilen sich statistische Verfahren und Methoden über viele Plugins hinweg. Teilweise ist dies nicht zu verhindern, wenn es sich um ganz spezifische, sehr eng mit einem Plugin zusammenhängende Verfahren handelt. Eine nähere Betrachtung zeigt jedoch, dass bereits bei sehr allgemeinen Verfahren, wie Mittelwertbildung, Standardabweichung, Korrelationskoeffizienten, etc., Überschneidung und damit unnötige Code-Replikationen in Mayday auftreten. Anzustreben wäre eine Aufwertung des zur Zeit sehr rudimentären Statistik Plugins durch mehr Verfahren. Falls die Realsisierung der eingangs des letzten Abschnitts erwähnten mathematischen Plugin-Bibliothek möglich ist, wäre dabei ein Szenario vorzuziehen, bei dem im Statistik Plugin selbst nur die Schnittstelle zum Nutzer residiert. Die Algorithmen und Strukturen, sozusagen das eigentliche statistische Backend, sollten innerhalb der allgemein zugänglichen mathematisch-statistischen Plugin-Bibliothek implementiert sein. Die Package-Strukturen von `mayday.statistics` und `mayday.core.misc.MathObjects` legen für die weitere Entwicklung dieser mathematischen, statistischen Bibliothek bereits eine gewisse Kategorisierung und Ordnung zu Grunde.

Nach wie vor besteht für Mayday ein großer Bedarf an der Implementierung von statistischen Verfahren innerhalb portierbaren Java's. Aspekte und Probleme, die

dies aufwirft, werden im nächsten Kapitel aufgegriffen (Kap. 5).

4.5.3 Die Organisation der Entwicklung

In Abschnitt 4.4.1 wurde bereits erwähnt, dass agile Methoden der Softwareentwicklung den Anforderungen der akademischen Softwareentwicklung recht nahe kommen und deshalb einen guten Ausgangspunkt für den Entwurf des eigenen Entwicklungsprozesses bilden. In Ansätzen sind bereits einige Praktiken des XP, einem der bekanntesten agilen Entwicklungsprozesse, bei der Mayday-Entwicklung verwirklicht, wie die räumliche Nähe der entwickelnden Personen, die evolutionäre Entwicklung, die kontinuierliche Integration und ansatzweise die testgetriebene Entwicklung. Gerade letztere sollte in Zukunft, dort wo sie möglich ist, konsequenter angewendet werden. Die schon existierenden JUnit-Tests [JUn08] in den mathematischen Code-Bestandteilen müssen weiter ausgebaut werden, da sie zugleich als regelmäßig prüfbarer Indikator der Korrektheit der Implementation grundlegender Verfahren von Mayday dienen. Zusätzlich ist eine Ausdehnung auf andere Bereiche und damit eine Erhöhung des abgetesteten Codes (Coverage) notwendig. Ist dieser Anteil ausreichend hoch, kann die Ausführung der Tests in die Build-Maschinerie von Mayday integriert und damit ebenfalls automatisiert werden.

JUnit-Test, Nutzer- und Anwenderdokumentation sollten bei der Durchführung von Entwicklungen im Rahmen von Examensarbeiten mehr in den Vordergrund gerückt werden. Dies erscheint ein auch anwendbarer Weg zur besseren Umsetzung des in Abschnitt 4.4.1 erwähnten kollektiven Codebesitzes. Regelmäßige Code Reviews sind zwar anzustreben, aber leider nur selten durchführbar, während Pair Programming unter den gegebenen Randbedingungen nicht anwendbar ist.

In Bezug auf die Unterstützung des Entwicklungsprozesses sollten neben dem Wechsel von CVS auf Subversion, die Integration weiterer Aspekte, d.h. auch die Build-Maschinerie von Mayday, in ein kompletteres SCM-System erfolgen. Im Hinblick auf die angestrebte Gewinnung und Integration externer Entwickler muß dieses Web-basiert sein. Empfehlenswert ist ein bewährtes System wie Trac [Tra08], das eine sehr große Zahl an Aspekten der Softwareentwicklung unter seiner Oberfläche vereint, wie Versionierung, Bugtracking, History- und Wiki-Dokumentation, und für das ein CruiseControl-Plugin existiert.

Weiterhin muß ein Release-Management für Mayday implementiert werden. Der bisherige Mechanismus des schnellen Uploads aus dem Build-System auf [DGS+08] kann weiterhin zum Verfügbarmachen täglicher Snapshots angewendet werden. Darauf aufbauend sollte jedoch eine manuelle Definition von Release-Versionen erfolgen. Möglicherweise müssen hierfür noch Regeln zur vorherigen Testung der zu veröffentlichenden Plugins und des Kerns aufgestellt bzw. gefunden werden.

5 Computational Statistics für Mayday

> . . . the null hypothesis is never proved or established, but is possibly disproved, in the course of experimentation. Every experiment may be said to exist only to give the facts a chance of disproving the null hypothesis
>
> *(R. A. Fisher)*

5.1 Bedeutung der Computational Statistics für die Genexpressionsanalyse mittels Microarrays

Wie schon bei der Einführung in das Feld der Microarray-Technologie von Teil I gezeigt, bildet das Schnittfeld der drei Gebiete Biologie, Statistik und Informatik die Grundlagen zur Auswertung von Microarray-Daten. Während die Biologie das eigentliche Anwendungsfeld darstellt, ist die Benutzung von Methoden aus den Bereichen Statistik und Informatik vor allem den Eigenschaften des „Messprozesses“ der Genexpression mit Hilfe von Microarrays geschuldet. Das ist insbesondere die Vielzahl an potentiellen Störquellen, die den gemessenen Expressionswert beeinflussen, und der große Umfang an erhobenen Daten, der für ein Hochdurchsatzverfahren, wie es Microarray-Experimente darstellen, typisch ist.

Beides verlangt den Einsatz geeigneter statistisch-mathematischer Methoden, um aus der großen Menge an „rauschbehafteten“ gewonnenen Daten valide Informationen über Strukturen, Muster und deren Zusammenhänge zu gewinnen, die dann letztendlich dem Generieren von Wissen eine verlässliche Grundlage liefern.

Statistische Methoden und damit auch neuere Entwicklungen auf dem Gebiet der Statistik, wie z.B. die Anwendung Bayesianischer Methoden [BL01, BH02, Bol04, Lee04, SG04], spielen deshalb eine herausragend wichtige Rolle für den weiteren Fortschritt in der Auswertung von Geneexpressionsexperimenten. Dies bedeutet, dass aktuelle Entwicklungen der Statistik zielgerichtet und anwendergerecht in die methodische Forschung der Genexpressionsanalyse integriert und in Form geeigneter Softwaretools für die Anwender zur Verfügung gestellt werden. Darauf muss bei der Entwicklung von Tools oder Plattformen zur Analyse von Microarray-Daten, wie sie beispielsweise Mayday darstellt, ein großes Augenmerk gelegt werden.

Eine wichtige aktuell voranschreitende Entwicklung in der Statistik wird unter dem Begriff Computational Statistics zusammengefasst [GHM04]. Ein weitreichendes, inhaltsübergreifendes Gebiet, das im Wesentlichen durch einen intensiven Einsatz des Computers gekennzeichnet ist und durch die rasante Entwicklung der Rechentechnik, insbesondere des PCs, erst ermöglicht wurde. Viele der neueren Methoden sind von einem hohen numerischen Aufwand geprägt, wie z.B. der Einsatz Bayes'scher Modelle oder aber auch die heutige Möglichkeit auf asymptotische Näherungen zu verzichten und zu exakten Lösungen übergehen zu können. Das sind Möglichkeiten, die neue Strategien für die Auswertung von Microarrays eröffnen.

Die Beziehung in der sich Genexpressionsanalyse und Statistik gegenüberstehen ist keinesfalls nur einseitig. Die aus diesem Anwendungsfeld heraus erwachsenden Aufgaben führen wiederum zum Anstoß neuer Entwicklungen auf dem Gebiet der Statistik, die traditionell nicht mit einer derartig großen Menge an Daten, in diesem Fall vor allem der Beobachtungszahl von bis zu einigen 100000 Transkripten, umzugehen hat. Als ein Fall sei hier nur das Finden der geeignetsten Korrekturmethode für das Problem des Multiplen Testens genannt, bei dem traditionelle Verfahren für große Microarrays oder gar Whole-Genome-Chips nicht adäquat sind und zu Entwicklungen wie beispielsweise dem RP [BAAH04] geführt haben.

Speziell bei der Anforderung an die Statistik sehr umfangreiche Datenmengen auszuwerten, steht die Microarraytechnologie nicht allein, sondern zusammen mit einer ganzen Reihe weiterer Anwendungen, wie z.B. aus dem finanzökonomischen Umfeld, der Auswertung von Onlinetransaktionen oder von funktionellen Kernspindaten (fMRI), um nur einige zu nennen [Rip04].

Ganz allgemein wird die Wissensgenerierung aus großen Datenmengen als *Data Mining* bezeichnet [HMS01, WF05]. Dafür ist die Anwendung von Methoden des *Machine Learning* von großer Wichtigkeit [WF05], die selbst zu einem großen Teil dem statistischen Lernen entstammen [HTF01] und teilweise auch rechenintensiv sind. Etwas provokativ formuliert es B. Ripley in [Rip04]:

> To paraphrase provocatively, ‚machine learning is statistics minus *any* checking of models and assumptions'.

Zusätzlich spielt für das Data Mining ein schneller Zugriff auf die umfangreichen Datenmengen eine bedeutende Rolle. Weshalb zugleich ein hoher Bedarf an leistungsfähigen Strategien und fortgeschrittenen Methoden der Informatik zur Speicherung, zum Transport und zur Verwaltung der erhobenen Daten, besteht. Auch dieser Aspekt im Schnittpunkt zwischen Informatik und Statistik ist ein wichtiger Punkt bei der Entwicklung von Software für die Genexpressionsanalyse.

Zusammenfassend betrachtet besteht aus den verschiedenen aufgeführten Gründen ein großer Bedarf für den Einsatz moderner Verfahren und Strategien der IT-Technologie für Genexpressionsexperimente. Diese Verfahren müssen dem Anwender aus dem medizinischen und biologischen Umfeld in einer geeigneten einfachen, leicht nutzbaren Weise zugänglich gemacht werden. In den Abschnitten dieses Kapitels soll hiervon lediglich der Aspekt der Computational Statistics herausgegriffen werden.

5.2 Computational Statistics

Computational Statistics stellt einen relativ neuen, sehr breit gefächerten und fragmentierten Zweig in der Entwicklung der Statistik dar. Zwar gab es schon immer einen hohen Bedarf nach statistischem Rechnen, entsprechend den jeweils modernsten Möglichkeiten der Rechentechnik. Aber gerade die sprunghafte Entwicklung der Rechenleistung innerhalb der letzten Jahre, vor allem durch den Einsatz des PCs, hat dieses Gebiet revolutioniert und dabei wesentlichen Einfluss auf die verwendeten Methoden genommen (siehe auch [Efr79b]). In [GHM04] findet sich hierzu das folgende Zitat:

> To do data analysis is to do computing. Statisticians have always been heavy users of whatever computing facilities are available to them. As the computing facilities have become more powerful over the years, those facilities have obviously decreased the amount of effort the statistician must expend to do routine analyses. As the computing facilities have become more powerful, an opposite result has occurred, however; the computational aspect of the statistician's work has increased. This is because of paradigm shifts in statistical analysis that are enabled by the computer.

Die Anfänge dieser Entwicklung kennzeichnen sehr rechenintensive Methoden, die Mitte des vorigen Jahrhunderts innerhalb der Physik zur Lösung von Zustandsgleichungen für Molekül- und Atomgruppen erstmals zur Anwendung kamen.

Als Beispiel seien hier die Veröffentlichung der *Monte-Carlo-Methode* in [MU49] und des ersten ursprünglichen Entwurfes zum Metropolis-Hastings-Algorithmus [MRR+53] genannt. Einen frühen Überblick über den nach kurzer Zeit schon weitgefächerten Einsatz von Monte-Carlo-Methoden findet man in [HH64], insbesondere auch eine Darstellung des intensiven Einsatzes des Metropolis-Algorithmus innerhalb der statistischen Physik in Abschnitt 9.3 von [HH64]. Hervorzuheben ist der bereits in [MU49] genannte Vorteil der hohen Parallelisierbarkeit dieser Methoden, der im Zusammenhang mit aktuellen Hardwareentwicklungen von großem Interesse ist (siehe Abschnitt 5.6).

Der Metropolis-Hastings-Algorithmus steht gemeinsam mit dem sogenannten Gibbs-Sampling im engen Zusammenhang mit den *Markov Chain Monte-Carlo*-Methoden (MCMC), die überhaupt erst die Berechnung von praxisrelevanten bayes' schen Statistikmodellen ermöglichen.

Bis ca. Mitte der 80iger Jahre des vorherigen Jahrhunderts galt die bayes'sche Lesart der Statistik als ein schöner, theoretischer Zugang, aber aufgrund der auftretenden Schwierigkeiten bei der konkreten Berechnung für reale Probleme als nicht anwendbar. Mit Hilfe der äußerst flexiblen und anpassbaren MCMC-Methoden lassen sich die im bayes' schen Ansatz auftretenden *A-Posteriori-Wahrscheinlichkeit* verschiedener Modelle numerisch ausreichend charakterisieren [Tie94, Gen03], so dass sie in nahezu allen Bereichen der Bayes' schen Analyse eingesetzt werden

[GL06], wie z.B. bei Punktschätzverfahren oder der Berechnung von Randwahrscheinlichkeiten von Modellen unter bestimmten Bedingungen.

Ein weiteres sehr wichtiges Teilgebiet mit einem sehr hohen Bedarf an rechentechnischer Leistungsfähigkeit sind die *Bootstrap*- und *Resampling*-Methoden. Die Wurzeln der grundlegenden Ideen dazu reichen zurück bis zur *Jackknife*-Methode, die von Quenouille (1949/1956) und Tukey (1958) [Mil74] beschrieben wurde. Der Begriff und die Methodik des Bootstraps wurden später von Bradley Efron 1979 [Efr79a] eingeführt und hat sich mittlerweile als eine Standardmethode der modernen Statistik etabliert [DH97]. Hierbei werden auf der Grundlage einer vorhandenen Stichprobe $(X_1, \ldots, X_n)$ immer wieder neue artifizielle Stichproben erzeugt $(X_1^*, \ldots, X_n^*)$, auf denen dann die interessierende Statistik $T(X_1^*, \ldots, X_n^*)$ berechnet wird. Dieser Schritt ist für die Bootstrap-Methode kennzeichnend, bedingt aber auch deren hohe Rechenintensität [Efr79b]. Durch die erzeugte Vielfalt an Stichproben und der damit verbundenen empirischen Verteilungsfunktion $F_{emp}(X)$ ist eine genaue analytische Kenntnis der zugrunde liegenden Verteilungsfunktion $F(x)$ nicht notwendig - unparametrischer Bootstrap.

Allerdings ist die Anwendung auf Fälle sinnvoll, in denen ein komplettes oder teilweises theoretisches Modell bekannt ist bzw. vorliegt - parametrischer Bootstrap (siehe auch Remark K in [Efr79a]). Zum Beispiel dann, wenn die theoretische Berechnung bestimmter interessierender Parameter zu umfangreich und komplex ist, oder auch um den Einfluss bestimmter Annahmen, die das verwendete Modell bedingen, zu erfassen bzw. zu überprüfen [DH97]. Mit Hilfe des Bootstraps lässt sich die Effektivität, die Validität und Robustheit der verwendeten statistischen Methoden abschätzen, beispielsweise über die Abschätzung von Standard Fehler und Konfidenzintervall [EG83, ET86, Efr87, DH97].

Aber auch der eher klassische Bereich der numerischen Algorithmen, wie sie beispielsweise in [Lan99b, PTV92, VTPF92] zu finden sind, und der Algorithmen zur Erzeugung von Zufallszahlen ausreichender Qualität [Gen03, GHM04] muss als wichtiges Fundament der vorgestellten Gebiete zur Computational Statistics gerechnet werden.

Insgesamt stellt das Handbuch der Computational Statistics [GHM04] den bisher umfassendsten und aktuellsten Versuch dar, einen Überblick über dieses weitgefächerte Gebiet zu geben.

5.3 Computational Statistics und Mayday

Aus der bereits in Abschnitt 5.1 erklärten Bedeutung, die die Computational Statistics für die Software zur Auswertung von Microarraydaten hat, folgt deren besondere Bedeutung für Mayday. Allerdings ist in dieser Hinsicht die gewählte Implementierungssprache Java als eine einschränkende Randbedingung zu sehen und die hier wählbaren Möglichkeiten für Mayday werden vor allem dadurch vorgegeben.

5.3.1 Numerik in Java

Java ist keine klassische Sprache zur Implementierung von numerischen Problemen oder des *High Performance Computing* (HPC) , wie die dafür traditionell vorherrschenden Implementierungssprachen Fortran oder C/C++. Java erregte bereits nach seiner Einführung durch das erste Release von Sun 1996 [Sun96] auch das Interesse der Numerik- und HPC-Community. Für die speziellen Aspekte des statistischen Rechnens wurde Java als aussichtsreich eingeschätzt [The99]. Im Zusammenhang mit der Veröffentlichung der wesentlich gereifteren Java 2 Plattform [Sun98] begann eine rege Entwicklung und Diskussion der Eignung von Java unter dem Aspekt der Numerik und des HPCs. So wurde im Jahre 1998 unter der Leitung von Sun Microsystems das Java-Grande-Forum, bestehend aus einer Reihe von Partnern aus öffentlichen Einrichtungen, den Hochschulen und der Industrie gegründet, um die Bemühungen zur Etablierung von Java als geeignete Sprache für die Numerik und für das HPC zu bündeln und zu koordinieren [Jav98]. Innerhalb des Java-Grande-Forums gibt es eine Reihe von Arbeitsgruppen, deren wichtigsten die folgenden beiden sind:

- Die *Java Numerics Group*, die sich vor allem den numerischen Aspekten von Java zuwendet und hauptsächlich vom *National Institute of Standards and Technology* (NIST) koordiniert wird [BP07].

- Die *Concurrency and Applications (Benchmark) Group*, die sich speziell den Fähigkeiten von Java für das parallele und verteilte Rechnen, sowie der Entwicklung geeigneter und aussagekräftiger Benchmarks widmet [EPC07]. Im Rahmen der Aufgabenstellungen dieser Gruppe haben in den letzten Jahren Fragestellungen rund um die Benutzung von Java für das Grid Computing [FK99, FKT01] an Bedeutung zugenommen.

Wesentlicher Grund für das Interesse an Java war vorrangig die Aussicht auf eine breite Portierbarkeit durch das verwendete Konzept einer virtuellen Maschine – der JVM. Aufgrund des für die Portierbarkeit verwendeten Konzepts der JVM ist es allerdings innerhalb von Java schwer möglich, spezielle Eigenschaften der unterliegenden Hardware für die Optimierungen von numerischen Algorithmen zu nutzen.

Außerdem läuft nativer Code, auf eine Plattform hin optimiert, prinzipbedingt etwas schneller als der einer virtuellen Maschine, wobei durch den Einsatz eines *Just-In-Time-Compiler* (JIT-Compiler) die Leistungsfähigkeit der JVM bedeutend gesteigert werden konnte [HAM00]. In Tabelle 5.1 findet sich eine Übersicht über die Vor- und Nachteile von Java im Hinblick auf die Numerik und HPC. Diese wurden größtenteils schon vor Jahren diskutiert [BDP+98, HAM00, BMPP01, SB01, Rit01a, Rit01b, FG03], sind aber noch immer aktuell.

Trotz des in Tabelle 5.1 scheinbaren Überwiegens der Vorteile, die Java für diesen Bereich zu bieten hätte, ist seit ca. 2003 ein abrupt zu nennender Rückgang des Interesses an Java zu erkennen. So fanden beispielsweise ab 2003 keine Meetings

Vorteile	Nachteile
Portabilität (JVM)	geringere Optimierungsmöglichkeiten für die eingesetzte Hardware, geringere Performanz
gute Integration von Netzwerk- und Internetprogrammierung	keine vollständige Unterstützung des IEEE 754 Standards für Floating-Point-Operationen
robustes und einfaches *Object Oriented Programming* (OOP)-Modell	fehlender Lowlevel- bzw. leichtgewichtiger Typ für Komplexe Zahlen
gute Unterstützung durch Software-Tools (Software-Engineering)	fehlende Unterstützung effizienter multidimensionaler Arrays
enthaltenes Sicherheitsmodell	fehlende breite Unterstützung durch numerische Bibliotheken
einfache Entwicklung von GUIs	nicht unerheblicher Aufwand, um Fortran oder C/C++-Routinen nach Java zu portieren
weite Verbreitung von Java und Java-Kenntnissen (niedrigere Entwicklungskosten)	keine Überladung einer Untermenge von Infix-Operatoren möglich

Tabelle 5.1: Übersicht über die Vor- und Nachteile von Java als Implementierungssprache von numerischen Algorithmen und Programmen für das HPC. Siehe Text für eine ausführlichere Erläuterung bestimmter Punkte. Die Übersicht stellt einen Konsens über verschiedene Quellen hinweg dar [BDP+98, HAM00, BMPP01, SB01, Rit01a, Rit01b, FG03].

des Java Grande Forums mehr statt (vgl. [Jav07b]) und die letzten auf [Jav07b] gelisteten Ereignisse sind der *Workshop on Java in Computational Science* auf der *International Conference on Computational Science* 2003 (ICCS 2003) sowie der 5. *International Workshop on Java for Parallel and Distributed Computing* auf dem *International Parallel and Distributed Processing Symposium* 2003 (IPDPS 2003). Zugleich bestätigt sich diese Beobachtung, wenn die Entwicklungsaktivität verschiedener freier Java-Bibliotheken für die Numerik betrachtet wird. Auch hier nahm bei vielen die Entwicklungsaktivität seit dieser Zeit ab. Näheres zu numerischen Java-Bibliotheken soll im nachfolgenden Abschnitt 5.3.3 erläutert werden.

Eine mögliche Erklärung des genannten Sachverhaltes soll in einer genauen Analyse der wichtigsten in Tabelle 5.1 aufgeführten Nachteile und einer Abwägung ihrer Relevanz erfolgen. Insbesondere unter dem Aspekt, dass sie noch immer bestehen und sich auch bei den wichtigen numerischen Punkten in der Zusammenarbeit mit Sun offensichtlich keine Verbesserung erreichen ließ (vgl. hierzu auch die Ausführungen und Zitate von Dr. James Gosling, dem „Vater von Java“, in [KD98] Seite 3,4, 5 und 7).

☞ **Performanz**
Leider gibt es keine aktuellen Studien zum direkten Vergleich der Performanz von Java mit den derzeit am häufigsten eingesetzen Sprachen C/C++ und Fortran. Allerdings belegen schon die älteren vorhandenen Studien wie z.B. [HAM00] und öffentlich zugängliche Benchmarks für aktuelle Plattformen wie z.B. [PM00], dass die reinen Laufzeitverluste nicht zu groß sind, um Java für numerische Berechnungen im großen Stil von vornherein zu disqualifizieren. Als schwerwiegendere Einschränkung zeigt sich, auch aus eigenen Erfahrungen bei der Arbeit mit Mayday, der Speicherbedarf und das Speichermanagement von Java bei großen umfangreichen Datensätzen. Zwei dafür sehr wichtige Gründe werden im Zusammenhang mit numerischen Berechnungen in den beiden übernächsten Punkten benannt und ausführlicher erläutert.

☞ **IEEE 754**
Der Standard IEEE 754 [IEE85] für Floating-Point-Operationen definiert die Darstellung und die genaue Durchführung von mathematischen Operationen auf binären Gleitkommazahlen in der *Floating Point Unit* (FPU) eines Rechners bzw. Prozessors. Im Standard sind unter anderem definiert:

- 4 Typen:
  - Single:
    4-byte `float` mit 24 sign. Bits und 8-bit Exponent
  - Double:
    8-byte `double` mit 53 sign. Bits und 11-bit Exponent
  - Single-Extended (implementierungsabhängig):
    5.5+-byte mit 32+ sign. Bits und 11+-bit Exponent
  - Double-Extended (implementierungsabhängig):
    10+-byte `long double` mit 64+ sign. Bits und 15+-bit Exponent
- 4 verschiedene Rundungsverfahren
- 5 verschiedene Exceptions, Traps und Flags, die bei Operationen aufgetretene Ausnahmen signalisieren
- Darstellung von speziellen Zahlen und Nichtzahlen, wie z.B. 0.0, `NaN` oder `Inf`

Die heute weit verbreitete Hardware, d.h. die allgemein in Rechnern eingesetzen *Central Processing Unit*s (CPU), enthält FPUs, die zum Standard IEEE 754 konform sind. Deshalb lassen sich alle standardkonformen Operationen sehr schnell in der vorhandenen Hardware ausführen. Leider unterstützt Java selbst von Haus aus nicht den kompletten IEEE 754 Standard. [KD98], geschrieben von einem der federführenden Akteure der Standardisierung, liefert einen sehr guten Überblick über die Problematik von Java in Bezug auf das Rechnen mit Gleitkommazahlen. So fehlt in Java die Unterstützung aller 4 Rundungsverfahren, die in ihrere Gesamtheit zur Identifikation numerischer Probleme von Algorithmen dienen können. Weiterhin fehlt Java die umfassende Unterstützung der IEEE 754 Flags und Traps (siehe Seite 22 von [KD98] für

das Beispiel einer desaströsen Folge dieser fehlenden Unterstützung). Java verwendet im Gegensatz zum Kernighan-Ritchie C eine bottom-up Gleitkomma-Semantik, die das Ausführen von Berechnungen in höherer Genauigkeit und das erst danach anschließende Runden verhindert (siehe Seite 44 von [Kah00], sowie Seiten 45 bis 54 von [KD98] für weitere Erkärungen und ein Beispiel der problematischen Folgen).
Als Ergebnis dieser Analyse, beispielsweise die erwähnte von Kahan, wurde durch Darcy u. a. eigens der Sprachdialekt Borneo [Dar98] ins Leben gerufen. Wodurch Java um die komplette Unterstützung des IEEE 754 Standards ergänzt werden sollte. Leider fand Borneo keinen Eingang in die Standarddistribution von Java. Ein im Jahr 2000 in dieser Richtung eingebrachter *Java Specification Requests* (JSR) [Mor00] wurde 2002 aufgrund fehlender allgmeiner Resonanz der Java-Community wieder zurückgezogen.

☞ **Komplexer Datentyp**
Für die Numerik im technisch wissenschaftlichen Bereich sind Berechnungen mit komplexen Zahlen von großer Wichtigkeit. In Java könnte man zwar komplexe Zahlen als eigene, nutzerdefinierte Klasse simulieren (vgl. auch [SB01, Rit01b]), braucht aber für jede einzelne Zahl ein ganzes Objekt, d.h. der dadurch bedingte Speicher-Overhead hätte im Vergleich zu einem vorhandenen primitiven Datentyp einen sehr hohen zusätzlichen Speicherbedarf, der gerade bei der Menge an Daten für das HPC sehr problematisch ist [SB01]. Weiterhin wird durch das Fehlen der Möglichkeit zum Überladen von Operatoren in Java die Notation der Operationen von Objekten dieser Klasse verkompliziert, unübersichtlich und unintuitiv.

☞ **Effiziente multidimensionale Arrays**
Sehr viele numerische Algorithmen operieren auf matrix-artigen Datenstrukturen, deren Effizienz dadurch eine besondere Bedeutung erhält. Dieser Gesichtspunkt muss unter zwei verschiedene Aspekte betrachtet werden - einem auf niedrigem, implementationsspezifischen Niveau und einem auf einer höheren Abstraktionstufe:

- **Statische, multidimensionale Arrays primitiver Datentypen**:
  Bei der Verarbeitung der großen Datenmengen des HPC kommt es oft auf ein Maximum an erreichbarer Geschwindigkeit der darauf operierenden Algorithmen an. Hierfür ist ein effizienter Zugriff auf die einzelnen Datenelemente eines multidimensionalen Arrays von großer Bedeutung. Multidimensionale Arrays sind in Java als Arrays von Arrays realisiert (vgl. hierzu den Code in Listing 5.1), d.h. es besteht in Java nicht die Möglichkeit ein zusammenhängendes, lokales Speicherlayout für ein multidimensionales Array sicherzustellen, um eine maximale Lokalität und somit eine optimale Ausnutzung vorhandener Caching-Strategien darunter liegender Soft- und Hardwareschichten für die optimierte Implementation numerischer Algorithmen auszunutzen (vgl. [SB01]).

Außerdem bedingt diese Implementierungsform, dass numerische Routinen zeitaufwendige Bereichsprüfungen für jedes einzelne enthaltene Array vornehmen müssten, weil diese von unterschiedlicher Länge sein können (siehe auch [Mor00]). Zur Demonstration enthält Listing 5.1 ein Code-Fragment, das ein zweidimensionales Array in „Dreiecksform“ erzeugt. Die erste Zeile des Code-Ausschnittes belegt lediglich ein Feld von Zeigern auf eindimensionale `double`-Felder, die in der Schleife von Zeile 2 bis 7 mit Arrays aufsteigender Länge belegt werden.

Listing 5.1: Java-Code zum Anlegen eines statischen, zweidimensionalen `double`-Arrays in Dreiecksform

```
1 double[][] multiArray=new double[10][];
2 for (int i=0;i<10;i++) {
3         multiArray[i] = new double[i+1];
4         for (int j=0;j<(i+1);j++) {
5                 multiArray[i][j]=j+1.0;
6         }
7 }
```

Ein im Jahr 2000 für die Einführung wirklich multidimensionaler Arrays eingebrachter JSR wurde 2005 aufgrund des schleppenden Fortschrittes und der nicht absehbaren Fertigstellung zurückgezogen.
Als eine nicht vernachlässigbare Einschränkung für das HPC mit Java kann sich auch die Beschränkung bei der Definition von eingebauten Vektoren bzw. Feldern auf den Typ `int` erweisen, womit maximal $2^{31} - 1 = 2147483647$ Werte für einen Vektor bzw. die Seitenlänge eines Feldes in Java zulässig sind.

– **Dynamische Kollektionen:**
Sind die Zeitanforderungen an eine Implementierung eines Algorithmus nicht ganz so strikt wie im eben erwähnten Punkt, wie z.B. in Mayday, greift man auf die bequemer handhabbaren und flexibleren Datenstrukturen zurück, die über die jeweils zugehörige Klassen- bzw. Template-Library der Sprache angeboten werden. Für Java ist es das sogenannte *Java-Collection-Framework*, das eine Reihe von Listen-, Mengen-, Baum- und Vektor-orientierten Datenkontainern anbietet. Auch hier zeigt ein Vergleich von Java mit C/C++ einen Nachteil in Bezug auf die Performanz und den Memory-Footprint der Datenstruktur für einfache numerische Datentypen. Im Collection-Framework, das Teil des Java-SDK von Sun ist [Sun07a], können nur Objekte gespeichert werden. Dem gegenüber können in den Datenkontainern der *Standardtemplate-Library* (STL) von C++ auch einfache Datentypen flexibel verwaltet werden. In einer grundsätzlichen Überarbeitung des Collection-Frameworks im Release 5 von Java [Sun04c, Now05], wurde es, mit Hilfe der eingeführten Generics, auf eine bessere Typisierung umgestellt und damit eine einfachere Handhabung und mehr Fehlerkontrolle durch den Compiler er-

reicht (Einsparung unhandlicher und potentiell gefährlicher expliziter Type-Casts). An der ausschließlichen Verwaltung von Object-abgeleiteten Klassen veränderte sich jedoch nichts. Im schlimmsten Falle kann ein Mehrbedarf bis zum 16-fachen des theoretisch notwendigen Speicherplatzes entstehen. Die Grundlage dieser Zahl, sowie eine ausführlichere Behandlung dieses wichtigen Aspekts von Java für rechen- und datenintensive Anwendungen erfolgt im nächsten Abschnitt 5.3.2 auf Seite 81.

☞ **Numerische Bibliotheken**
Ein großer Teil gut bewährter, ausgetesteter numerischer Bibliotheken liegt in C/C++ oder Fortran-Code vor, wie z.B. BLAS [BLA07] oder auch LAPACK [Lap07], dessen optimierte Portierung nach Java einen beträchtlichen Aufwand verursachen würde. Aus diesem Grund gab es Ansätze mittels automatisierter Übersetzer, wie f2j [SD07a] ein unbedingt notwendiges Subset von Fortran-Code in Java übersetzen zu lassen. Ein Beispiel dafür ist JLAPACK, das ebenfalls auf [SD07a] zu finden ist. Die andere Möglichkeit besteht darin, über das native Interface von Java (*Java Native Interface* (JNI)) auf kompilierten Code zuzugreifen. Diese Möglichkeit bringt jedoch neben dem zeitlichen Verlust durch eine weitere Zwischenschicht und dem Mehraufwand bei der Programmierung der Kommunikationsschicht den Nachteil mit sich, dass die geschriebene Anwendung selbst einen nicht hardwareneutralen Teil beinhaltet, bzw. von diesem abhängig wird. Wodurch der größte Vorteil der Verwendung von Java nicht mehr voll zum Tragen kommt.
Insgesamt liegen zwar eine Reihe von numerischen Java-Bibliotheken vor, es lässt sich jedoch im Bereich von Open-Source keine ausmachen, die umfassend genug ist, um beispielsweise eine ähnlich allgemeine Verbreitung wie die Gnu Scientific Library (GSL) [Fre07b] für C/C++ zu erreichen. Außerdem ist bei vielen Bibliotheken in den letzten Jahren das Fehlen einer aktiven Entwicklung zu verzeichnen.
Eine genauere Aufschlüsselung und Analyse erfolgt in Abschnitt 5.3.3 auf Seite 92.

Interessant wird im Zusammenhang mit der Frage nach der geeignetsten Sprache für spezielle wissenschaftliche und ingenieurtechnische Anwendungen auch die Entwicklung neuer Sprachen wie Fortress und D sein.

Fortress ist ein von Sun speziell für den Bereich des HPCs und die akademische Softwareentwicklung konzipierter Nachfolger von Fortran, der momentan in einem sehr frühen Stadium als β-Release vorliegt [Sun07b]. Fortress selbst ist Open-Source und ein Kern der Sprachspezifikation läuft zu Demonstrationszwecken bereits auf einem JVM-basierten Interpreter.

D [Dig07, Sta07] hingegen positioniert sich als ein moderner Nachfolger von C++, in dem aber auch fortgeschrittene Konzepte anderer Sprachen wie Java, C# oder Ruby aufgegriffen wurden. D behält sich explizit mehr Flexibilität bei der Wahl der

verwendeten Sprachmittel vor. So ist beispielsweise neben der Benutzung eines automatischen auch die Verwendung eines expliziten Speichermanagements möglich. D selbst sieht sich in der Nachfolge von C++ als dedizierte Sprache für große, performante, speicherschonende Anwendungen, wie sie z.B. in der Numerik und bei der Systementwicklung üblich sind. So kennt D beispielsweise ein Operator-Overloading und besitzt einen eingebauten Datentyp für komplexe und imaginäre Zahlen, deren Fehlen sich in Java als nachteilig für seinen Einsatz in der Numerik bemerkbar machte (siehe Tabelle 5.1 auf Seite 76). Außerdem bietet D eine bessere Unterstützung des IEEE 754 Standards und erlaubt auf x86-CPUs die Verwendung eines 80 Bit langen Gleitkommazahl-Typen `real`.

5.3.2 Speichersignatur numerischer Datenfelder in Java

Gerade der Speicherverschnitt von Datenstrukturen ist ein schwerwiegender leistungsbegrenzender Aspekt von rechen- und datenintensiven Anwendungen, wie sie beispielsweise auch Mayday darstellt. Die zunehmend größeren Datenmengen, die von den im Experiment eingesetzten Hochdurchsatzverfahren generiert werden, verschärfen die hierbei vorliegenden Anforderungen an die Anwendung. So ist beispielsweise die in der Array-Technologie erhobene Datenmenge über die letzten Jahre rasant gestiegen. Sind es bei einem großen Expressionsarray, wie dem 2004 herausgebrachten HGU 133 Plus 2.0 Chip von Affymetrix, ca. 47000 Transkripte ($\sim$ 38500 davon sind Gene), die mittels ca. 61000 Probesets detektiert werden [Aff04], so verteilen sich auf aktuellen SNP Arrays, wie dem Affymetrix Genome-Wide Human SNP Array 6.0, mehr als 1.8 Millionen genetische Marker (906600 SNPs und 946000 nicht-polymorphe Regionen zur Copy-Number Detektion) [Aff07]. Damit liefert die Durchführung eines einzigen derartigen Experiments an einem einzelnen Individuum einen sehr großen Vektor, der die Messwerte aller Features enthält. Ziel eines biologischen Experiments oder einer medizinischen Studie ist die Verallgemeinerung auf die zugrunde liegende Population oder zumindest eine Teilpopulation, weshalb das Einzelexperiment, unter statistischem Gesichtspunkt, an ausreichend vielen Individuen durchgeführt wird. Fasst man ein solches gesamtes Experiment oder eine Studie als Matrix zusammen, bei der sich in den Zeilen die Features des Einzelexperimentes befinden und in den Spalten die Replikate, können sich dabei umfangreiche Größen ergeben.

Leider fehlen verlässliche und aktuelle Angaben bezüglich des Speicherverschnittes numerischer Datentypen für die unterschiedlichen in Java vorhandenen Datenstrukturen. Aus diesem Grund wurden die Daten für das aktuelle Release Java2SE 6.0 [Sun07a] für zwei verschiedene, unterliegende System- und Hardware-Architekturen selbst erhoben:

☞ **IA32** - *32-bit Intel Architektur*:
- Prozessor (CPU): Intel Pentium M mit 1700 MHz
- verfügbarer Gesamtspeicher (RAM): 503.62 MB
- System: openSUSE 10.2 (i586)

- Linux-Kernel: 2.6.18.2-34-default i686
- verwendete JVM: Java HotSpot Server VM version 1.6.0_01-b06

☞ **AMD64** - *64-bit AMD Architektur*:

- Prozessor (CPU): AMD Athlon 64 X2 Dual Core Processor 3800+
- verfügbarer Gesamtspeicher (RAM): 3.86 GB
- System: openSUSE 10.2 (X86-64)
- Linux-Kernel: 2.6.18.8-0.3-default x86_64
- verwendete JVM: Java HotSpot 64-Bit Server VM version 1.6.0_01-b06

Da auf der **AMD64**-Architektur standardmäßig die Server-JVM lief und der Schalter `-client` dort keine Beachtung fand, wurde für eine bessere Vergleichbarkeit auch auf **IA32** die Server-JVM benutzt.

x	y	Elemente	Speicherbedarf in MB					
			double (8 B)	float (4 B)	long (8 B)	int (4 B)	short (2 B)	byte (1 B)
32768	64	2097152	16	8	16	8	4	2
32768	128	4194304	32	16	32	16	8	4
65536	128	8388608	64	32	64	32	16	8
65536	256	16777216	128	64	128	64	32	16
131072	256	33554432	256	128	256	128	64	32
262144	256	67108864	512	256	512	256	128	64

Tabelle 5.2: Simulierte Datenmatrizen - x bezeichnet die Anzahl der Features des Einzelexperimentes und y die Anzahl an Wiederholungen an Individuen der Grundgesamtheit - und ihr theoretisch minimaler Speicherbedarf in MBytes, wenn man das für die numerischen Datentypen übliche Speicherlayout voraussetzt (B = Bytes).

Um die bei dem oben beschriebenen Anwendugsszenario großer Datenmatrizen auftretenden Speicheranforderungen in Java simulieren zu können, wurden die in Tabelle 5.2 angegebenen Matrizen für verschieden numerische Datentypen erstellt, mit Zufallszahlen ausreichender Größe gefüllt und mit folgenden verschiedenen Speicherstrukturen verwaltet:

☞ **f** - *einfaches zweidimensionales Feld*:
Eine Matrix mit einem eingebauten Feld abzuspeichern, stellt die einfachste Form der Datenstruktur für eine Matrix in Java dar:
double[][] matrix=**new double**[rows][cols];
Hierbei wird, wie bereits in Abschnitt 5.3.1 und Listing 5.1 auf Seite 79 beschrieben, ein Array angelegt, das Zeiger auf Arrays des jeweiligen primitiven Datentyps besitzt. Die Matrix selbst ist nicht mehr flexibel, so dass beim Hinzufügen von Spalten oder Zeilen, jeweils ein neues Array angelegt und alle

aktuellen Daten umkopiert werden müssen. Diese Form stellt die Variante der Abspeicherung einer Matrix in Java mit minimalem Speicheraufwand, aber auch mit der geringsten Felxibilität dar. Sie bildet damit eine untere Schranke für die Abschätzung des Speicherbedarfs.

☞ **c.f** - *eindimensionale Arrays innerhalb des Collection-Frameworks verwaltet*: Eine Mischform der eben gezeigten einfachen Speicherweise mit den flexibleren Methoden des Collection-Framework wäre die Variante innerhalb einer Struktur des Collection-Framework Arrays des zu nutzenden primitiven Datentyps zu verwalten:
ArrayList<**double**[]> matrix=**new** ArrayList<**double**[]>(rows);
Diese Lösung verspricht, im Vergleich zur vorhergehenden Lösung, einen sehr moderaten Overhead. Weil Arrays von primitiven Datentypen in Java selbst wieder Objekte darstellen, lassen sie sich direkt und ohne zusätzliche Konvertierung mit den Strukturen des Collection-Frameworks verwalten. Gleichzeitig kann, zumindest in einer Dimension, die Flexibilität des Collection-Frameworks genutzt werden. So lassen sich bei der oben gezeigten Lösung ohne großen Aufwand neue Zeilen hinzufügen, wobei aber neue zusätzliche Matrixspalten ähnlichen Aufwand erzeugen wie in der im Punkt **f** genannten Methode.
Ein weiterer, im Zusammenhang mit Mayday und der dort verarbeiteten Art von Daten, nicht unwesentlicher Sachverhalt ist, dass man eine problemangepasste Wahl von Datenkontainern des Collection-Frameworks vornehmen kann. Geschieht der Zugriff auf einzelne Zeilen der Datenmatrix über eine alphanumerische ID statt über einen einfachen Index, wie bei Genexpressionsmatrizen üblich, so verspricht die Verwendung einer Hashmap Vorteile.

☞ **c** - *beide Dimensionen werden mit Hilfe des Collection-Framework verwaltet*: Eine Lösung maximaler Flexibilität, aber mit maximalem Speicherverbrauch ergibt die Verwendung von Datenkontainern des Collection-Framework für beide Dimensionen der Matrix:
ArrayList<ArrayList<Double>> matrix=**new** ArrayList<ArrayList<Double>>(rows);
Wie schon bei der Deklaration zu sehen ist, werden jetzt statt flacher primitiver numerischen Datentypen, deren Wrapperobjekte verwaltet, d.h. für jeden einzelnen numerische Wert entsteht der zusätzliche Speicherverschnitt, den die Verwendung eines Objektes verursacht. Andererseits lassen sich so Spalten wie auch Zeilen der Matrix beliebig leicht erweitern oder veringern.

Für die Wahl der Dimensionierung der Testmatrizen wurden aktuelle und für die Zukunft plausible Anforderungen an die Verarbeitung bioinformatischer Daten in Mayday zugrunde gelegt. Das heißt, die Zeilengröße spiegelt die Anzahl der untersuchten genetischen Marker innerhalb eines einzelnen Experiments (z.B. die Transkriptsonden eines Microarrays) und die Spaltenanzahl die Zahl an insgesamt durchgeführten Experimenten (biologische Replikate) wider.

Im Unterschied zu Sprachen wie C oder C++ verwendet Java ausschließlich ein automatisches Speichermanagement, das die Allokation und die Freigabe von Speicher für die Anwendung eigenständig durchführt und diese Aktionen vor dem Anwendungsentwickler verbirgt. Das hat zweifellos Vorteile, besteht doch ein großes Problem von Software, die in C bzw. C++ vorliegt, in Speicherfehlern, wie z.B. Speicherlöchern oder Pufferüberläufen, bei denen der Entwickler den belegten Speicher nicht wieder freigegeben hat bzw. bei letzterem nicht ausreichend viel Speicher vorbereitet hat. Auf der anderen Seite erschwert sich damit aber auch die Vorhersagbarkeit des Speicherverhaltens schon einfacher Anwendung, wie sie beispielsweise die hier zur Messung entwickelten Programme darstellen. Es ist für den Anwendungsentwickler weder kontrollier- noch vorhersehbar, wann der Garbage-Collector belegte Ressourcen auf dem Heap wieder freigibt.

In seiner einfachsten Ausführung müsste ein Garbage-Collector alle von der Anwendung belegten Speicherabschnitte durchgehen und auf ihre Erreichbarkeit von der Anwendung aus überprüfen. Ist diese nicht mehr gegeben, so kann der zugehörige Speicher wieder frei- und in den Pool freien Speichers oder aber an das Betriebssystem zurückgegeben werden. Dieser einfache Algorithmus mit einer Komplexität $O(n)$ verbietet sich jedoch für große Anwendungen, die einen Großteil ihrer umfangreichen Datenmengen im Speicher halten müssen. Unakzeptabel ist beispielsweise das plötzliche, mehr oder weniger lang andauernde Erstarren einer Anwendung, weil der Garbage-Collector entsprechend viel CPU-Last erzeugt. Solche Erfahrungen mit den einfacheren Garbage-Collectoren der ersten Java-Releases brachten Java in den anfänglichen Ruf, für performante und größere Anwendungen ungeeignet zu sein.

Optimale Garbage-Collection-Strategien sind nach wie vor ein aktueller Forschungsgegenstand. Unter anderem die Frage des zeitlichen Abstandes in dem jeweils eine Garbage-Collection durchgeführt werden sollte.

Die Grundlage der zur Zeit in Java implementierten Garbage-Collection-Strategie beruht auf einem generationsbasiertem Speicherverwaltungsmodell (siehe [Sun04d], [Sun06a]), das durch empirische Beobachtungen des Verhaltens von Anwendungen gestützt wird. Im Gegensatz zum naiven Ansatz, der alle Objekte des Heaps untersucht, werden die Heapobjekte in verschiedenen Pools unterschiedlichen Alters gehalten, in sogenannten *Generational Collections*. Wesentliche Grundlage ist die Beobachtung der sogenannten *Infant Mortality*, d.h. es entstehen viele kleine Objekte mit kurzer Lebenszeit, wie z.B. Schleifenvariablen, temporäre Variablen in Blöcken oder Iteratoren und nur wenige große mit langer Lebensdauer. Das erlaubt effektivere Collection-Strategien, da die Collection poolweise gestaffelt in entsprechend anpassbaren Zeitabständen durchgeführt werden kann und sie nicht jeweils den gesamten Bestand an aktiven Objekten betrifft [Sun04d]. Findet eine Collection im Pool der jüngeren Generation Objekte, die bereits eine längere Lebensdauer aufweisen, so kann sie diese in die nächste Generation weiterschieben. Im Allgemeinen wird der Anwendungsentwickler nicht mit diesen Feinheiten des Garbage-Collectors konfrontiert, doch für ausreichend große und speicherintensive Anwendungen, insbesondere Server- oder parallelisierte Anwendungen kann ein spezielles Tuning des

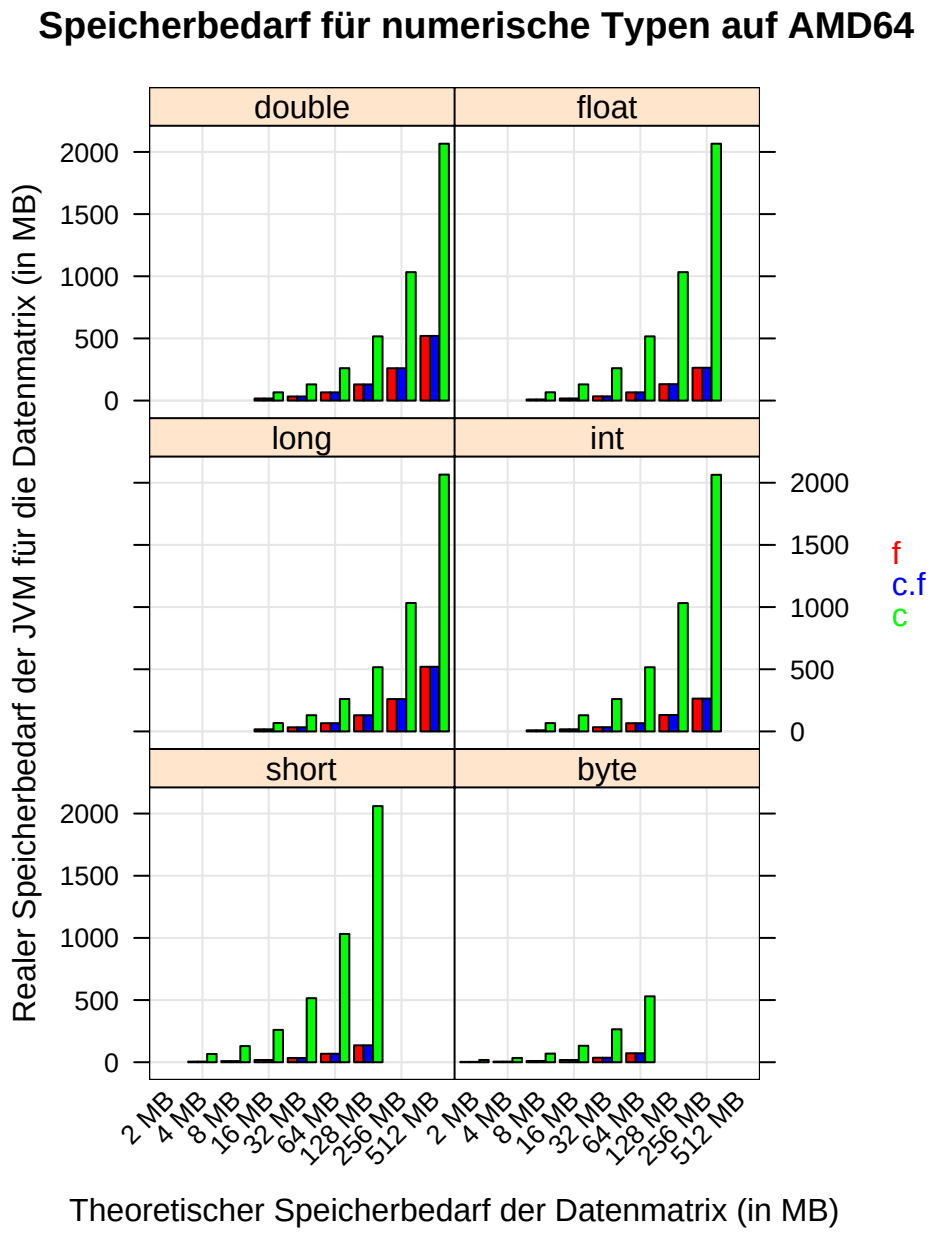


Abbildung 5.1: Darstellung des Speicherbedarfs der 64-bit Sever-JVM auf **AMD64** für Datenmatrizen von Zufallszahlen unterschiedlichen Typs (**double**, **float**, **long**, **int**, **short** und **byte**) und verschiedener Größe (angegeben mit dem theoretisch minimal notwendigen Speicherbedarf für die ganze Matrix). Die Verwaltung der Matrix erfolgt in einem eingebauten zweidimensionalen Datenfeld primitiver Datentypen mit fester Größe **f** (rot), in einem gemischten Ansatz aus Arrays primitiver Datentypen, die in einer Struktur des Collection-Framework verwaltet werden **c.f** (blau) und ausschließlich mit Datenkontainern des Collection-Framework **c** (grün). (Siehe Text für nähere Erläuterungen.)

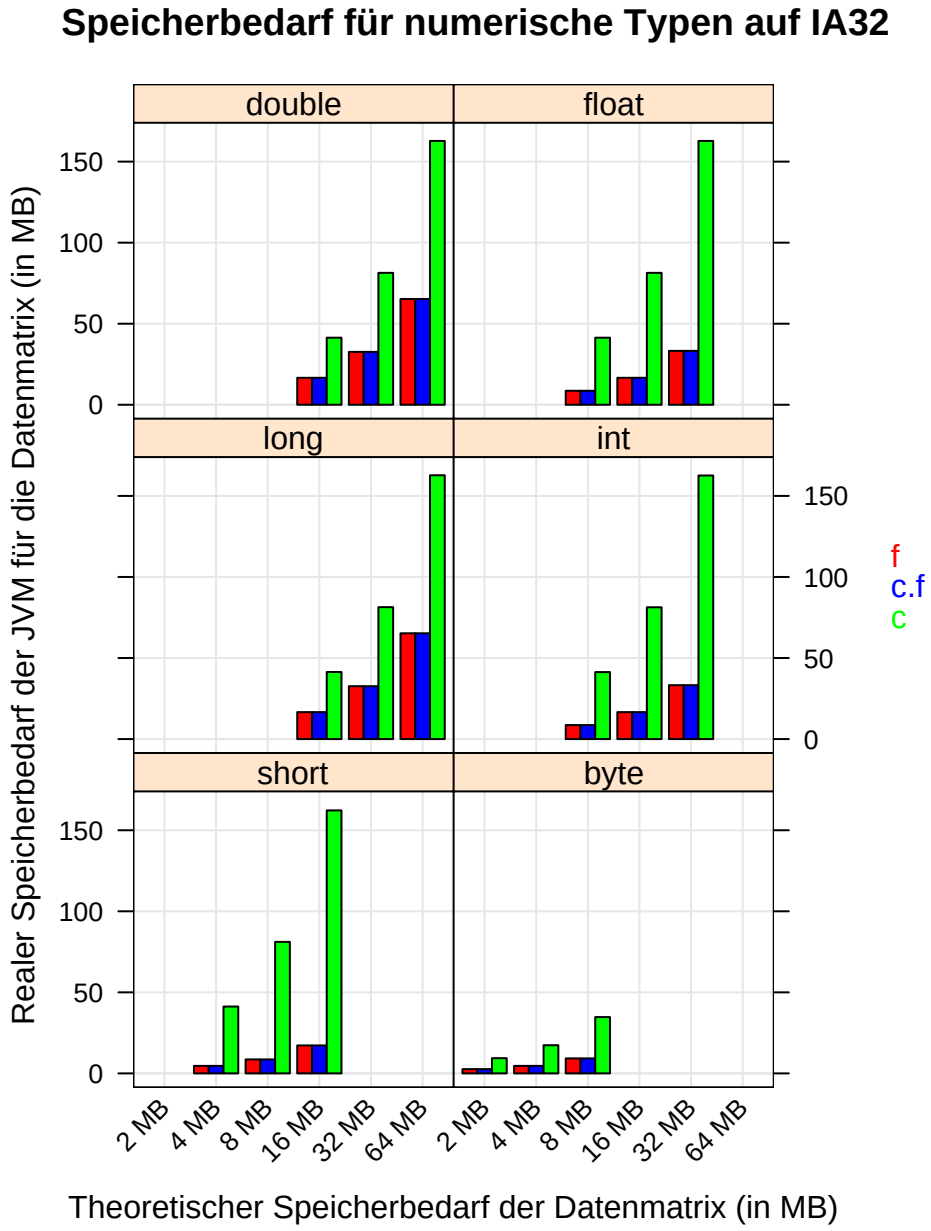


Abbildung 5.2: Darstellung des Speicherbedarfs der 32-bit Sever-JVM auf **IA32** für Datenmatrizen von Zufallszahlen unterschiedlichen Typs (**double**, **float**, **long**, **int**, **short** und **byte**) und verschiedener Größe (angegeben mit dem theoretisch minimal notwendigen Speicherbedarf für die ganze Matrix). Die Verwaltung der Matrix erfolgt in einem eingebauten zweidimensionalen Datenfeld primitiver Datentypen mit fester Größe **f** (rot), in einem gemischten Ansatz aus Arrays primitiver Datentypen, die in einer Struktur des Collection-Framework verwaltet werden **c.f** (blau) und ausschließlich mit Datenkontainern des Collection-Framework **c** (grün). (Siehe Text für nähere Erläuterungen.)

Garbage-Collectors notwendig werden [Sun04d].

Die Dynamik des Speicherverhaltens einer Java-Anwendung schlägt sich auch in den hier vorgenommenen Speichermessungen nieder, die eher einen empirischen, statistischen Charakter haben als einen streng deterministischen. So weisen beispielsweise mehrere Durchläufe des gleichen Messprogrammes geringe Unterschiede in dem gemessenen Speicherbedarf auf. Insgesamt wurde für jeden Typ ein eigenes Messprogramm angefertigt, das sich an der in [Rou02] veröffentlichten Messstrategie orientierte, d.h. jeweils vor und direkt nach dem Anlegen wurde mehrmals der Durchlauf des Garbage-Collectors explizit initiiert, um einer Verfälschung der Messergebnisse durch unbereinigte Speicherbereiche entgegenzuwirken.

Um zusätzlich mehr Sicherheit bei der Abschätzung der gemessenen Speichergrößen zu gewährleisten, wurde stichprobenartig für verschiedene Messungen die Plausibilität der Ergebnisse, durch die Inspektion der Prozesse mit der JConsole [Sun06a, Chu06b, Chu06a], den normalen Systemtools von Unix bzw. Linux und der Kombination des HPROF-Schalters für die JVM bzw. dem Tool jmap und die Auwertung der erzeugten Heap-Dumps mit dem Tool jhat [Sun06b, O'H05, Chu06b, Chu06a, Bat07a, Sun07c], getestet.

Speicher-	numerische Datentypen					
struktur	double	float	long	int	short	byte
f	1.028643	1.058322	1.028638	1.057298	1.114604	1.229182
c.f	1.028647	1.057312	1.028647	1.057305	1.114633	1.229264
c	4.064451	8.140199	4.076403	8.132982	16.214306	8.530245

Tabelle 5.3: Verhältnis des realen Speicherbedarfes der Sever-JVM zum theoretisch minimal nötigen in Abhängigkeiteit des numerischen Datentyps der Elemente und der genutzten Datenstruktur zur Verwaltung der Datenmatrix auf einem **AMD64**-System.

Speicher-	numerische Datentypen					
struktur	double	float	long	int	short	byte
f	1.026043	1.052083	1.026048	1.052091	1.104158	1.208380
c.f	1.026052	1.052091	1.026047	1.052091	1.104183	1.208386
c	2.557279	5.114572	2.557028	5.109494	10.197502	4.458351

Tabelle 5.4: Verhältnis des realen Speicherbedarfes der Sever-JVM zum theoretisch minimal Nötigen in Abhängigkeiteit des numerischen Datentyps der Elemente und der genutzten Datenstruktur zur Verwaltung der Datenmatrix auf einem **IA32**-System.

Die gesamten Messergebnisse für die beiden System-Plattformen **AMD64** und **IA32** sind jeweils in den Trellis-Plots der Abbildungen 5.1 und 5.2 visualisiert. Sehr auffällig sind sowohl die hohe Speicherbelegung bei der alleinigen Lösung über das Collection-Framework von Java, wie auch die Unterschiede im Ressourcenbedarf

zwischen den beiden System-Plattformen. Untersucht man diese beiden Aspekte weiter und berechnet das relative Verhätnis aus dem gemessenen und dem theoretischen Speicherbedarf der Datenmatrizen für jedes System, mittelt man dann über die Ratios aller erhobenen Datenmatrizen hinweg (artihmetisches Mittel), so ergibt sich das in den Tabellen 5.3 und 5.4 gezeigte Bild:

- ☞ Die beiden Lösungsansätze **f** und **c.f** unterscheiden sich nur marginal in ihrem geringeren Mehrbedarf an Speicher im Vergleich zum theoretischen Minimum, woraus sich, bei Bedarf für einen alternativen Zeilen- oder Spaltenzugriff, eine Empfehlung für die Mischlösung aus Strukturen des Collection-Framework als Zugriffsstruktur und eingebauten Arrays als letztliche Speicherstruktur der numerischen Daten ergibt. Zu beachten ist auch, dass die in den Tabellen 5.3 und 5.4 dargestellten Zahlen Mittelwerte sind. Je größer die Datenmatrizen selbst sind, desto geringer wird, relativ gesehen, der durch die zusätzliche Indexstruktur verursachte Overhead und das Verhältnis besser.

- ☞ Generell wird auf der 64-bit Plattform **AMD64** mehr Speicher gebraucht als auf der 32-bit **IA32**. Dies gilt in sehr deutlicher Form für die Speichervariante **c**, aber auch für **f** und **c.f**, wenn auch in geringerem Maße. Dieser geringe Unterschied für **f** und **c.f** bleibt selbst nach einer Bereinigung der Daten um die Verzerrung (Bias) durch die notwendige Beschränkung der Messung auf kleinerer Matrizen auf der **IA32**-Plattform erhalten.
  Betrachtet man den für die Numerik wichtigsten Datentyp **double**, so kann der Unterschied zwischen einem Faktor von $\simeq 4$ auf **AMD64** und $\simeq 2.5$ auf **IA32** bei der Speichervariante **c** nicht vernachlässigt werden. Wird eine umfangreiche Anwendung, aufgrund der Speicherbeschränkung auf 4 GB für Prozesse bei 32-bit Plattformen, auf ein 64-bit System migriert, muss dieser Mehrbedarf an Speicher bei starker Verwendung des Collection-Framework vor allem im Hinblick auf die reale Hauptspeichergröße der Zielhardware beachtet werden.

- ☞ Ein unbedarfter bzw. unüberlegter Einsatz der Kontainer des Collection-Framework für rechen- bzw. datenintensive Anwendungen kann einen enormen Mehrbedarf an Speicher verursachen. Hier sind Spitzenwerte bis zum ca. 16-fachen des theoretisch notwendigen Speichers möglich. Der in dieser extremen Form vermutlich aus einer dynamischen Anpassung des Wertebereichs der numerischen Wrapper-Objekte an den tatsächlichen gespeicherten Inhalt der Variablen entsteht, was insbesonder Datentypen mit eingeschränktem bzw. kleinerem Wertebereich betrifft. Aber auch Datentypen „maximaler“Länge, wie **double** oder **long** erzeugen mit ihren Wrapper-Objekten einen erheblichen Mehrbedarf von jeweils $\simeq 4$ auf **AMD64** und $\simeq 2.5$ auf **IA32** des theoretischen Minimalbedarfs.

Im Vergleich dazu sei noch der Speicherbedarf für numerische Daten unter C/C++ für die gleichen Systeme **IA32** und **AMD64** genannt. Auf beiden Plattformen wur-

de mit dem jeweiligen System-Compiler und Linker aus der Gnu Compiler Collection [Fre07a] `g++ (GCC) 4.1.2 20061115 (prerelease) (SUSE Linux)` kompiliert und gelinkt.

Als Speichervarianten dienten für einen Vergleich die folgenden drei Formen, wobei hier die ersten beiden Alternativen der einfacheren **f** der drei Java-Varianten entprechen, während die dritte (**stl.m**) der Lösung mit Hilfe des Collection-Frameworks (**c**) am ähnlichsten ist:

- ☞ **c.m** - *einfaches lineares Feld mit den Mitteln von C*:
  Die Datenmatrix wird einfach als lineares Speicherfeld alloziert und der spalten- bzw. reihenweise Zugriff auf die Elemente muss über Index- bzw. Pointerberechnungen sichergestellt werden, die in jeweiligen Zugriffsmethoden implementiert sind:
  TYPE∗ matrix = (TYPE∗) malloc((rows∗cols)∗**sizeof**(TYPE));
  Genutzt werden für das Anlegen der Datenstruktur die Sprachmittel des C-Sprachkerns. In Bezug auf die Flexibilität ist diese Lösung sehr eingeschränkt. Einzelne Zeilen oder auch Spalten können nicht hinzugefügt werden, sondern nur durch das komplett neue Anlegen eines Speicherbereiches und das Umkopieren der alten Daten.

- ☞ **low.m** - *einfaches lineares Feld mit den Mitteln von C++*:
  Diese Lösung ist der ersten in all ihren Vor- und Nachteilen äquivalent und unterscheidet sich von dieser nur in der Benutzung der von C++ angebotenen bequemeren Sprachmittel:
  TYPE∗ matrix = **new** TYPE[rows∗cols];

- ☞ **stl.m** - *Darstellung der Matrix mit Hilfe der STL als Vektor von Vektoren*:
  Für diese Lösung werden die Datenkontainer der *Standard Template Library* (STL) verwendet, die das C++-Pendant zum Java Collection-Framework darstellt. Auch sie stellt eine Lösung maximaler Flexibilität dar. Zur Verwaltung der beiden Dimensionen der Matrix finden Vektoren Verwendung - entsprechend dem Vorschlag von Stroustrup (S. 901 Abschn. C.7.1 in [Str00]):
  vector< vector<TYPE> > matrix(rows,vector<TYPE>(cols));

Die Messergebnisse für den nativ kompilierten C/C++-Code der beiden System-Plattformen **AMD64** und **IA32** sind in den Trellis-Plots der Abbildungen 5.3 und 5.4 zu finden. Zu beachten ist, dass der **long**-Typ unter der 64-bit Plattform **AMD64** auf den 8-Byte-Typ **long long** [Mer07] abgebildet wird (der hier der Einfachheit halber und in Übereinstimmung zu den Java-Typ-Definitionen auch als **long** bezeichnet wurde), während er unter der 32-bit Plattform **IA32** auf den 4-Byte-Typ abgebildet wird und damit dort den gleichen Wertebereich wie **int** hat. Die Messwerte sind mit Hilfe der Speicherüberwachung (Memcheck) von Valgrind [Val07] in der aktuellen Version 3.2.3 erhoben worden.

In den Messwerten erkennt man sehr gut den Grundsatz „do not pay for what you don't use“von C/C++ [Mer07]. Die Low-Level-Realisierungen der Matrix erzeugen

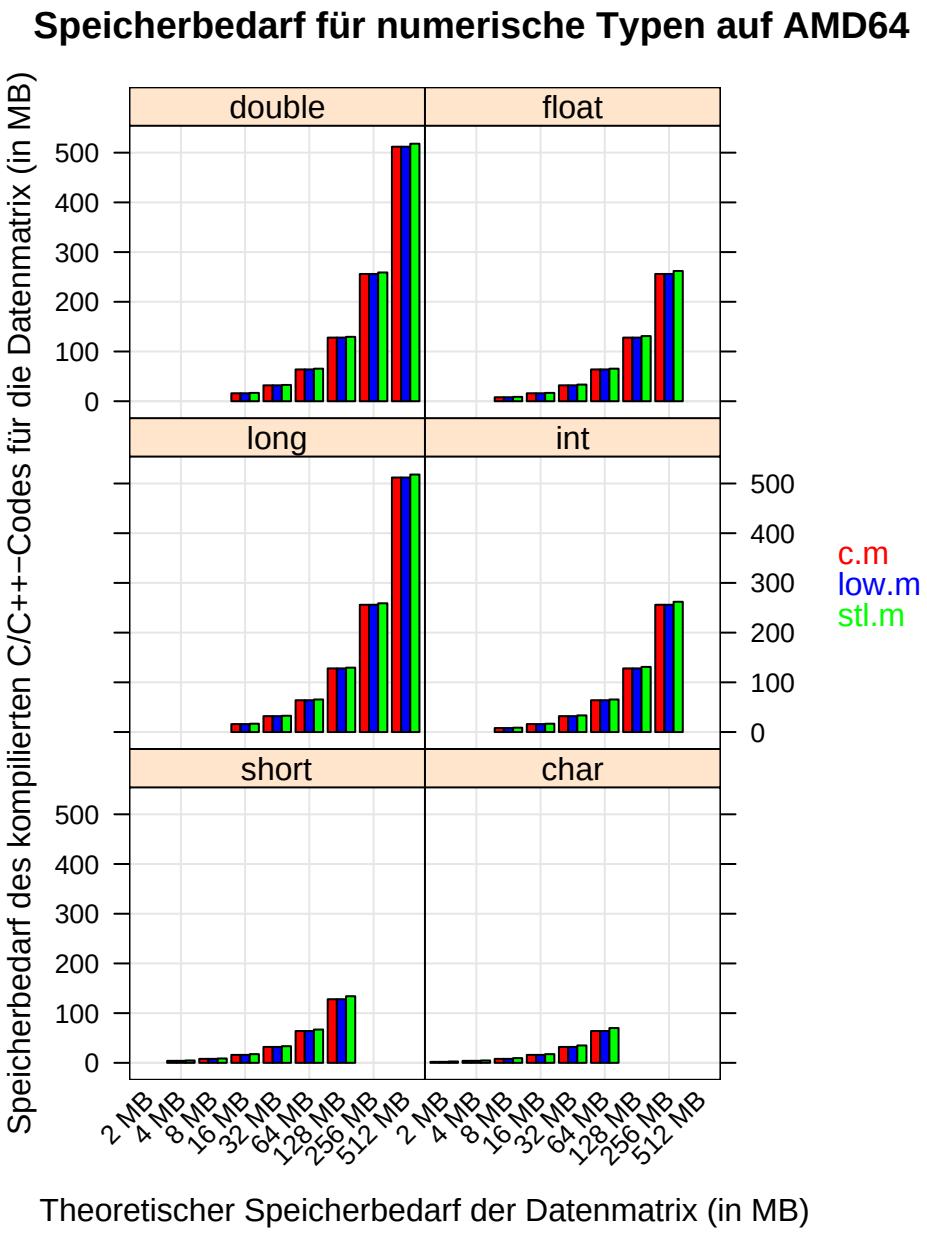


Abbildung 5.3: Darstellung des Speicherbedarfs kompilierten C/C++-Codes auf der 64-bit Plattform **AMD64** für Datenmatrizen von Zufallszahlen unterschiedlichen Typs (**double**, **float**, **long**, **int**, **short** und **char**) und verschiedener Größe (angegeben mit dem theoretisch minimal notwendigen Speicherbedarf für die ganze Matrix). Die Verwaltung der Matrix erfolgt zum einen auf sehr niedriger Stufe als eindimensionaler Speicherbereich mit den von C **c.m** (rot), bzw. von C++ **low.m** (blau) zur Verfügung gestellten Mitteln, und zum anderen mit Hilfe des STL-Datenkontainers `vector` **stl.m** (grün), der die höchste Flexibilität und eine Lösung, vergleichbar der mit Hilfe des Collection-Frameworks in Java, bietet. (Siehe Text für nähere Erläuterungen.)

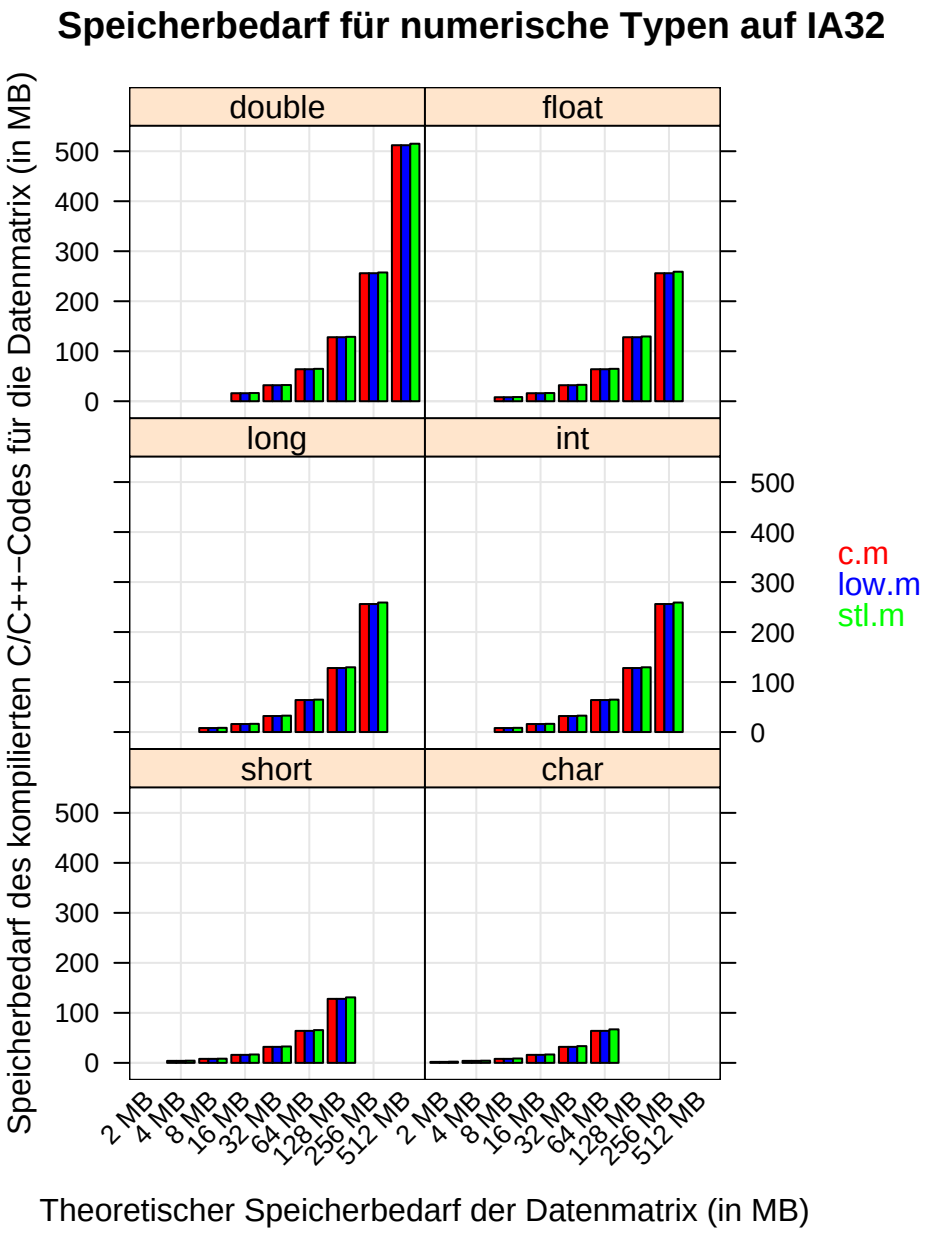


Abbildung 5.4: Darstellung des Speicherbedarfs kompilierten C/C++-Codes auf der 32-bit Plattform **IA32** für Datenmatrizen von Zufallszahlen unterschiedlichen Typs (**double**, **float**, **long**, **int**, **short** und **char**) und verschiedener Größe (angegeben mit dem theoretisch minimal notwendigen Speicherbedarf für die ganze Matrix). Die Verwaltung der Matrix erfolgt zum einen auf sehr niedriger Stufe als eindimensionaler Speicherbereich mit den von C **c.m** (rot), bzw. von C++ **low.m** (blau) zur Verfügung gestellten Mitteln, und zum anderen mit Hilfe des STL-Datenkontainers `vector` **stl.m** (grün), der die höchste Flexibilität und eine Lösung, vergleichbar der mit Hilfe des Collection-Frameworks in Java, bietet. (Siehe Text für nähere Erläuterungen.)

keinerlei Overhead. Selbst die flexible Lösung per STL-Kontainer hat nur einen, gerade im Vergleich zur entsprechenden Java-Lösung, sehr geringen Overhead, wie in Tabelle für beide System-Plattformen zu erkennen ist.

System-Plattform		numerische Datentypen					
		double	float	long	int	short	char
AMD64	Min.	1.011723	1.023441	1.011723	1.023441	1.046879	1.093754
	Mean	1.021502	1.042986	1.021502	1.042986	1.085955	1.171892
	Max.	1.046906	1.093781	1.046906	1.093781	1.187531	1.375031
IA32	Min.	1.005863	1.011723	1.011723	1.011723	1.023441	1.046879
	Mean	1.010759	1.021502	1.021502	1.021502	1.042986	1.085955
	Max.	1.023468	1.046906	1.046906	1.046906	1.093781	1.187531

Tabelle 5.5: Verhältnis des realen Speicherbedarfes der STL-basierten Matrix-Lösung in C/C++ im Vergleich zum theoretisch minimal nötigen Speicher für die Matrix auf den Plattformen **AMD64** und **IA32**. Dargestellt sind das Minimum, das Maximum und der Mittelwert, der sich für die einzelnen numerische Datentypen ergebenden Verteilung über alle in Tabelle 5.2 aufgeführten Datenmatrizen hinweg.

Die hier gezeigten Messungen sollen lediglich den ungefähren Rahmen kennzeichnen, den Java - auch im Vergleich zu einer für diesen Bereich so wichtigen Sprache wie C/C++ - rechenintesiven wissenschaftlichen Anwendungen mit umfangreichen Datensätzen bietet. Dabei stellen, wie in der Aufzählung auf Seite 82 dargelegt, die Speichervarianten **f** und **c** jeweils eine untere und eine obere Schranke dar, zwischen denen sich andere Lösungen positionieren müssen, wie z.B. die Ansätze des Commons-Primitives-Projekt der Apache-Foundation [The06a], GNU Trove [Fri07a] oder Javolution [Dau07a] aus dem Real-Time- und Embedded-Bereich von Java, die ebenfalls einen Versuch unternehmen, möglichst flexible Datenkontainer für primitive Datentypen und einen damit verbundenen geringeren Speicherverbrauch bereitzustellen.

5.3.3 Überblick über numerische Bibliotheken in Java

Wie bereits im Unterpunkt zu numerischen Bibliotheken in der Aufzählung auf Seite 80 erwähnt, spielt die Unterstützung rechenintesiver Anwendungsentwicklung durch numerische Bibliotheken innerhalb einer Sprache eine große Rolle für deren Anwendbarkeit in dem Gebiet des HPC. Der Entwurf eines numerischen Algorithmus, dessen Validierung und die anschließende korrekte, angepasste Abbildung auf eine möglichst effiziente Implementierung in der Zielsprache erfordern oft ein hohes Maß an Spezialwissen, sowie an Zeit- und Arbeitsaufwand. Numerische Bibliotheken stellen ihre Routinen als sogenannte Building-Blocks anderen Anwendungsprogrammierern zur Verfügung. Beispiele solcher verbreiteter und anerkannter Bibliotheken in anderen Programmiersprachen sind die schon erwähnten BLAS [BLA07], LAPACK [Lap07] oder die GSL [Fre07b].

Verschafft man sich einen Überblick über numerische Bibliotheken in Java, so entsteht ein zwiespältiges Bild. Es gab sehr viele Ansätze, von denen ein großer Teil nicht mehr aktiv weiterentwickelt wurde. Zentrale Einstiegsseite für das Auffinden wichtiger Bibliotheken ist der Abschnitt „Libraries“auf der Seite [Jav07a] der Numerics Working Group des Java Grande Forums. Hiervon ausgehend soll eine Übersicht der relevantesten, zumeist freien und für den Anwendungsbereich der Auswertung von Microarray-Daten interessantesten Bibliotheken gegeben werden:

- ☞ **Colt [Cer07]**
  Die Colt-Library wurde am CERN (Europäische Organisation für Kernforschung) entwickelt und ist speziell auf den Bereich der Hochenergiephysik und deren Datenauswertungsprozessen zugeschnitten. Es enthält eine Reihe von Datenstrukturen für die Datenauswertung, grundlegende Methoden der linearen Algebra, mehrdimensionale Arrays, Monte-Carlo Methoden und Histogramm-Routinen. Aus dem Bereich der Statistik gibt es eine Reihe von Methoden der deskriptiven Statistik, verschiedene Zufallszahlengeneratoren, spezielle Funktionen und Integrale sowie unterschiedlichste Verteilungs- und Wahrscheinlichkeitsdichtefunktionen. Insbesondere ist die für den t-Test notwendige Student-Verteilung implementiert, allerdings fehlt jeglicher Ansatz für statistische Inferenz.
  Letztes Release: Version 1.2.0, 09.09.2004

- ☞ **Commons-Math [The07b]**
  Commons-Math ist die Mathematik-Bibliothek des Jakarta-Projektes der Apache Software Foundation. Sie enthält Methoden aus einer Reihe von mathematischen Gebieten, beispielsweise spezielle Funktionen, grundlegende Methoden der linearen Algebra und aus dem Gebiet der numerischen Analyse, Zufallszahlengeneratoren für verschiedenste Zufallsprozesse, komplexe Zahlen, numerische Integrationsmethoden für gewöhnliche Differentialgleichungen und für die Optimierung. Auf dem Gebiet der Statistik finden sich nicht nur deskriptive Verfahren und unterschiedliche Verteilungsfunktionen, sondern auch ein vorbereiteter Ansatz für Methoden der statistischen Inferenz. Die parametrischen Tests t-Test und χ^2-Test sind bereits implementiert.
  Letztes Release: Version 1.1, 2005 - allerdings befindet sich Release 1.2 in aktiver Entwicklung

- ☞ **JAMA [The05]**
  JAMA (**JA**va **MA**trix Package) ist eine oft verwendete Bibliothek für die Arbeit mit Matrizen. Sie ist klein, leicht vertändlich und übersichtlich. Implementiert sind grundlegende Operationen (Subtraktion, Multiplikation, ...) und die Zerlegung von Matrizen (*Singular Value Decomposition* (SVD), *LR-Zerlegung* (LR), *QR-Zerlegung* (QR), ...). Die Entwicklung begann mit Release 1.0.0 im Jahr 1998. Dieser frühe Entwicklungsbeginn hat zur weiten Verbreitung beigetragen.
  Letztes Release: Version 1.0.2, 13.07.2005

☞ **Jampack [Stu]**
Jampack (**Ja**va **M**atrix **Pack**age) hat in etwa die gleichen Ziele wie JAMA, liegt deshalb eigentlich in direkter Konkurrenz zu dieser und zeigt keine aktive Entwicklung mehr.

☞ **JLapack [SD07b]**
JLapack stellt eine mit dem Tool F2J [SD07a] durchgeführte Java-Portierung der bekannten numerischen Fortran-Bibliothek LAPACK [Lap07] nach Java dar. Kernpunkt der Bibliothek sind Methoden der linearen Algebra. Letztes Release: Version 0.8, 27.03.2007

☞ **JMSL - Numerical Library for Java Applications [Vis06a]**
JMSL ist eine kommerzielle rein Java-basierte, numerische Bibliothek, die von der Firma Visual Numerics vermarktet wird. Visual Numerics vertreibt ihre numerische Bibliothek auch in den Sprachen C, C# und Fortran. Von ihrer Ausrichtung her zielt JMSL vor allem in Richtung Datenanalyse in den Bereichen Finanzmarkt, Bioinformatik und Life-Sciences. Zu letzterem findet sich auf der Webseite der JSML[Vis06a] das folgende Zitat:
„The JMSL Numerical Library includes statistical algorithms that will assist statisticians and analysts with applications in areas such as bioinformatics and life sciences where data sets tend to be extremely large and complex.“
Tatsächlich bietet die JSML die breiteste Unterstützung aller hier erwähnten Bibliotheken für statistische Auswertungen, beispielsweise deskriptive statistische Größen (Kurtosis, Median, Maximum, Minimum, Modus, ...), grundlegende Verfahren der statistischen Inferenz, wie parametrische Tests (χ^2-Test, Wilk-Shapiro-Test, Lilliefors Test), nicht-parametrische Tests (Wilcoxon-Test, Vorzeichen-Test) und ANOVA, eine Vielzahl verschiedener stetiger und diskreter Verteilungsfunktionen (t-, Gamma-, Beta-, F-, χ^2-, Binomial-, Poisson-, Normalverteilung ...), Regression, Faktor-Analysis, hierarchisches und k-means Clustering, Kontingenztabellen und Diskriminanzanalyse [Vis06b]. Neben den statistischen Routinen findet sich eine vielfältige Auswahl an Grafik-Funktionen, um die gewonnenen Analyse-Werte darzustellen, darunter Contour-Plots, Box-Plots, Bar-Plots, Polar-Plots, 3D-Plots, Dendrogramme und Heatmaps [Vis06b].
Letztes Release: Version 4.0, 15.06.2006

☞ **MTJ - Matrix Toolkit for Java [The06b]**
Das MTJ zielt darauf ab, Methoden und Routinen der linearen Algebra für die Arbeit mit Matrizen unter Java zur Verfügung zu stellen. Im Unterschied zu JAMA ist das MTJ wesentlich umfangreicher und bietet mehr Typen an Matrizen an. Als Option erlaubt es MTJ, auf nativ kompilierte BLAS-Bibliotheken [BLA07], wie auch LAPACK [Lap07] zuzugreifen, falls diese im System vorhanden sind. Ansonsten nutzt MTJ JLapack für die zugrunde liegenden Berechnungen und verbindet so Portabilität mit der Möglichkeit zur

Leistungsoptimierung.
Letztes Release: Version 0.9.9, 28.11.2006

☞ **JScience [Dau07b]**
JScience selbst basiert auf den schlanken Datenkontainern von Javolution [Dau07a] und hat die Vision, die umfangreichste Bibliothek für alle Wissenschaften zu werden. Das ist auch an der gewählten Package-Struktur zu erkennen. Allerdings sind derzeit nur die grundlegendsten Fundamente realisiert und es fehlen Methoden, die für die Datenauswertung von Microarrays notwendig sind. Interessant ist diese Bibliothek durch ihre Zielstellung für den gesamten wissenschaflichen Bereich, einschließlich der Biologie, durch die darunter liegende Bibliothek Javolution, in der sie realisiert ist und durch die bestehende aktive Entwicklung. Letztes Release: Version 4.1.2, 05.07.2007

☞ **JSci [Hal06]**
JSci verfolgt von seiner Zielsetzung her einen ähnlich generellen Ansatz wie JScience und möchte Methoden für alle Wissenschaftszweige anbieten, so finden sich unter anderem auch Packages für die Biologie und Chemie, die jedoch nicht sehr weit ausgebaut sind. Der Schwerpunkt der vorhandenen Routinen liegt im Bereich der Physik. So gibt es beispielsweise eine Auswahl an Wavelet-Transformationen. Im Unterschied zu JScience bietet JSci jedoch in ihrem Statistics-Package bereits eine Reihe von Verteilungsfunktionen, wie z.B. die F- und die t-Verteilung an. Es liegen noch keine vorbereiteten Strukturen für die statistische Inferenz vor.
Letztes Release: Version 0.943, 26.05.2006

☞ **NUMAL for Java [Lau03]**
Das Buch [Lau03] liefert eine Portierung der Bibliothek NUMAL (**Num**erical procedures in **Al**gol 60) nach Java. Das Buch selbst stellt lediglich die Dokumentation der auf der zugehörigen CD mitgelieferten Bibliothek dar. Insgesamt ist diese Bibliothek umfangreich, aber auf die Bereiche der Physik und Ingenieurwissenschaften zugeschnitten. So fehlen beispielsweise statistische Funktionen gänzlich. Letztes Release: 2003

☞ **SSJ - Stochastic Simulation in Java [L'E07]**
Die SSJ stellt eine Java-basierte Bibliothek zur stochastischen Simulation dar. Entstanden ist sie unter der Leitung von Pierre L'Ecuyer an der Universität Montreal. Sie bietet eine große Auswahl an Funktionen für die Erzeugung von Zufallszahlen verschiedensten Typs (siehe hierzu auch P. L'Ecuyers Kapitel auf Seite 35 bis 70 in [GHM04]), eine Vielzahl unterschiedlicher Verteilungsfunktionen (zum Teil auch bi- und multivariat), einfache Methoden der deskriptiven Statistik, Anpassungstests (Goodnes-of-Fit-Tests), wie den Kolmogoroff-Smirnoff-Test, den χ^2-Test und verschiedene weitere Strukturen zur Ereignis-Simulation, sowie zur Simulation von Monte-Carlo-Prozessen.
Letztes Release: Version 1.2.2, 25.04.2007

Alle hier genannten Bibliotheken sind aufgrund ihrer erzielten Bedeutung, ihrer noch aktiven Entwicklung oder ihrer Funktionsvielfalt von aktuellem Interesse. Insgesamt ist zu beobachten, dass viele numerische Java-Bibliotheken keine rege aktive Weiterentwicklung erfahren haben und dass es bisher keine Bibliothek gibt, die alle numerischen Teilgebiete ausreichend integriert. Gerade der schnelle und frühe Erfolg des Java-SDKs von Sun [Sun07a] zeigt den Vorteil der Verbreitung einer Sprache zusammen mit einer mächtigen und einheitlichen, zugehörigen Klassenbibliothek. Jede der aufgezählten Bibliotheken hat Vorteile bzw. Stärken für bestimmte Anwendungsbereiche und ein Java-Projekt muss deshalb meist auf mehrere verschiedene Bibliotheken zurückgreifen.

Generell ist jedoch der Bereich der statistischen Verfahren und Methoden nur unzureichend implementiert. JMSL ist die am weitesten gereifte und in ihrem Funktionsangebot interessanteste Bibliothek für dieses Einsatzgebiet. Jedoch steht einem Einsatz im Umfeld freier und Open-Source-Projekte, wie z.B. auch Mayday, ihre kommerzielle und unfreie Lizenz entgegen. Dadurch verbleibt als geeignetste, freie Bibliothek für die Verwendung in Mayday Commons-Math. Commons-Math besitzt, neben den auch in anderen Bibliotheken zu findenden Verteilungsfunktionen und Methoden zur deskriptiven Statistik, schon ein vorbereitetes Framework für die statistische Inferenz und hat bereits einen implementierten t- und χ^2-Test aufzuweisen.

Außerdem verfolgt Commons-Math zugleich einen generellen Ansatz als mathematische Bibliothek des Jakarta-Projektes und fügt sich gut in das Gesamtkonzept der über das Jakarta-Projekt angebotenen Bibilotheken ein. Im Gegensatz dazu richtet sich die auch aus statisitischer Sicht interessante Bibliothek SSJ mehr auf den speziellen Fall der Zufallszahlenerzeugung und die stochastische Simulation. Für die Auswertung von Genexpressionsdaten spielen jedoch auch Methoden der linearen Algebra, wie die Eigenwertzerlegung oder die *Singular Value Decomposition* (SVD), eine große Rolle. Andere interessante Bibliotheken mit neuen Ansätzen, z.B. JScience, bieten noch nicht den notwendigen Funktionsumfang, um bereits als mathematische Bibliothek für ein Projekt wie Mayday in Frage zu kommen.

5.4 Parametrische und nichtparametrische Tests

Bereits in Abschnitt 5.2 wurde auf die verstärkte Anwendung rechenintensiver Methoden in der Statistik, wie das Bootstrapping und Resampling eingegangen. Konzeptionell eng damit verwandt und gleichfalls mit einem hohen Rechenaufwand verbunden sind in der statistischen Inferenz die nichtparametrischen bzw. verteilungsfreien Testmethoden [BLB00]. Ihnen kommt ganz allgemein in der angewandten Statistik [LD98] und besonders in der Auswertung von Genexpressionsarrays eine immer wichtiger werdende Rolle zu. Als Beleg hierfür sei aus der Einführung der aktuellen Auflage von [Goo05] zitiert:

> ... In previous editions of this text, my rhetoric was somewhat tentative.
> I was saying, in effect, „Gee guys, permutation methods provide a prac-

> tical real-world alternative to asymptotic parametric approximations. Why not give them a try?"But today, the theory, the software, and the hardware have come together. Distribution-free permutation procedures are the primary method for testing hypotheses. Parametric procedures and the bootstrap are to be reserved for the few situations in which they may be applicable. Four factors have forced this change:
>
> 1. Desire by workers in applied fields to use the most powerful statistic for their applications. Such workers may not be aware of the fundamental lemma of Neyman and Pearson, but they know that the statistic they want to use – a complex score or a ratio of scores, does not have an already well-tabulated distribution.
> 2. Pressure from regulatory agencies for the use of methods that yield exact significance levels, not approximations.
> 3. A growing recognition that most real-world data are drawn from mixtures of populations.
> 4. A growing recognition that missing data is inevitable, balanced designs the exception.
>
> ... On the other hand, certain relatively robust parametric tests such as Student's t continue to play an essential role in statistical practice.

Der große Vorteil nichtparametrischer Tests sind ihre geringeren Anforderungen an die Voraussetzungen, die für ihre Durchführbarkeit erfüllt sein müssen. Parametrische Tests beinhalten die Annahme eines konkreten, parametrisierbaren Grundmodells bezüglich der Verteilung eines Merkmals in der Grundpopulation aus der die Stichprobe stammt. Nicht selten sind diese Modellannahmen erst bei größeren Stichprobenanzahlen erfüllt und die Statistik lediglich asymptotisch exakt. In den für die Praxis wichtigsten Fällen handelt es sich dabei meistens um ein Modell, das auf der Normalverteilung oder einer von dieser abgeleiteten Verteilung beruht. Demgegenüber setzen nichtparametrische Methoden zumeist lediglich die Stetigkeit und die Homomerität (Verteilungsgleichheit zwischen den Stichproben) der den zu untersuchenden Stichproben zugrunde liegenden Verteilungsfunktionen voraus. Außerdem können sie auch mit Daten umgehen, die nur mit einem ordinalen oder nominalen Skalenmaß erhoben wurden.

Kapitel 4 von [BLB00] identifiziert drei grundlegende Aspekte für die Entscheidung über den Einsatz parametrischer oder nichtparametrischer Tests und entwickelt darauf aufbauend ein schrittweises dreistufiges Entscheidungsschema, das hier kurz zusammenfassend angegeben werden soll:

1. **Messskalenniveau**
   Liegt das Messskalenniveau unterhalb einer kardinalen Skala (Verhältnis- und Intervallskala [BLB00]), sollte ein nichtparametrischer Test eingesetzt werden. Das bedeutet bei Vorliegen von nominalen oder ordinalen Daten ist verteilungsfrei zu testen, aber bereits bei der Verletzung der Äquidistanz bei Intervallskalen kann dies angeraten sein - trotz relativer Robustheit verschiedener

parametrischer Tests, wie z.B. des t-Tests, gegnüber Verletzungen des Intervallskalenpostulats.

2. **Mathematisch-statistische Voraussetzungen des Test**
   Die Aussagen statistischer Tests sind nur im Rahmen ihrer mathematisch-statistischen Voraussetzungen korrekt. Diese Voraussetzungen müssen vorher überprüft werden. Für parametrische Tests handelt es sich hierbei oft um das Vorliegen einer Normalverteilung und bei stichprobenverlgeichenden Tests um die Varianzhomogenität der Referenzpopulationen. Ersteres ist besonders wichtig für kleine Stichprobenumfänge ($N \leq 30$), bei größeren Stichprobenumfängen wirkt sich der zentrale Grenzwertsatz (S. 144 in [BHPT95]) positiv aus. Für nichtparametrische Tests reicht oft als Voraussetzung die Stetigkeit der zugrundeliegenden Verteilung und die Homomerität der Stichproben aus. Sind die Voraussetzungen für einen parametrischen Test verletzt, sollte man verteilungsfrei testen. Sind auch deren Voraussetzungen nicht gesichert, kann es angebracht sein, ganz auf statistische Tests zu verzichten und lediglich Trends oder Hypothesen aufgrund einer geeigneten visuellen Datenexploration zu formulieren.
   Häufig wird in Lehrbüchern das Vorschalten von Normalverteilungs- und Varianzhomogenitätstests empfohlen. Deren Anwendung gerade dann kritisch zu sehen ist, wenn es sich dabei wiederum um parametrische Tests mit bestimmten Voraussetzungen handelt, wie z.B. der Bartlett-Test [SH06]. Aus diesem Grund wird seit einiger Zeit geraten, sich nicht auf diese voraussetzungsüberprüfenden Tests zu beziehen und die Verteilungsvoraussetzungen, wie auch die Stichprobencharakteristika (beispielsweise die Korrelation von Umfang und Mittelwert) durch visuelle Exploration der Daten zu prüfen [BLB00, WH07]. Deutliche Verletzungen sind so erkennbar. Treten nur kleine Abweichungen auf, so kann bei großen Stichprobenumfängen oder aber bei der Verwendung robusterer parametrischer Tests trotzdem parametrisch getestet werden.
   Eine weitere Möglichkeit, Voraussetzungsverletzungen entgegenzuwirken, besteht darin, durch geeignete Skalentransformation, wie z.B. einer logarithmischen Transformation, die Verteilungs- und Varianzcharakteristik hinreichend zu korrigieren. [BLB00] empfiehlt jedoch solche Transformationen nicht durchzuführen, wenn sie inhaltlichen Argumenten grob widersprechen und die Interpretation zu stark erschweren. Entsteht die Problematik aus konstruktiven Gründen des Experiments bzw. der Messung heraus, so ist eine Transformation sogar empfehlenswert und kann die Interpretation verbessern.

3. **Robustheit**
   Die Robustheit eines Tests bezeichnet dessen Unempfindlichkeit gegenüber Verletzungen seiner Voraussetzungen und gegenüber ungewöhnlichen Stichprobeneigenschaften [BLB00], d.h. dass für einen robusten Test bei Verletzung seiner Voraussetzungen die Quote der Fehlentscheidungen nur gering wächst.

Die Robustheit statistischer Tests wird mit der Monte-Carlo-Methode abgeschätzt und kann für den konkreten Fall schwierig sein.
Parametrische Test mit einer F-, einer χ^2- oder t-Verteilung gelten als relativ robust gegenüber der Verletzung einer ihrer Vorraussetzungen. Sind jedoch gleichzeitig 2 Voraussetzungen nicht erfüllt, oder liegt eine Voraussetzungsverletzung und eine ungewöhnliche Stichprobeneigenschaft vor, so muss mit einer hohen Beeinträchtigung der Resultate gerechnet werden.

Ganz allgemein erfolgte die Anwendung von Tests, insbesondere des t-Tests, als eine statistische Methode für die Detektion von differentiell exprimierten Genen, schon recht früh [TOTZ01, Pan02]. Expressionsdaten, die mit Microarrays gewonnen wurden, sind jedoch sehr oft nicht normalverteilt und das sogar nach einem bereits erfolgten, geeigneten Vorverarbeitungsverfahren [HTLS01, TOTZ01, Pan02]. Aus diesem Grunde ist in einer Reihe von Veröffentlichungen die Verwendung nichtparametrischer Tests empfohlen worden [TGB+02, XL03, GPH+03, NL04, NS04, BAAH04, BH05], wie z.B. der *Wilcoxon-Mann-Whitney-Test* (WMW-Test) [Wil45, MW47], der *Fisher-Pitman-Permutationstest* [Pit37], *Baumgartner-Weiß-Schindler-Test* (BWS-Test) [BS98, Neu02] oder das RP [BAAH04, BH05]. Durch ihre Unempfindlichkeit gegenüber Ausreißern weisen rangbasierte Verfahren einen weiteren Vorteil in der Genexpressionanalyse auf. Ausreißer sind ein sehr weit verbreitetes Problem bei Microarraydaten.

5.4.1 Der Wilcoxon-Mann-Whitney-Test für die Rangsummen zweier unabhängiger Stichproben

Aufgrund seiner Bedeutung für den folgenden Abschnitt, soll hier noch einmal kurz das Wesentliche und wichtige Eigenschaften des WMW-Tests zusammengefasst werden. Der 1945 von Wilcoxon [Wil45] vorgeschlagene Test für die Rangsumme zweier Stichproben wurde 1947 von Mann und Whitney [MW47] in seiner sogenannten U-Form ausformuliert. Beim Wilcoxon- und Mann-Whitney Test handelt es sich somit eigentlich um äquivalente Formen ein und desselben Tests, was in der modernen statistischen Literatur oft durch die gemeinsame Nennung als WMW-Test zum Ausdruck gebracht wird. Die jeweils verwendeten Statistiken lassen sich leicht ineinander umrechnen.

Seien $X = \{x_1, x_2, \ldots, x_m\}, X \in \mathbb{R}^m$ und $Y = \{y_1, y_2, \ldots, y_n\}, Y \in \mathbb{R}^n$ zwei Stichproben für eine *Zufallsvariable* (ZV) Z, die hinsichtlich der den Stichproben zugrundeliegenden Verteilung von Z verglichen werden sollen. Sind weiterhin die folgenden Voraussetzungen an die Verteilungsfunktionen $F(x)$ und $G(x)$ von Z die den Stichproben X und Y zugrunde liegen erfüllt:

- $F(x)$ und $G(x)$ sind stetig.
- $F(x)$ und $G(x)$ sind gleich bzw. hinreichend ähnlich. (Nähere Erläuterungen hierzu erfolgen weiter unten.)

So kann mit dem nichtparametrischen WMW-Test, bei zweiseitiger Fragestellung - die Nullhypothese

$$H_0 : \forall x \in \mathbb{R} \text{ gilt } F(x) = G(x) \quad \text{gegen} \quad H_A : \exists\, x^* \in \mathbb{R} \text{ mit } F(x^*) \neq G(x^*) \tag{5.1}$$

abgeschätzt werden. Formuliert man diesen allgemeineren Ansatz des WMW-Test in die für die U-Form geeignetere Variante um, so ergibt sich die Abschätzung

$$H_0 : P(x > y) = \frac{1}{2} \quad \text{gegen} \quad H_A : P(x > y) \neq \frac{1}{2} \tag{5.2}$$

D.h. ausformuliert, unter Annahme der Gültigkeit von H_0 - also der gleichen Lokation und Form von G und F, insbesondere gleichem *Lagemaß* und gleicher *Dispersion* - ist die Wahrscheinlichkeit bei einer beliebigen Realisierung der ZV für die Stichproben X und Y die größere in X erhalten zu haben exakt $\frac{1}{2}$. Als Teststatistik $U(X, Y)$ ergibt sich für die U-Form des Tests:

$$U(X, Y) = U_{XY} = U_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} \{I(x_i < y_j) + I(x_i = y_j)\} \tag{5.3}$$

wobei I die Indikatorfunktion für Aussagen Ω repräsentiert:

$$I : \{wahr, falsch\} \rightarrow \{0, 1\} \quad \texttt{mit} \quad I(\Omega) = \begin{cases} 1 & ; \quad \Omega \mapsto wahr \\ 0 & ; \quad \Omega \mapsto falsch \end{cases} \tag{5.4}$$

Der zweite Term innerhalb der geschweiften Klammer trägt sogenannten *Bindungen* Rechnung, die insbesondere bei Daten auf einer Intervall- und Ordinalskala auftreten können.

Man kann U jedoch auch aus der Rangsumme T (Wilcoxon-Form) der kombinierten Stichprobe von X und Y erhalten (siehe S. 392 ff in [SH06]).

Bildet man aus den beiden Stichproben X und Y aufsteigende sortierte (gerankte) Zahlenfolge C

$$C = \{c_1 \leq c_2 \leq c_3 \leq \cdots \leq c_N\} \tag{5.5}$$

vom Umfang $N = m + n$, dann ordnet die Funktion $R(x)$ jedem Beobachtungswert seinen Rang bzw. seine Position innerhalb dieser Zahlenfolge zu. Beim Auftreten von Bindungen wird allen Zahlen der Bindung die mittlere Positon als Rang zugeordnet.

$$R(x) = \frac{\min(i|\forall c_i = x) + \max(i|\forall c_i = x)}{2}, x \in X \cup Y \tag{5.6}$$

Mit Hilfe dieser Funktion läßt sich die Teststatistik T nach Wilcoxon [Wil45] bestimmen als

$$T_X = \sum_{i=1}^{m} R(x_i) \tag{5.7}$$

Die Umrechnung zwischen der T- und der U-Statistik erfolgt dann relativ einfach aus :

$$\begin{aligned} U_{YX} &= (R(x_1) - 1) + (R(x_2) - 2) + \ldots + (R(y_m) - m) \\ &= \sum_{i=1}^{m} R(x_i) - \frac{m(m+1)}{2} = T_X - \frac{m(m+1)}{2} \end{aligned} \tag{5.8}$$

und mit dem symmetriebedingt aus Formel 5.3 folgenden

$$U_{XY} + U_{YX} = U_1 + U_2 = mn \tag{5.9}$$

schließlich

$$U_{XY} = mn + \frac{m(m+1)}{2} - T_X \tag{5.10}$$

Aus Gleichung 5.3 folgt unter H_0 für den Erwartungswert $E(U)$ und die Varianz $Var(U)$ von U:

$$E(U) = \frac{mn}{2} \quad \texttt{und} \quad Var(U) = \frac{mn(m+n+1)}{12} \tag{5.11}$$

Die Berechnung von Annahme- bzw. Ablehnungsbereich, sowie dem p-Value wird für kleine Stichprobenumfänge auf der Grundlage der zugehörigen WMW-Verteilung durchgeführt (siehe Abschnitt 5.5.2 zu näheren Erläuterungen der WMW-Verteilung), die für einige kritische Werte α beispielsweise in [SH06] tabelliert sind. Für Stichprobenumfänge von $m + n > 60$ kann man die asymptotische Näherung über die Standardnormalverteilung $\mathcal{N}(0, 1)$ benutzen (vgl. S. 395 von [SH06]):

$$\frac{\left|U - \frac{mn}{2}\right|}{\sqrt{\frac{mn(m+n+1)}{12}}} \sim \mathcal{N}(0, 1) \tag{5.12}$$

Aus Gründen des hohen Aufwands und der Kosten für die Replikation liegen erfahrungsgemäß die Stichprobengrößen aktueller Microarray-Studien weit unter diesem Wert und erreichen bestenfalls mittlere Stichprobenumfänge [ZP03, BH05]. Damit machen sie eine exakte Berechnung mit Hilfe der WMW-Verteilung notwendig. Im Bereich der Genexpressionsanalyse wird häufig die in [TGB+02] getroffene Aussage zitiert, dass bereits ab einem Stichprobenumfang von 8 pro Gruppe die Verwendung der Näherung genutzt werden kann. Dem widerspricht aber deutlich die in [BLS00] durchgeführte empirische Studie an verschiedenen Softwaretools zur Berechnung des WMW-Tests. Der dort verwendete Datensatz hatte die Stichprobenumfänge $n = m = 12$ und die Testergebnisse reichten, aufgrund der unterschiedlichen, von der Software verwendeten Verfahren (exakte Berechnung und verschiedene Näherungsmethoden), von insignifikant bis signifikant zum 5%-Niveau.

Allgemein gilt der WMW-Test als die verteilungsfreie Alternative zum parametrischen t-Test auf Gleichheit der Mittelwerte zweier Stichproben [SH06]. Diese Interpretation ist jedoch nicht ganz exakt und führt häufig zu einer Fehlinterpretation [SF98]. Der WMW-Test beurteilt weder die Differenz der Mittelwerte, noch

die Differenz der Mediane, sondern lediglich die Rangsortierung beider Stichproben. Gerade die falsche Annahme als reinem Vergleich der Stichproben-Mediane ist weit verbreitet und findet sich sowohl in Lehrtexten zur Statistik wie auch in den Dokumentationen verschiedener statistischer Softwaretools wieder [BLS00].

Vergleicht man die Formulierung des t-Tests bei zweiseitiger Fragestellung,

$$H_0 : \mu_x = \mu_y \quad \text{gegen} \quad H_A : \mu_x \neq \mu_y, \tag{5.13}$$

wobei gilt $\mu_x = E(X)$, $X \sim \mathcal{N}(\mu_x, \sigma_x)$ und $\mu_y = E(Y)$, $Y \sim \mathcal{N}(\mu_y, \sigma_y)$ [BHPT95], mit der der der Hypothesen beim WMW-Test in Gleichung (5.1) fällt auf, dass beim WMW-Test nicht nur die Lokation, sondern auch die genaue Form der Verteilungsfunktionen H und F gleich sein müssen. Das heißt, dass beim WMW-Test auch alle *zentralen Momente* höherer Ordnung von H und F einen entscheidenden Einfluss auf den Ausgang des Tests haben, insbesondere die Dispersion bzw. Varianz. Eine ähnlich hohe Anforderung an die Verteilungsfunktionen der beiden Stichproben stellt auf der Seite des t-Tests lediglich *Students* zweistichproben t-Test, für den $\sigma_x = \sigma_y$ (entspricht $\sigma_x^2 = V(X) = V(Y) = \sigma_y^2$) gefordert wird.

Nur unter der Bedingung einer sehr ähnlichen Verteilungsform und gleicher Varianzen $\sigma_x = \sigma_y$ beurteilt der WMW-Test die Differenz der Mediane als Maß der zentralen Tendenz der zugrunde liegenden Verteilungen beider Stichproben.

Lässt man diese Forderung fallen, so liegt eine Fragestellung vor, die in der statistischen Literatur als *Behrens-Fisher-Problem* bezeichnet wird [Beh29, Fis39]. Es sind eine Reihe von Ansätzen zur Lösung des Behrens-Fisher-Problems für parametrische Tests vorgeschlagen worden [FH56, Jam59, Pat65, MS70, Sch70, DW75]. Einer der verbreitetsten davon ist der Ansatz, der von Welch [Wel38, Wel47], Aspin [Asp48, AW49] und Sattertwaihte [Sat46] ausgearbeitet worden ist. Welch's t-Test gilt als eine sehr robuste Variante des t-Tests [Wan71, Rux06] und wird für den allgemeinen, praktischen Einsatz empfohlen [DW75, Sch70, BR87, Rux06], wenn er auch nicht in allen Fällen ein optimales Ergebnis liefert [MS70, SF98].

Im nichtparametrischen Fall, bzw. als mögliche Lösung für das verallgemeinerte Behrens-Fisher-Problem, ist der WMW-Test nicht anwendbar, da seine Typ I Fehlerrate durch die Heteroskedastizität verändert wird [SF98, Kas01]. Insbesondere existiert bei unterschiedlichen Stichprobenumfängen m, n eine deutliche Interaktion zwischen diesen und den Varianzen σ_x, σ_y, in deren Abhängigkeit die Typ I Fehlerrate sehr schwankt [SF98, Kas01, BM00, Rux06]. Genau dieses Verhalten zeigen auch einige Permutationstests und machen deren Anwendung bei Vorliegen eines Behrens-Fisher-Problems fraglich [Hay00]. Aus diesem Grund sind andere rangbasierte Testprozeduren als Lösung vorgeschlagen worden, wie z.B. von Fligner und Policello der *Robust Rank-Order-Test* [FP81, Fel03, Fel05] und von Brunner und Munzel der *Brunner-Munzel-Test* (BM-Test) [BM00]. Beispiele für die Anwendung des BM-Tests finden sich in [RBN07] und speziell im Bereich der Microarrayanalyse in [NL04]. In dieser Veröffentlichung wurde der BM-Test in einem zweistufigen Verfahren mit einem BWS-Test kombiniert und sollte dessen Empfindlichkeit gegenüber Heteroskedastizität ausgleichen. Der BWS-Test ist ein weiterer nichtparametrischer Test, der etwas weniger konservativ ist als der WMW-Test und

dabei mindesten eine ebensolche bzw. höhere Power aufweist als der WMW-Test [NL04, NS04]. [NS04] empfiehlt die Anwendung des BWS-Tests explizit für die Analyse von Microarray-Daten.

5.5 Implementierung statistischer Tests in Java

Wie der Zusammenfassung von Abschnitt 5.3.3 zu entnehmen ist, gibt es im Bereich der frei zugänglichen Software keine numerischen Bibliotheken in Java, die über eine ausreichend vielfältige Implementierung von Methoden der statistischen Inferenz verfügen, wie sie allein standardmäßig zur Auswertung von Microarrays angewendet werden. Insbesondere verteilungsfreie Verfahren fehlen. Wie ebenfalls am Ende von Abschnitt 5.3.3 dargelegt und begründet, stellt aus der für diese Arbeit interessierenden Sicht die Bibliothek Commons-Math des Jakarta-Projektes, den besten Ausgangspunkt für eine Erweiterung bzw. Ergänzung durch zusätzliche Methoden des statistischen Testens dar.

5.5.1 Apache Commons-Math als Ausgangspunkt für statistische Erweiterungen

Grundlage statistischer Tests bilden die jeweils zugehörigen Verteilungsfunktionen. In Commons-Math sind aktuell (Entwicklerversion 1.2., Package **org.apache.commons.math.distribution**) bereits die folgenden Verteilungsfunktionen implementiert, diskret:

- Binomialverteilung
- Hypergeometrische Verteilung
- Poisson-Verteilung
- Negative Binomialverteilung (Pascal-Verteilung)

und kontinuierliche:

- Normalverteilung
- Exponentialverteilung
- Gammaverteilung
- Studentsche t-Verteilung
- χ^2-Verteilung
- F-Verteilung
- Cauchy-Verteilung (Cauchy-Lorentz-Verteilung)

- Weibull-Verteilung

Darauf aufbauend enthält das für statistische Tests vorgesehene Package **org.apache.commons.math.stat.inference** der aktuellen Entwicklerversion lediglich die Implementierungen der parametrischen Tests:

- t-Test [SH06] mit den möglichen Fragestellungen bzw. Einstellungen
  - Einstichproben- oder Zweistichprobentest
  - einseitig oder zweiseitig
  - gepaart oder ungepaart Beobachtungen beim Zweistichprobentest
  - homoskedastisch (Student'scher t-Test mit gepoolter Varianz als Schätzer für die Standardabweichung σ^* der Verteilung in der Population) oder heteroskedastisch (Welch-Test – Abschätzung von σ^* mit Hilfe der Welch-Satterthwaite-Aproximation der Freiheitsgrade [Wel38, Sat46, Wel47])
  - Entscheidung bezüglich eines vorher festgelegtes Sinifikanzniveaus α oder Rückgabe eines p-Values

- χ^2-Test [SH06, Coc52]
  - χ^2-Anpassungstest (vgl. [SH06] S. 333 ff, [Pea11])
  - χ^2-Test auf Unabhängigkeit für eine Zweiwegtafel (vgl. [SH06] S. 519 ff, [Fis22, Yat34])

Verteilungsfreie Tests sind nicht vorhanden und für eine Implementierung exakter nichtparametrischer Tests fehlen bereits die dazu benötigten Verteilungsfunktionen.

Aufgrund des in Abschnitt 5.4 geschilderten breiten Einsatzes und trotz der in Abschnitt 5.4.1 dargestellten Besonderheiten und Einschränkungen, stellt der WMW-Test [Wil45, MW47] für die Rangsummen zweier unabhängiger Stichproben im Hinblick auf die Anwendung für die Analyse von Microarraydaten die wichtigste nichtparametrische Testmethode dar [TGB+02]. Software für die Auswertung von Microarrays muss deshalb Routinen für seine Durchführung anbieten, weshalb die Zielsetzung darin bestand, Commons-Math in dieser Richtung zu erweitern.

Notwendige Voraussetzung für die Implementierung des vollständingen WMW-Tests ist die Realisierung einer möglichst effizienten Funktion für die Verteilung der Rangsumme des WMW-Tests. Soweit wie möglich sollte dabei aus den in Abschnitt 5.4.1 genannten Gründen die exakte Verteilung nutzbar sein, mindestens jedoch für kleine bis mittlere Stichprobenumfänge, und erst bei großen Stichprobenumfängen $n+m > 60$ auf die asymptotische Näherung (vgl. Formel 7.109 bis 7.111 in [SH06]) über die Standardnormalverteilung zurückgegriffen werden.

Auch die Ergebnisse von [BLS00] legen in Verbindung mit den zwischenzeitlich gemachten Fortschritten in der Leistungsfähigkeit der Hardware das Anstreben einer exakten Lösung so weit wie möglich nahe.

5.5.2 Effiziente Implementierung der exakten WMW-Verteilung in Java

Im vorliegenden Abschnitt wird die WMW-Verteilung für Stichproben kontinuierlicher Verteilungsfunktionen F, G und einem „idealisierten Messprozess“ auf einer Verhältnisskala ohne das Auftreten von Bindung betrachtet. Dies ist der einfachere Fall und für eine entsprechend hohe numerische Auflösung der Genexpressionswerte bei geringem Stichprobenumfang ist die Wahrscheinlichkeit für Bindungen auch gering. Sehr viel wahrscheinlicher werden Bindungen jedoch, falls die Daten in einem Vorverarbeitungsschritt in Klassen eingeteilt (gebinnt) und damit auf eine ordinale Skale transformiert wurden.

Mathematische Grundlagen

Schränkt man sich darauf ein, dass Bindungen nicht vorkommen, entfällt der zweite Term von Formel 5.3 für die Teststatistik nach Mann-Whitney [MW47] $U(X, Y)$ und sie vereinfacht sich zu

$$U(X,Y) = U_{XY} = \sum_{i=1}^{m} \sum_{j=1}^{n} I(x_i < y_j) \tag{5.14}$$

Aus Sicht der Rangsumme nach Wilcoxon bedeutet diese Beschränkung für Formel 5.5 das Entfallen der Gleichheit und jedes gezogene Stichprobenpaar X und Y führt zu einer eindeutigen Abfolge. Damit ist es möglich, von den konkreten, numerischen Verhältnissen innerhalb einer Stichprobe zu abstrahieren und nur noch zwischen den Elementen der Stichproben $a \in X$ und $b \in Y$ zu unterscheiden, da für die Rangsumme von T_X lediglich die Positionen der a-Symbole innerhalb der gemeinsamen, geordneten Zahlenfolge $C = \{c_1, c_2, \ldots, c_N\}$ von Bedeutung sind:

$$\underbrace{\{a, a, \ldots, a\}}_{m} \times \underbrace{\{b, b, \ldots, b\}}_{n} \mapsto \underbrace{\{a|b, a|b, \ldots, a|b\}}_{m+n=N} = C \tag{5.15}$$

Die Anzahl der insgesamt möglichen Sortierreihenfolgen entspricht der Zahl an möglichen Zuordnungen der Positionen $1, \ldots, N$ zu allen m Elementen von X

$$A_{ges} = \binom{m+n}{m} = \binom{N}{m} \tag{5.16}$$

Insgesamt hängt die Verteilung der einzelnen Rangsummen T_X und der zugehörigen U_{YX}-Werte nur noch vom Umfang m der ersten Stichprobe und der Gesamtzahl N an Beobachtungen ab. Es lässt sich daraus eine Rekursionsformel für die Zahl an möglichen Anordnungen für einen bestimmten U_{YX}-Wert $A'(u|N, m)$ (eine bestimmte Rangsumme T_X $A'(t|N, m)$) ableiten.
Insbesondere gilt die Symmetrie bezüglich m und n:

$$A(u|N,m) = A(u|N, N-m) = A(u|N,n) \tag{5.17}$$

Entstammt die letzte Zahl c_N der gemeinsamen Anordnung C der Stichprobe Y, $c_N = y_n$, ändert sich am Wert der Statistik, nach Formel 5.14 nichts:

$$U_{YX,N} = \sum_{i=1}^{m}\sum_{j=1}^{n-1} I(y_j < x_i) + \sum_{i=1}^{m} I(y_n < x_i) = U_{YX,N-1} + 0 = U_{YX,N-1},$$

entstammt es jedoch X, $c_N = x_m$, so gilt

$$U_{YX} = \sum_{i=1}^{m-1}\sum_{j=1}^{n} I(y_j < x_i) + \sum_{j=1}^{n} I(y_j < x_m) = U_{YX,N-1} + \sum_{j=1}^{n} 1 = U_{YX,N-1} + n.$$

Und damit ergibt sich die folgende Rekursionsgleichung

$$\{c_1, c_2, \ldots, c_{N-1},\ c_N\} \mapsto A'(u|N,m) = \begin{cases} A'(u|N-1,m) & c_N \in Y \\ + & \\ A'(u-n|N-1,m-1) & c_N \in X \end{cases} \tag{5.18}$$

Eine weitere wichtige Relation für $A'(u|N,m)$ folgt sofort aus Formel 5.9:

$$0 \le u \le mn : A'(u|N,m) = A'(nm-u|N,m) \tag{5.19}$$

Das bedeutet $A'(u|N,m)$ ist symmetrisch zu $\frac{1}{2}mn$.
Definiert man

$$A(u|N,m) = A'(U \le u|N,m) = \sum_{u^*=0}^{u} A'(u^*|N,m) \tag{5.20}$$

als die kumulierte Anzahl an möglichen Anordnungen, die auf einen Wert der Statistik U führen, der kleiner oder gleich u ist, ergibt sich die gleiche Rekursionsformel wie 5.18. Somit kann man in der Symbolik von Formel 5.18 bleiben, und es ergeben sich als Randbedingungen der Rekursion:

$$u < 0 \quad : \quad A(u|N,m) = 0 \tag{5.21}$$

$$u \ge nm \quad : \quad A(u|N,m) = \binom{N}{m} \tag{5.22}$$

$$0 \le u \le nm, n = 1 \vee m = 1 \quad : \quad A(u|N,m) = u + 1 \tag{5.23}$$

Unter Gültigkeit der Nullhypothese H_0 sind alle Anordnungen von a und b in C gleich wahrscheinlich. Dies ergibt zusammen mit Formel 5.16 und 5.18 für die WMW-Verteilung ohne das Auftreten von Bindungen

$$WMW_f(U \le u|N,m) = \frac{A(u|N,m)}{A_{ges}} = \frac{A(u|N,m)}{\binom{N}{m}}. \tag{5.24}$$

Damit folgt aus Formel 5.19 die Beziehung

$$\frac{mn}{2} < u \le mn : WMW_f(U \le u|N,m) = 1 - WMW_f(U \le mn - u - 1|N,m). \tag{5.25}$$

Algorithmus für die Berechnung von $A(u|N,m)$

Limitierender Faktor in Formel 5.24 ist die Bestimmung von $A(u|N,m)$. Hierfür gilt es, eine ausreichend performante Implementierung zu finden. Diese sollte schnell und möglichst speicherschonend sein, um eine exakte Berechnung des WMW-Test auch für größere Stichprobenumfänge zu gestatten. Die Ergebnisse von [BLS00] belegen die Notwendigkeit, so weit wie möglich auf eine Näherung zu verzichten und die exakte WMW-Verteilungsfunktion für den Test zu benutzen.

Ein interessanter Ansatz für die Berechnung der exakten Verteilung nichtparametrischer Tests präsentieren van de Wiel et al. in [vdWBvdL99] über die Benutzung der *Erzeugendenfunktionen* von *Partitionsfunktionen*. Insbesondere [Buc99] versucht einen Vergleich verschiedener Rekursionsformeln mit dem Ansatz über die Erzeugendenfunktion für die WMW-Verteilung zu führen. Als Ergebnis scheint nach [Buc99] dieser neue Ansatz den verschiedenen Rekursionsformeln gegenüber überlegen zu sein, während die Rekursionsformeln untereinander als gleichwertig einzustufen sind. Folgerichtig geben die Autoren von [Buc99] auch die Empfehlung, die Erzeugendenfunktion für die Berechnung der WMW-Verteilung zu nutzen. Jedoch ist kritisch anzumerken, gemacht wurde der Vergleich der Methoden innerhalb von Mathematica [Wol07], einem umfangreichen mathematisch-naturwissenschaftlichen Programmpaket, das unter anderem Methoden für das symbolische Rechnen enthält. Das Testen der Rekursionsformeln in der Mathematica eigenen, maschinenfernen Hochsprache kann einen bedeutenden Mehraufwand erzeugen, der die Rekursionsformeln unnötig benachteiligt. Dieses Problem wurde von den Autoren auch erkannt und zugestanden.

Ein weiterer Nachteil dieses Ansatzes für eine allgemein zugängliche, offene Implementierung ist das notwendige Vorhandensein eines computeralgebraischen Systems bzw. einer Bibliothek für symbolische Computeralgebra, wie beispielsweise Mathematica. Einerseits existiert kein vergleichbar leistungsfähiges freies System und andererseits würde diese zusätzliche Anforderung einen weiteren Installationsschritt darstellen.

Zusammenfassend bleibt damit die Implementierung der Rekursionsformel als die geeignetste Wahl zur Erweiterung der Apache Commons-Math Library.

Aus dem Anwendungsgebiet der Genexpressionsanalyse mittels Microarrays heraus kann eine zusätzliche Anforderung an die Implementierung formuliert werden: Tests auf eine differenzielle Expression zwischen zwei Gruppen werden zumeist genweise für alle auf dem Chiptyp aufgebrachten Gensonden durchgeführt, d.h. es wird in einem Analysevorgang ein statistischer Test sehr oft angewendet [Efr04]. Die jeweiligen Stichprobengrößen für X bzw. Y können dabei jedoch durch die Ausfälle von Messungen (irreguläre Spots, kontaminierter Hintergrund der Spots, ... usw.) sehr unterschiedlich sein: $0 < m^* \leq m$ und $0 < n^* \leq n$. Dies bedeutet, dass es sich lohnt, eine einmal berechnete Rekursion für A mit all ihren Zwischengliedern als *Look-Up-Table* (LUT) im Speicher zu halten und darauf zurückzugreifen, bis der komplette Analysevorgang für eine Expressionstudie, d.h. alle statistischen Tests dieses Typs, abgeschlossen sind.

Für eine Darstellung des Algorithmus ist die äquivalente Formulierung

$$A(u,m,n) = A(u|N,m), N = m+n$$

von A nützlich.

Aus dem im vorherigen Abschnitt erwähnten Randbedingungen und Abhängigkeiten lässt sich ein kompakter Algorithmus formulieren, der eine LUT LT_A zur Geschwindigkeiststeigerung verwendet:

Algorithmus 1 : $A(u,m,n)$

Data : $u,m,n \in \mathbb{N}_0; LT_A[\frac{1}{2}nm+1,n,m]$
Result : Anzahl aller möglichen Anordnungen $A(u,m,n)$

```
1  if u < 0 then return [0]            // laut Definition A(u < 0, m, n) = 0
2  if u = 0 then return [1]            // nur eine Anordnung mit x_i < y_i, ∀i, j
3  if u > mn then return [(m+n choose m)]   // A(u > nm, m, n) = (m+n choose m)
4  if m > n then                       // Austausch nach Formel 5.17, so dass m < n
5      m ⇤⋈⇥ n
6  if u < n then                       // Ist u < n, sind nur die ersten y_i mit 1 ≤ i ≤ u
   für
7      n ⇤ u                           // die Auszählung relevant: A(u, m, n) = A(u, m, u)
8      if u < m then                   // Gilt zusätzlich u < m, so
9          m ⇤ u                       // folgt A(u, m, n) = A(u, u, u)
10     else
           // → fortsetzen mit A(u|m, u) ∧ m ≤ (u == n)
11 if u > ½mn then   // u > mn/2 : A(u|m, n) = (m+n choose m) − A(mn − u − 1|n, m)
12     return [(m+n choose n) − A(mn − u − 1, n, m)]   // vgl. Formel 5.25
13 if m = 1 ∨ n = 1 then            // Randbed.: A(u, 1, n) = A(u, m, 1) = u + 1
14     return [u + 1]                                  // vgl. Formel 5.23
15 if LT_A[u, m, n] ≠ 0 then        // Ist der Wert bereits tabelliert?
16     return [LT_A[u, m, n]]
17 else                             // Nein → nach Formel 5.18 berechnen und setzen
18     LT_A[u, m, n] ⇤ A(u, m, n − 1) + A(u − n, m − 1, n)
19     return [LT_A[u, m, n]]
```

In den Kommentaren zu jeder einzelnen Zeile von Algorithmus 1 findet sich eine kurze Erklärung und der Verweis auf die zu Grunde liegenden Formeln bzw. Zusammenhänge, die zumeist in den vorangehenden Kapiteln abgeleitet wurden. Lediglich der `if`-Block ab Zeile 6 basiert auf einem noch nicht dargelegten, für den Algorithmus wichtigen Zusammenhang, der Speicherplatz spart und die Laufzeit verbessert.

Betrachtet man A für ein u mit $u < n$, wobei angenommen wird $m < n$, so spielen für die Berechnung nur u und y_i eine Rolle, d.h. $1 \leq i \leq u$. Dies wird bei

der Visualisierung des Extremfalles von Formel 5.3 für die geordneten Stichproben X und Y in Form einer Matrix schnell klar:

X	Y			
	y_1	y_2	$\dots$	y_n
x_1	0	0	$\dots$	0
x_2	0	0	$\dots$	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_{m-1}	0	0	$\dots$	0
x_m	1	1	$\dots$	1

Der extremste Fall für das Auftreten von u ist der, in dem das größte Element von X, d.h. x_m, größer ist als alle $y_i - u$ Einsen in der letzten Spalte. Dies bedeutet jedoch im Umkehrschluss, dass für ein bestimmtes u auch nur die ersten u Elemente von Y betrachtet werden müssen. Analoge Überlegungen für den anderen extremen Fall, dass alle x_i größer sind als y_1 aber kleiner als alle weiteren $y_i - u$ Einsen in der ersten Spalte führen auf den `if`-Block in Zeile 8 des Algorithmus 1.

m	n	$u =$ 0	1	2	3	4	5	6	7	LT_A kompakt		
1	1	⊢— Formel 5.23 → 1								⊢—		
1	2	1	2	⊢—						⊢—		
1	3	1	⊢—							⊢—		
1	4	⊢—								⊢—		
1	5	⊢—								⊢—		
2	1	⊢— Formel 5.17 → (1, 2)								⊢—		
2	2	1	2	[4]	⊢—					4	⊢—	
2	3	1	2	4	[6]	⊢—				6	⊢—	
2	4	1	2	4	⊢—					⊢—		
2	5	1	2	4	⊢—					⊢—		
3	1	⊢— Formel 5.17 → (1, 3)								⊢—		
3	2	⊢— Formel 5.17 → (2, 3)								⊢—		
3	3	1	2	4	[7]	[10]	⊢—			7	10	⊢—
3	4	1	2	4	7	[11]	[15]	[20]	⊢—	11	15	20
3	5	1	2	4	7	11	[16]	[22]	[28]	16	22	28

Tabelle 5.6: Darstellung des Inhaltes der LUT LT_A beim Ablauf des Algorithmus 1 für die Berechnung von $A(u, 3, 5)$. Die blau markierten Einträge sind die, die eigentlich Information tragen und in einer LUT unbedingt tabelliert werden müssen. Auf der rechten Seite der Tabelle ist die kompakte Darstellung von LT_A zu sehen, die unter Zuhilfenahme geeigneter Zugriffsfunktionen speicherschonender für Algorithmus 1 benutzt werden kann.

Die im Algorithmus 1 verwendete LUT LT_A weist mit ihren Dimensionen von $\left(\frac{1}{2}nm + 1\right) \times n \times m$ den einfachsten Ansatz, aber auch den größten Speicherver-

brauch auf. Für einen Eindruck der LUT-Nutzung durch den dargestellten Algorithmus ist die Betrachtung eines überschaubaren Beispiels hilfreich. Tabelle 5.6 beschreibt die Einträge für die Berechnung der $A(u, 3, 5)$ für alle u.

Im rechten Teil der Tabelle ist die Form von LT_A gezeigt, auf die, im Sinne einer möglichst speichersparenden Datenstruktur, die verwendete LUT verdichtet werden muss. Zu berücksichtigen ist bei diesem Entwurf auch die Zielsprache Java und damit verbunden, das in Abschnitt 5.3.2 in Bezug auf den Speicherbedarf von Java-Datenstrukturen Gesagte. Das heißt, die entworfene logische Struktur von LT_A sollte möglichst in einer `f-` oder einer `c.f`-analogen Form verwaltbar sein.

Ähnlich dem Aufbau der Tabelle 5.6 wäre eine Organisation als lineares Feld von Feldern günstig. Für die Zahl an Einträgen in A_{tabled} ergibt sich damit:

$$S_{tab} = \sum_{i=2}^{m} (n + 1 - i) = \frac{1}{2}(m - 1)(2n - m) \tag{5.26}$$

Für die Feldbreite S_u einer Speicherzelle A_{mn} von LT_A an der Stelle $[m, n]$ liefert eine genauere Betrachtung von Tabelle 5.6 und dem `if`-Block ab Zeile 6:

$$S_u = \left\lfloor \frac{mn}{2} \right\rfloor + 1 - n = n \left(\left\lfloor \frac{m}{2} \right\rfloor - 1 \right) + 1 \tag{5.27}$$

Zu beachten ist, dass aus dem Algorithmus 1 heraus bereits die Randbedingung $1 < m \leq n$ erfüllt ist.

Für das Lesen und Schreiben der Einträge von A_{tabled} muss aus den konkreten Indexwerten i, k der Index $i_{A_{tabled}}$ für das jeweilige Feld bestimmt werden. Als Zugriffsfunktion $f(i, k)$ erhält man für die Berechnung des zugehörigen Indexes:

$$\begin{aligned} i_{A_{tabled}} = f(i, k) &= \sum_{j=2}^{i-1} (n + 1 - j) + (k - i) \\ &= \left[(n + 1)(i - 2) - \frac{1}{2}(i - 2)(i + 1) \right] + (k - i) \\ &= \left[n - \frac{1}{2}(i - 1) \right] (i - 2) + k - i \end{aligned} \tag{5.28}$$

Fasst man diese Operationen zu einer gemeinsammen Struktur zusammen, so ergibt sich der in Algorithmus 2 definierte *Abstrakte Datentyp* (ADT), der in seinen Zugriffsoperationen, seinem Definitions- und Wertebereich genau zur Berechnungsvorschrift von Algorithmus 1 passt.

Einpassung der WMW-Klassen zur Berechnung von $A(u|N, m)$ in Apache Commons-Math

Um einen groben Gesamtüberblick über die Struktur von Apache Commons-Math zu geben, ist in Abbildung 5.5 das Package-Diagramm dargestellt (Alle nachfolgenden Diagramme entsprechen den üblichen Regeln der *Unified Modeling Language*

Algorithmus 2 : $LT_A(u, m, n)$

Data : $n_{max}, m_{max} \in \mathbb{N}_0$, Struktur `Entry` als indiziertes, eindimensionales Feld ganzer Zahlen

Result : Struktur LT_A als LUT für Algorithmus 1

```
1  Initialisierung                    // Aufruf mit den maximal möglichen Zahlen
   m_max, n_max
2  begin
3      S_tab ⇐ 1/2 (m_max − 1)(2n_max − m_max)                  // Formel 5.26
4      A_tabled ⇐ S_tab × [↦ Entry[]]          // Leeres Zeigerfeld anlegen
5  end

6  getIndex(i,k)              // Indexarithmetik für den Zugriff auf A_tabled
7  begin
8      i_{A_tabled} = [n_max − 1/2 (i − 1)] (i − 2) + k − i          // Formel 5.28
9      return [i_{A_tabled}]
10 end

11 getA(u,m,n)                          // Lesender Zugriff auf A_tabled
12 begin
13     i_{A_tabled} ⇐ getIndex(m,n)
14     A_mn ⇐ A_tabled[i_{A_tabled}]
15     if A_mn = [↦ Entry[]] then
16         S_u ⇐ ⌊1/2 mn⌋ − n + 1                                // Formel 5.27
17         A_mn ⇐ [↦ Entry[S_u]]
18     end
19     return [A_mn[u − n]]
20 end

21 setA(u,m,n,val)                      // Schreibender Zugriff auf A_tabled
22 begin
23     i_{A_tabled} ⇐ getIndex(m,n)
24     A_mn ⇐ A_tabled[i_{A_tabled}]
25     if A_mn = [↦ Entry[]] then
26         S_u ⇐ ⌊1/2 mn⌋ − n + 1                                // Formel 5.27
27         A_mn ⇐ [↦ Entry[S_u]]
28     end
29     A_mn[u − n] ⇐ val
30 end
```

(UML)-Syntax [Bal04, BRJ99].). Sehr gut ist die Zuordnung verschiedener Packages und der in ihnen enthaltenen Subpackages zu bestimmten Gebieten der Mathematik zu erkennen.

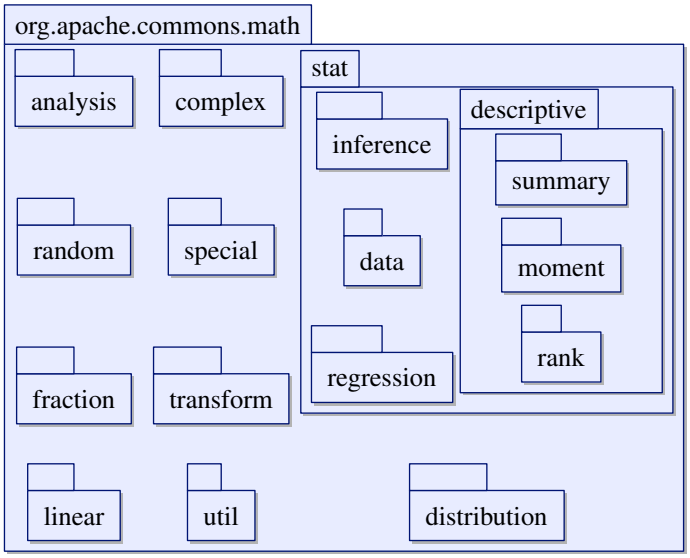


Abbildung 5.5: Das komplette Package-Diagramm von Apache Commons-Math, gibt einen groben Überblick über die logische Organisation der Bibliothek. In der Abbildung sind auch die Subpackagestrukturen erkennbar.

Für die Implementierung der WMW-Verteilung ist das `distribution`-Package die logische Zuordnungseinheit. Etwas diskussionsfähig ist, dass dieses Package, sowie auch das `rand`-Package, nicht im `stat`-Package zu finden sind. Möglicherweise zogen die Entwickler ein zukünftiges Refactoring [Fow05] mit einem Package für die Wahrscheinlichkeitstheorie in Betracht.

Für die in Abschnitt 5.5.1 als momentan vorhanden erwähnten Verteilungsfunktionen existiert im `distribution`-Package bereits eine gestaffelte Klassenhierarchie. Abbildung 5.6 stellt alle beteiligten notwendigen Klassen, Schnittstellen und ihre Beziehungen untereinander in einem Klassendiagramm dar. Zu erkennen ist die zunehmende Spezialisierung sowohl der Schnittstellen, wie auch der eigentlichen Klassen von Eigenschaften der Verteilungen ganz allgemein hin zu kontinuierlichen und diskreten Verteilungsarten. Für jede Verteilungsfunktion muss eine eigene Schnittstelle ergänzt werden, z.B. *`TDistribution`* bzw. *`WMWDistribution`*, die die spezifischen Eigenschaften der jeweiligen Verteilung, wie beispielsweise Art und Umfang der Parameter, den generelleren Schnittstellen *`ContinuousDistribution`* und *`IntegerDistribution`* hinzufügt.

Realisiert werden die Verteilungen dann schließlich in den Implementierungsklassen, wie beispielsweise `TDistributionImpl` und `WMWDistributionImpl`, die die vorher eingeführten Schnittstellen realisieren und von den abstrakten Klassen *`AbstractContinuousDistribution`* bzw. *`AbstractIntegerDistribution`* erben.

Abbildung 5.7 stellt das Klassendiagramm der Implementierungsklassen für die

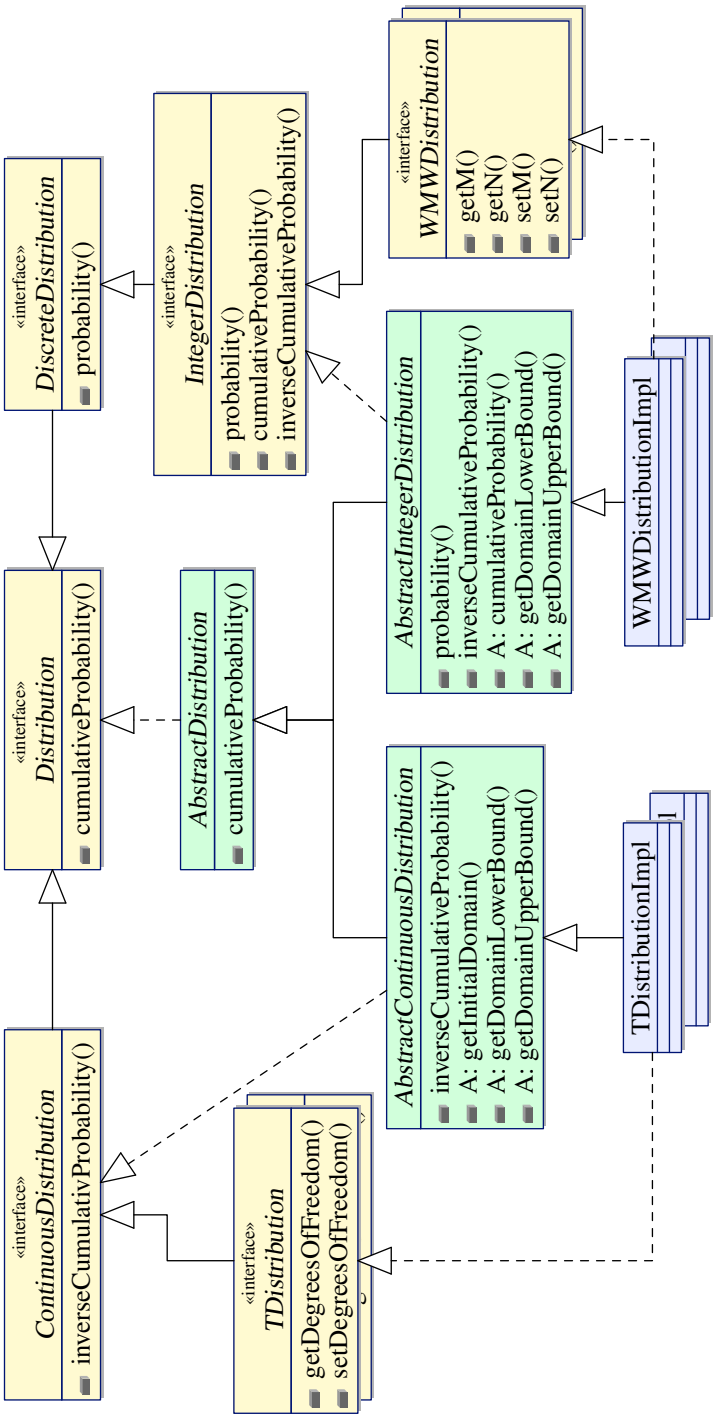


Abbildung 5.6: Klassendiagramm der Schnittstellen, Klassen und Beziehungen, die an der unmittelbaren Implementierung der Verteilungsfunktionen in Apache Commons-Math beteiligt sind. Schnittstellen sind gelb und abstrakte Klassen grün hervorgehoben. Abstrakte Methoden sind mit einem vorgestellten `A` gekennzeichnet. Als Beispiel für die Realisierungsklassen auf der untersten Ebene wurde die kontinuierliche t-Verteilung und die diskrete WMW-Verteilung gewählt.

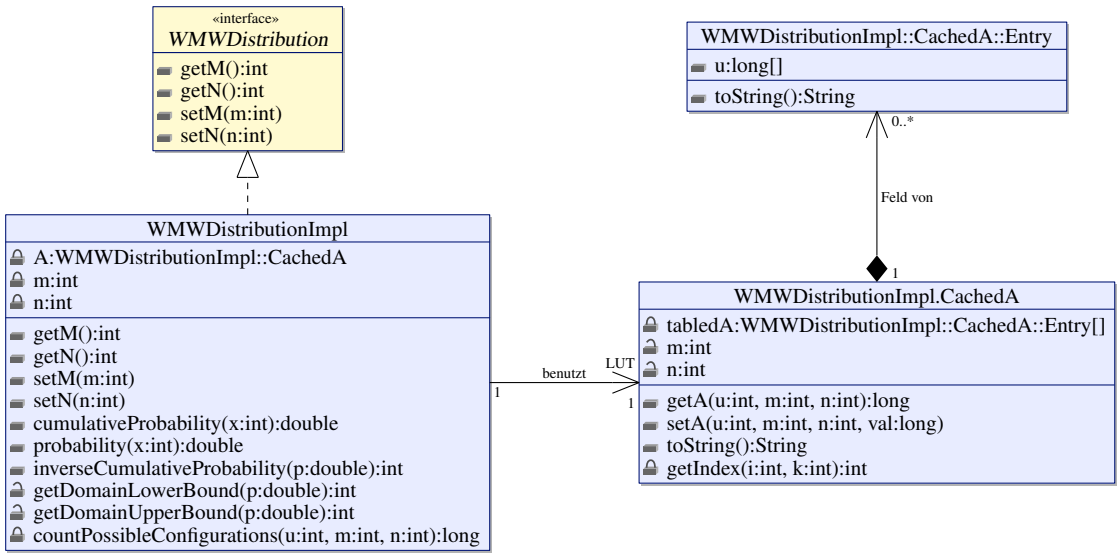


Abbildung 5.7: Klassendiagramm der Klassen, die an der Implementierung der WMW-Verteilung beteiligt sind und ihrer Beziehungen. Da die Hilfsklassen `CachedA` und `Entry` so speziell und eng an den Kontext der Klasse `WMWDistributionImpl` gekoppelt sind, erfolgte ihre Realisierung als innere Klasse von `WMWDistributionImpl`.

WMW-Verteilung dar. Der Algorithmus 1 ist im Wesentlichen in der privaten Methode `countPossibleConfigurations` realisiert, die intern von `WMWDistribution-Impl` zur Berechnung von $A(u|N, m)$ benutzt wird. Diese Methode greift auf die auf Seite 111 algorithmisch beschriebene, spezialisierte LUT zurück, die in der Klasse `CachedA` implementiert ist. Einträge in `CachedA` werden nur angelegt, wenn sich tatsächlich ein Rückgriff auf die entsprechenden Felder als notwendig erweist.

Die `toString`-Methoden der beiden Klassen `CachedA` und `Entry` dienen als Logging-Funktionen einer übersichtlichen Ausgabe des Inhaltes der LUT, vor allem zu Zwecken des Debuggings.

Die Klasse `WMWDistributionTest`, die Commons-Math für die WMW-Implementierung hinzugefügt werden musste, erweitert die Testklassen des Packages `distribution`. Abbildung 5.8 stellt das Klassendiagramm der verteilungsbezogenen Testklassen dar.

Zusätzlich zu den bereits dargestellten Ergänzungen, ist die Implementierung der WMW-Verteilung noch in den Erzeugungsmechnanismus des `distribution`-Packages zu integrieren. Dies wird in Commons-Math für alle Verteilungen über ein Factory-Pattern gelöst (siehe Seiten 18ff, 25ff in [ES04] bzw. 107ff, 131ff in [GHJV96]).

Für die Erzeugung von `WMWDistributionImpl`-Objekten sind die beiden Factory-Klassen `DistributionFactory` und `DistributionFactoryImpl` entsprechend durch eine Methode `createWMWDistribution` zu ergänzen.

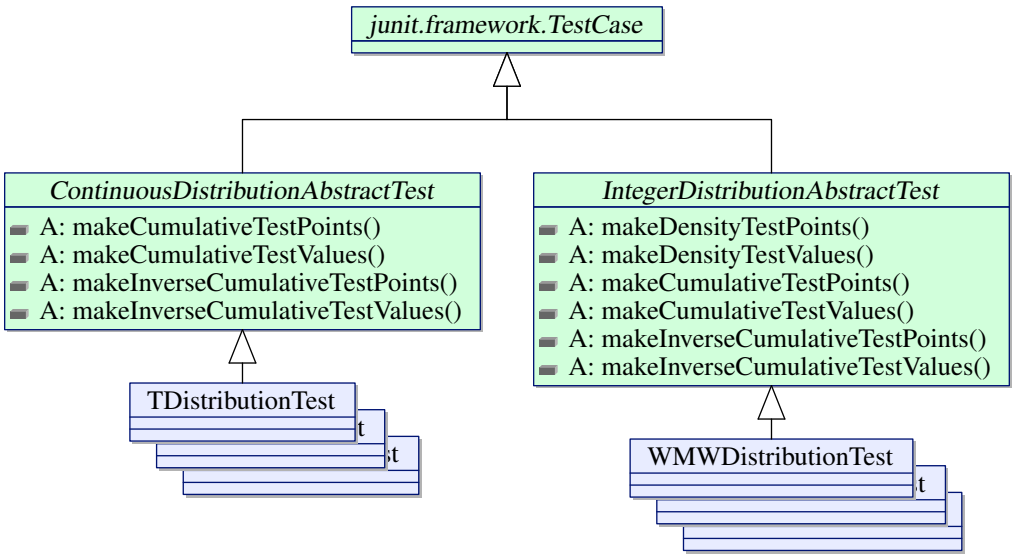


Abbildung 5.8: Klassendiagramm der verteilgungsbezogenen Testklassen im Package `distribution`. Ausgangspunkt ist das in Java übliche Testframework `junit.framework`. Auch hier sind die abstrakten Klassen grün hervorgehoben und abstrakte Methoden durch ein vorgestelltes `A` gekennzeichnet. Für die beiden abstrakten Testklassen wurden nur die enthaltenen abstrakten Methoden aufgeführt. Diese müssen dann von realisierenden Testklassen der Verteilungen implementiert werden.

5.6 Ausblick im Zusammenhang mit aktuellen Entwicklungen

Die Realisierung der WMW-Verteilung in Apache Commons-Math ist ein erster Ausgangspunkt für eine Implementierung des WMW-Tests in einer freien Java-Bibliothek. Damit lässt sich ein exakter WMW-Test realisieren, wenn im vorhandenen Datenmaterial keine Bindungen auftreten. Die nach Abschnitt 5.5.2 vorgenommene Implementierung von $A(u|N,m)$ ist besonders für die Situation bei der Genexpressionsanalyse geeignet - viele simultane Tests, bei denen Stichprobenwerte teilweise ausfallen (vgl. hierzu die Ausführungen auf Seite 107).

Treten zwischen den Werten der beiden Stichproben Bindungen auf, so muss entweder eine Korrektur des mittels des obigen Verfahrens errechneten Wertes vorgenommen werden (vgl. [BLS00]), oder eine exakte, auf das Vorkommen von Bindungen ausgelegte Methode angewendet werden. An dieser Stelle ist wichtig anzumerken, dass die in der Fachliteratur wie beispielsweise [SH06] tabellierten Werte für den WMW-Test im allgemeinen auch nicht für den Fall von auftretenden Bindungen gelten und deshalb vor der Anwendung korrigiert werden müssten.

Die Ergebnisse von [BLS00] zeigen, dass der Unterschied zwischen exakten und korrigierten Werten entscheidend sein kann und legen die Implementierung des exakten Verfahrens nahe. Hierzu existieren eine Reihe von Veröffentlichungen, die einen möglichen Weg weisen. Ein früh entwickeltes Verfahren wird in [Klo66] vorge-

stellt, das mit Hilfe des Netzwerk-Algorithmus wesentlich weiter verbessert werden konnte [MPT84, MPW88] und noch einmal in [CK97] eine Optimierung erfahren hat.

Der Netzwerk-Algorithmus [MP80] ist ein Verfahren, das auch für andere statistische Fragestellungen anwendbar ist und dessen Implementierung aus diesem Aspekt heraus lohnenswert wäre.

Im Hinblick auf die Auswertung von Genexpressionsdaten ist eine Implementierung weiterer nichtparametrischer rang- oder permutationsbasierter Verfahren, wie die in Abschnitt 5.4 genannten Beispiele BWS, BM und RP, erstrebenswert. So könnte auf einfachere Art und Weise eine größere Vielfalt an statistischen Methoden auf Genexpressionsdaten angewendet und getestet werden.

Eine sehr wichtige Entwicklung im Bereich der Informatik, die erst in den letzten zwei Jahren begonnen hat, ist für die Implementierung statistischer Verfahren von großem Potential. Der Paradigmenwechsel, die Leistungsfähigkeit der *Central Processing Unit* (CPU) nicht mehr durch eine Steigerung des Systemtaktes zu erhöhen, sondern durch die Integration von mehreren Recheneinheiten, sogenannten Kernen oder Cores, innerhalb einer einzigen CPU, eröffnet und erfordert neue algorithmische Ansätze für diesen Bereich.

Bisher war eine echte Ausführungsparallelität nur speziellen, teureren Rechnern für das *Symmetrische Multiprozessing* (SMP) vorbehalten. Mit der Durchsetzung der Multi-Core-Prozessoren im Massenmarkt wird der SMP-Rechner zum Standard auf jedem Arbeitsplatz. Beginnend bei Dual- und Quadcore-CPUs sind bereits lauffähige Prototypen mit bis zu 80 Einheiten und bis zu 2 Teraflop Leistung gezeigt worden [Int07].

Eine weitere Entwicklungslinie in Richtung massiv paralleler Systeme von normalen Arbeitsplatzrechnern stellen die Bemühungen der Grafikhardwarehersteller dar, die Leistungskapazität ihrer hochgradig parallel arbeitenden, sehr leistungsstarken *Grafikprozessor*s (GPU) auch als Multi-Core-Recheneinheiten für ganz allgemeine Problemstellungen zu erschließen - *General Purpose Computing on GPU* (GPGPU) (siehe [GPG07]). Hier sind die Initiativen *Complete Unified Device Architecture* (CUDA) [NVI07] und *Close To Metal* (CTM) [AMD06] der Hersteller Nvidia und ATI/AMD zu nennen. In [Dub07] finden sich Beschreibungen erster CUDA-basierter HPC-Projekte für den Life-Science-Bereich und die Bioinformatik.

Auch kommerzielle Entwickler von Statistiksoftware verstärken ihre Bemühungen in Richtung einer besseren Ausnutzung der Multi-Core-Architektur. So wurden zum Beispiel im neuen Release 16 des Statistikpaketes SPSS entsprechende Algorithmen für die lineare Regression, die Korrelation, die partielle Korrelation und die Faktorenanalyse verwendet [SPS07].

Eine ganze Reihe rechenintensiver statistischer Verfahren zur Microarrayauswertung, wie die erwähnten Permutationstests und Resampling-Methoden, sind sehr gut parallelisierbar und ihre Leistungen könnten deshalb deutlich mit der Anzahl an verfügbaren Recheneinheiten skalieren. Zieht man das Wachstum der Ausgangsdaten durch die steigende Integrationsdichte der Microarrays in Betracht, erscheint es lohnend und notwendig, Entwicklungen in dieser Richtung voranzutreiben.

6 Spezielle Methoden für die visuelle explorative Datenanalyse von Microarraydaten

> Numerical quantities focus on expected values, graphical summaries on unexpected values.
>
> *(J. W. Tukey)*

6.1 Bedeutung und Problematik der visuellen explorativen Datenanalyse

Ziel jeglicher Auswertung von Daten stellt die Extraktion und Entdeckung von in ihnen enthaltenen Strukturen und Zusammenhängen dar. Die einfachste Möglichkeit hierfür bestände darin, sie in einer geeigneten Darstellung zu visualisieren, dem Nutzer zu präsentieren und dadurch die visuellen Analysefähigkeiten des Menschen auszunutzen. Leider sind die grundlegenden Darstellungsvarianten bei den derzeit gebräuchlichsten Kommunikationsmitteln sehr begrenzt. Sobald Daten mehr als drei Dimensionen besitzen, ist eine einfache Koordinatendarstellung nur schwer möglich. Schon drei Dimensionen bereiten Schwierigkeiten, weil die Darstellungsfläche in Dokumenten, wie auch auf herkömmlichen Monitoren und Displays, zweidimensional ist und bereits hier auf eine Projektion eines höherdimensionalen Raumes auf einen niedrigdimensionalen zurückgegriffen werden muss. So formuliert E.R. Tufte in [Tuf90] im ersten Kapitel „Escaping Flatland“:

> Even though we navigate daily through a perceptual world of three spatial dimensions and reason occasionally about higher dimensional arenas with mathematical ease, the world portrayed on our information displays is caught up in the two-dimensionality world of the endless flatlands of paper and video screen. All communication between the readers of an image and the makers of an image must now take place on a two-dimensional surface. Escaping this flatland is the essential task of envisioning information - for all the interesting worlds (physical, biological, imaginery, human) that we seek to understand are inevitably and happily multivariate in nature. Not flatlands.

Gerade die interessantesten Fragestellungen zeichnen sich durch ihre multivariate Natur aus und die Suche nach geeigneten Darstellungsverfahren hat in der Statistik eine lange Tradition.

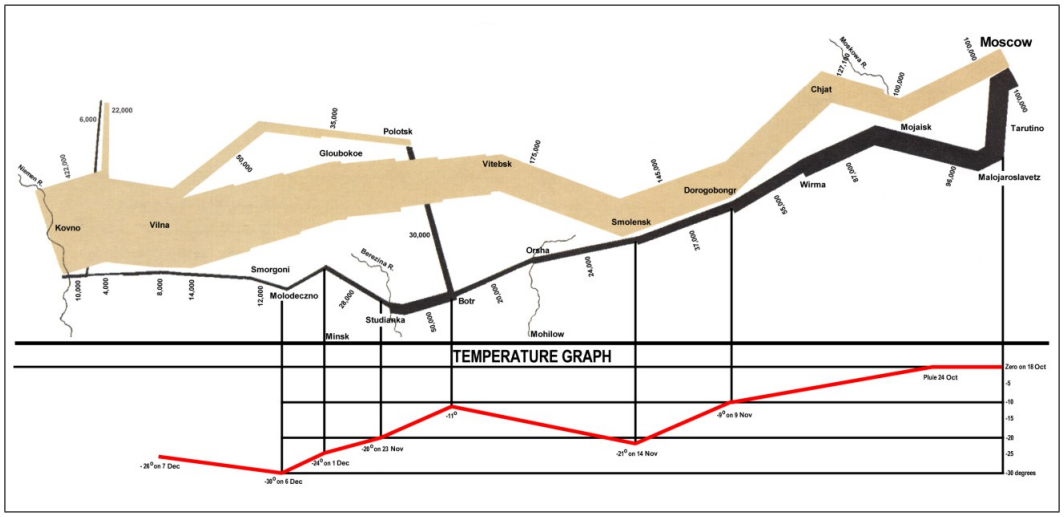


Abbildung 6.1: Schematisierte Reproduktion von Charles Minards Darstellung des Russlandfeldzuges von Napoleon 1812. Diese Darstellung gilt als ein historischer Meilenstein in der Visualisierung multivariater Daten und stellt 6 bzw. 7 Größen gleichzeitig dar (siehe Text). Die Grafik ist `http://www.napoleonic-literature.com/1812/1812.htm` entnommen.

Als einer der großen historischen Meilensteine in der Visualisierung multivariater Daten, der hier beispielhaft erwähnt werden soll, gilt Charles Minards (1781-1870) Darstellung von Napoleons Russlandfeldzug von 1812 [Tuf83, Fri99, Fri07b] (siehe Abbildung 6.1 für eine schematisierte Reproduktion des historischen Vorbildes). Tufte bezeichnet sie gar als „It may well be the best statistical graphic ever drawn“. Innerhalb dieser Grafik werden die Daten von 6 bzw. 7 verschiedenen Größen dargestellt (siehe [Wil05] für eine ausführlichere Diskussion und eine Erklärung zur 7. nicht so offensichtlichen Größe). Damit wird eine sehr hohe Informationsdichte erreicht, ohne dabei gleichzeitig den Betrachter in seiner visuellen Aufnahmefähigkeit zu überfordern.

In [Tuf83], [Tuf90] und [Fri07b] finden sich sowohl hervorragend gelungene als auch abschreckende Beispiele für die Datenvisualisierung. Zum Teil besteht bei der Visualisierung zur explorativen Datenanalyse eine erhebliche Gefahr, den Betrachter bewusst oder unbewusst in die Irre zu leiten und Effekte sowie Strukturen als dateninhärent vorzutäuschen. Hierzu finden sich viele Beispiele in den oben genannten drei Quellen.

Eine ganze Reihe von verschiedensten Darstellungsvarianten haben sich über die Zeit entwickelt, etabliert und bewährt. Zum Beispiel die folgenden - um nur einige zu nennen:

- **Scatterplot**:
  Beim Scatterplot erfolgt eine einfache Darstellung eines Symbols an der Position x und y in einer zweidimensionalen Ebene. Zusätzliche Informationen können zu den einzelnen Datenpunkten in begrenzter Weise durch die Wahl unterschiedlicher Formen, Farben oder Farbsättigungen für die Symbole dargestellt werden. Außerdem ist das Einzeichnen weiterer Größen möglich, zum Beispiel an den Legenden und Achsen.

- **Scatterplotmatrix**:
  Scatterplotmatrizen stellen eine Erweiterung des einfachen Konzepts der Scatterplotdarstellung auf mehr als zwei Dimensionen dar. Hierbei werden für n Dimensionen die $n * (n-1)$ Paare von Dimensionen jeweils als Scatterplot in Form einer Matrix dargestellt. Dabei muß man beachten, dass die Anzahl der Dimensionen 10 nicht überschreiten sollte $n \leq 10$.

- **Parallelkoordinatenplot**:
  Der *Parallelkoordinatenplot* (PKP) ist bei einer begrenzte Anzahl an Dimensionen sehr gut für eine Darstellung multivariater Daten geeignet. Deshalb wurde er auch als Ausgangspunkt für die in diesem Kapitel vorgestellten Arbeiten gewählt. Ab Abschnitt 6.3 auf Seite 123 erfolgt eine nähere Beschreibung des PKP.

- **Multidimensional-Scaling**:
  Beim Multidimensional-Scaling werden die k-dimensionalen Datenpunkte $\vec{x_i} = (x_{i1}, \ldots, x_{ik})$ auf Punkte $\vec{y_i} = (y_{i1}, \ldots, y_{il})$ innerhalb eines niedrig-dimensionaleren Raumes abgebildet $l \ll k$, wobei in Anbetracht der Visualisierbarkeit meistens $l = 2$ oder $l = 3$ gewählt wird. Die nichtlineare Abbildung der Datenpunkte erfolgt so, dass die Distanzen zwischen den Punkten im Bildraum den ursprünglichen Distanzen im Orginalraum so genau wie möglich entsprechen $\|\vec{y_i} - \vec{y_j}\| \simeq \|\vec{x_i} - \vec{x_j}\|$ [VR02]. Man kann sich dabei die Bildpunkte $\vec{y_i}$ als die Projektion der Datenpunkte $\vec{x_i}$ auf eine elastisch verformbare Ebene vorstellen, die so zwischen die Datenpunkte gelegt wird, dass die Abstände zwischen den $\vec{x_i}$ erhalten bleiben.

- **Principal Components Analysis (PCA)**:
  Bei der PCA wird im Gegensatz zum Multidimensional-Scaling [VR02, Pea01] eine starre Projektionsebene bzw. ein Projektionsraum so im Raum der Datenpunkte $\vec{x_i}$ rotiert, dass deren Projektionen $\vec{y_i}$ auf diese Ebene die beste Approximation der ursprünglichen Datenpunkte ergeben. Auch hierbei wird im Hinblick auf eine Visualisierung meistens ein zweidimensionaler oder maximal dreidimensionaler Projektionsraum gewählt.

- **Andrews-Plot**:
  Der Andrews-Plot stellt in seiner ursprünglich vorgeschlagenen Form [And72] eine Fourier-Interpolation des k-dimensionalen Datenvektors $\vec{x} = (x_1, \cdots, x_k)$

mit der Funktion

$$f_x(t) = \frac{1}{\sqrt{2}} x_1 + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \cdots \quad (6.1)$$

dar, die im Intervall $t \in (-\pi, \pi)$ gezeichnet wird. Dabei sollten die wichtigsten Komponenten, d.h. die Dimensionen, die den wichtigsten Größen entsprechen, den niedrigeren Frequenzen zugeordnet werden, weil diese den größten Einfluss auf das Aussehen der Kurve haben. Den Andrews-Plot kennzeichnen Eigenschaften, wie zum Beispiel die Erhaltung des Mittelwertes, der Varianzen und der euklidischen Distanzen [And72]. Eng beieinander liegende Datenpunkte resultieren somit auch in sehr ähnlichen Funktionsverläufen, weshalb der Andrews-Plot besonders gut geeignet ist, um Clusterungen in multidimensionalen Daten zu finden.

Ein weiterer spezifischer Aspekt der Darstellung von Daten auf dem Bildschirm ist die potenziell mögliche Dynamik der Visualisierung. So lassen sich beispielsweise Daten aus ganz verschiedenen, sogar wählbaren Perspektiven darstellen oder auch die gesamte Darstellung in einer vorher festgelegten Form animieren. Eine geeignete Anwendung kann eine deutliche Verbesserung der Rezeption durch den Betrachter bewirken. Aufgrund des derzeitig noch auf statischen papiergebundenen Darstellungen ruhenden Schwerpunktes der statistischen Datenvisualisierung wird diesem Aspekt noch nicht gebührend Aufmerksamkeit gezollt. Allerdings sind die Ansätze für eine diesbezügliche Entwicklung schon zu erkennen, beispielsweise in Bereichen wie *On Line Analytical Processing* (OLAP) oder der Prozesssimulation.

6.2 Datenvisualisierung in der Bioinformatik

Aufgrund des sehr hohen Datenaufkommens in der Bioinformatik, kombiniert mit einer im Vergleich zu technischen Bereichen sehr hohen Datenvariabilität, bzw. einem sehr großen potentiellen Fehleranteil des einzelnen Datenwertes, kommt einer anwendungsbezogenen und geeigneten Darstellung ein hoher Stellenwert zu. Die Daten sind oft hochdimensional und entstammen modernen Hochdurchsatzverfahren, die immer mehr in der alltäglichen Arbeit von Biologen und Medizinern an Bedeutung gewinnen. Das betrifft neben den Genexpressionsdaten auch Sequenzdaten der Genomik und ihre Anwendung in der Phylogenie oder Proteindaten aus der Massenspektroskopie. Für viele dieser Bereiche wurden und werden geeignete bereichsspezifische Visualisierungen entwickelt. Die Notwendigkeit des Entwurfes geeigneter bereichsspezifischer Visualisierung für die Bioinformatik wurde unter anderem in [RDC+02] dargelegt.

Beispielhaft seien einige Arbeiten dazu erwähnt. Peeters et al. [PvdWFvW04] stellen eine Applikation zur interaktiven Präsentation von Sequenzdaten mit ihren zugehörigen Annotationen vor. Verschiedene Standardtechniken aus dem Gebiet der Informationsvisualisierung wurden in dieser Fallstudie implementiert und erfolgreich getestet. Etwas spezifischer auf die Verknüpfung von genomischen und

proteomischen Daten ausgerichtet ist PQuad [HSPWR04]. Anhand von farbkodierten horizontalen Linien, die die DNA repräsentieren, soll die genomische Lokation differentiell exprimierter Proteine dargestellt werden. Dendroscope [HRR+07] stellt einen interaktiven Betrachter für die visuelle Analyse großer phylogenetischer Bäume dar. Es ermöglicht unter anderem die Auswahl verschiedener Darstellungsformen von Bäumen und die Einstellung unterschiedlicher Parameter der Visualisierung. GVis [HJS+05] wurde für die visuelle Analyse großer phylogenetischer Bäume zusammen mit den genomischen Daten der Organismen entworfen. Es erlaubt durch den Einsatz verschiedener Abstraktionsebenen und damit eines variierenden *Level of Detail* (LOD) das schnelle Navigieren zwischen den verschiedenen Hierarchien bis hinunter auf die Ebene der einzelnen Organismen. GVis unterstützt somit Analysen zur Genvorhersage (*Gene-Finding*) und zur Identifikation von orthologen Genen. Der *MEtaGenome ANalyzer* (MEGAN) [HAQS07] erlaubt die interaktive visuelle Analyse umfangreicher metagenomischer Datensätze (vgl. Abschnitt 2.2 über Resequenzierungsarrays auf Seite 21 zu einer Form der Gewinnung solcher Datensätze). Der leistungsfordernde Vergleich der vorliegenden Sequenzdaten mit der Datenbank bekannter Sequenzen erfolgt in einem Vorverarbeitungsschritt auf entsprechend performanten Systemen. Die Visualisierung der dabei gewonnen Vergleichszahlen und Statistiken kann dann mit MEGAN auch auf leistungsärmeren Systemen durchgeführt werden. Die in [LLBB05] vorgestellte Anwendung zur Detektion von differentieller Proteinexpression aus LC/MS-Daten (*Liquid-Chromatography* (LC)/*Mass-Spectrometry* (MS)) orientiert sich mehr am Ergebnis des Messvorganges. Farbkodierte Höhenfelder in zwei und drei Dimensionen visualisieren das m/z-Verhältnis (Masse pro Ladung) der Fragmente (aus der MS) und die Zeit (aus der LC).

Auf dem Gebiet der Genexpressionsanalyse als einem statistiklastigen Bereich kann auf den erarbeiteten Fundus an Erfahrungen zur statistischen Visualisierung zurückgegriffen und bereits bewährte Verfahren adaptiert werden. Saraya et al. [SND05] untersuchten drei freie und zwei kommerzielle Microarray-Visualisierungstools (Cluster/Treeview [ESBB98], TimeSearcher [HBMS03], Hierarchical Clustering Explorer (*HCE*) [SS02] und Spotfire [Spo08b] und GeneSpring [Agi07]) im Hinblick auf ihre Anwendungstauglichkeit. Dabei wurden die in der Tabelle 6.1 aufgeführten Visualisierungs- und Interaktionsverfahren der einzelnen Anwendungen zur Microarrayanalyse an Datensätzen von 170 bis 1060 Genen (Datenpunkte) unter 3 bis maximal 90 Konditionen (Dimensionen) evaluiert.

Die Autoren dieser Anwendungsstudie hoben vor allem zwei Ergebnisse für den Bereich der Visualisierung von Genexpressionsdaten hervor:

- Anwendungen und Visualisierungen, die für einen bestimmten Kontext entworfen wurden, sind für andere Anwendungen oft nur von begrenztem Nutzen.
- Interaktionsfähige Visualisierungen sind für das erfolgreiche Gewinnen von Wissen aus den vorhandenen Daten von sehr hoher Wichtigkeit.

Es finden sich in ihrere Studie traditionelle aus der Statistik bekannte Verfah-

Tool	Visualisierungen	Interaktion
Cluster/Treeview	Heatmaps, Cluster-Dendrogramme	F+C
TimeSearcher	Parallelkoordinaten, Graphen	Brushing, F+C, DQ
HCE	Parallelkoordinaten, Heatmaps, Scatterplots, Histogramme, Cluster-Dendrogramme	Brushing, Zooming, F+C, DQ
Spotfire 7.2 Functional Genomics	Parallelkoordinaten, Heatmaps, Scatterplots (2D/3D), Histogramme, Balken-/Tortendiagramme, Dendrogramme, Spreadsheet-Übersichten, Clustering	Brushing, Zooming, F+C, DQ
GeneSpring 5.0	Parallelkoordinaten, Heatmaps, Scatterplots (2D/3D), Histogramme, Balkendiagramme, Physical-Position-Views, Array-Layout-Views, Pathway-Übersichten, Spreadsheet-Übersichten, Clustering, Gen zu Gen Vergleiche	Brushing, Zooming

Tabelle 6.1: Kurzübersicht über die Visualisierungs- und Interaktionsmöglichkeiten gängiger Microarray-Tools, entsprechend üblichen Techniken wie z.B. *Focus+Context* (F+C) und DQ. (entnommen [SND05] und angepaßt)

ren, wie beispielsweise (2D/3D) Scatterplots, Heatmaps, Scatterplotmatrizen, Histogramme, Dendrogramme und Profilplots, sowie auch eher spezialisierte Varianten wie zum Beispiel die Pathway-Übersichten, Physical-Position-Views und Array-Layout-Views.

Auch die im Teil II auf Seite 29 vorgestellte eigene Plattform Mayday [DGN06] implementiert aufgrund ihrer Wichtigkeit eine ganze Reihe der Standardvisualisierungsverfahren, wie auch speziell entwickelte, z.B. die Enhanced Heatmap [GDN05].

6.3 Parallelkoordinaten in der Bioinformatik

*Parallelkoordinatenplot*s (PKP) [Ins85] sind ein weit verbreitetes Verfahren, um multidimensionale Daten darzustellen. PKPs bieten eine sehr gute Möglichkeit, die innere Struktur multivariater Daten zu visualisieren, zu erkennen und zu analysieren.

In der biomedizinischen Anwendung wären hier beispielhaft das WAVE-System zu nennen, das Scatterplots in Verbindung mit PKPs zur Visualisierung einer 4D-Herzsimulation benutzt [GRW+00] und SignatureSpace [PBM05], in dem Parallelkoordinaten dazu verwendet werden, Volumendatensätzen eine Ausgangsklassifikation zuzuweisen.

Der auf dem Gebiet der Genexpressionsanalyse weit verbreitete Profilplot stellt eine einfache Form des PKPs dar. Er ist als ein Standardplot in den meisten Anwendungspaketen für die umfassende Genexpressionsanalyse implementiert, wie beispielsweise in GeneSpring [Agi07] und Mayday [DGN06].

Eine sehr interessante Anwendung auf dem Gebiet der Genexpression wird in Rübel et al. [RWK+06] vorgestellt. Im Rahmen des „Berkeley Drosophila Transcription Network Project“ erfolgte die Entwicklung einer Plattform für die quantitative dreidimensionale Analyse von Genexpressionsmustern früher Embryonalstadien von Drosophila. Dabei werden die Daten bis hinunter auf die zelluläre Ebene aufgelöst und dargestellt. Die Anwendung des PKP erfolgt hier zur Darstellung der Expressionslevels einiger ausgewählter Gene je Zelle. Eine Zelle wird von einem Linienzug im PKP repräsentiert. Der Expressionslevel der verschiedenen Gene ist auf den einzelnen Dimensionen abgetragen. Einige Erweiterungen und Verbesserungen der Darstellung wurden verwendet, um die Wahrnehmung interessanter Strukturen in der großen Menge an Daten zu erleichtern, beispielsweise wurde der PKP auf eine dreidimensionale Version erweitert.

Eine wesentliche Einschränkung von PKPs stellt jedoch das *Overdraw-Problem* dar. Soll insbesondere eine große Menge an Datenpunkten (Linienzügen) dargestellt werden, so verdecken sich diese gegenseitig und verhindern auf diese Art das Erkennen der Muster einer inneren Struktur. Ein weithin genutzter Ansatz zur Lösung dieses Problems besteht in einer geclusterten Darstellung der Daten im PKP. Johansson et al. [JLJC05] kombinierten die Clusterung mit Texturen und Transferfunktionen F_T, die verschiedene Aspekte eines Datenwertes x auf seine Opazität α abbilden, $F_T(x) = \alpha(x)$, um ein gegenseitiges Verdecken zu verhindern. Fanea et al. [FCI05] schlagen eine interessante interaktive Kombination aus Parallelkoordinaten mit Starglyphen vor, um die jeweiligen Beschränkungen beider Techniken zu minimieren und ihre Stärken zu verbinden. Die vorgestellten *Parallelglyphen* kombinieren das schnelle Erfassen von Werten, das Starglyphen ermöglichen, mit der detaillierten Darstellung von Parallelkoordinaten.

6.4 Erweiterte Parallelkoordinaten für die Bioinformatik - SpRay

Wie bereits im obigen Abschnitt 6.3 erwähnt, erfolgt die Anwendung des PKPs auf dem Gebiet der Genexpression hauptsächlich als Profilplot, d.h. es werden die Expressionswerte der einzelnen Gene entlang der Konditionen unter denen diese Werte gewonnen wurden, beispielsweise unterschiedliche Zelltypen oder der gleiche Zelltyp bei verschiedenen physiologischen Bedingungen, abgebildet. Somit entspricht ein Linienzug jeweils einem Gen und die Konditionen den Dimensionen. Hierbei machen sich die oben genannten Probleme des normalen Profilplots bzw. des einfachen PKPs recht schnell bemerkbar, wie z.B. das Auftreten eines Overdraw-Problems schon bei Genexpressionsdatensätzen von lediglich geringem bis mittlerem Umfang. Dies liegt einerseits daran, dass bereits Genexpressionsdatensätze dieser Größenordnung zu umfangreich für eine derartige Darstellung sind und andererseits daran, dass die zu entdeckenden Effekte oft von kleiner Magnitude sind und im Bereich des „Rauschens“ der uninteressanten Profile liegen. Zur besseren Verdeutlichung der Vorteile der vorgeschlagenen Erweiterungen des PKPs erfolgt die Darstellung eines künstlich erzeugten Datensatzes in SpRay, eine Anwendung in der die erweiterte Form der Parallelkoordinatendarstellung zu Demonstrationszwecken implementiert wurde [DHNB06]. Die künstlich erzeugten Daten simulieren die rauschbehaftete Messung der Genexpressionswerte y_{kji} verschiedener Zellzyklus-assoziierter Gene zu 21 verschiedenen Zeitpunkten t_i $(i = 1, \ldots, 21)$ (siehe auch Tabelle 6.2). Grundlage des Datensatzes bilden k, $k = 1, \ldots, 3$, verschieden große Cluster mit dem jeweiligen Umfang $j = 1, \ldots, n_k$ von sinusförmigen Schwingungen mit unterschiedlichen additiven Konstanten C_k. Die Variationen der Phasenlage $\Delta\varphi_{kj}$ und der Amplitude Δa_{kj}, wie auch der Meßfehler ε_{kji} der einzelnen Expressionswerte, folgen einer Normalverteilung $\mathcal{N}(\mu, \sigma^2)$ mit jeweils vorgegebenen Parametern für μ

Datensatz	Datenpunkte insgesamt	Dimensionen	relevante Datenpunkte
Künstl. erzeugter Datensatz	2850	21+0	400 + 450

Tabelle 6.2: Überblick über die wesentlichen Eckdaten des künstlich erzeugten Datensatzes. Es ist die Anzahl der insgesamt enthaltenen Datenpunkte, die Anzahl der Dimensionen (Originaldatenraum + statistisch abgeleiteter Parameterraum) und die Datenpunkte der zwei Cluster, die als Ergebnis einer Analyse extrahiert werden sollten (relevante Datenpunkte) angegeben.

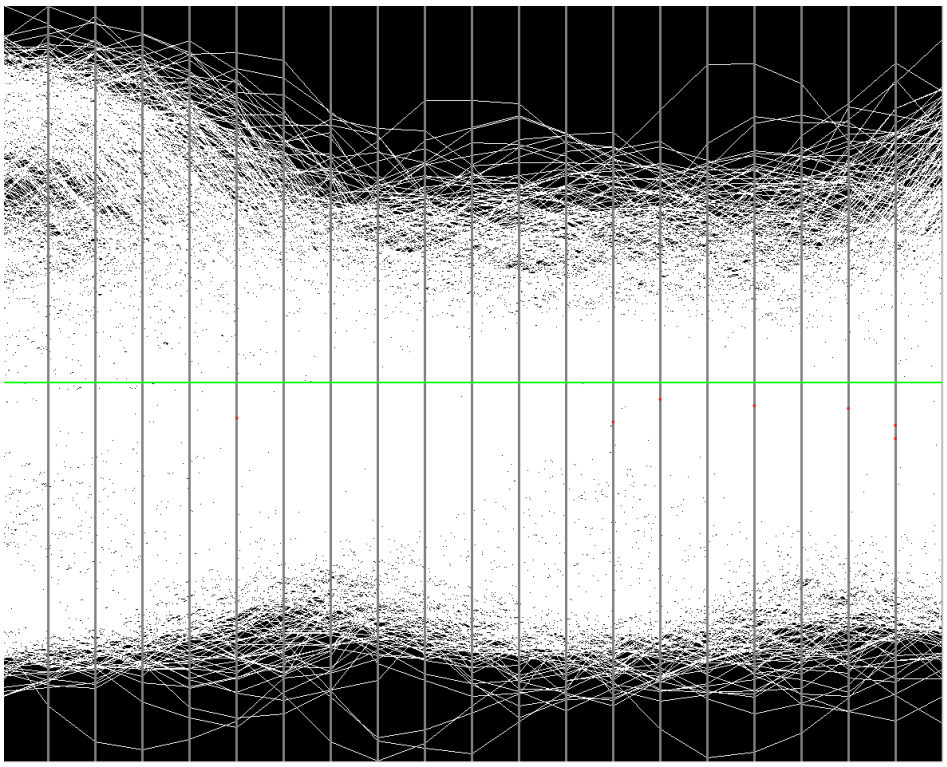


Abbildung 6.2: Einfache Anwendung des PKPs auf einen Datensatz mittleren Umfangs, der nach dem Modell von Gleichung 6.2 erstellt wurde. Diese Anwendung entspricht der Form des weitverbreiteten Profilplots und zeigt sehr deutlich ein Overdraw-Problem.

und σ^2. Damit ergibt sich als Modell für die Daten:

$$\begin{aligned} y_{kji} &= (a_k + \Delta a_{kj})\, sin\left(\frac{2\pi}{T} t_i + \{\varphi_k^0 + \Delta\varphi_{kj}\}\right) + C_k + \varepsilon_{kji}, \\ \Delta a | \mu_{\Delta a}, \sigma^2_{\Delta a} &\sim \mathcal{N}(\mu_{\Delta a}, \sigma^2_{\Delta a}), \\ \Delta\varphi | \mu_{\Delta\varphi}, \sigma^2_{\Delta\varphi} &\sim \mathcal{N}(\mu_{\Delta\varphi}, \sigma^2_{\Delta\varphi}), \\ \varepsilon | \mu_\epsilon, \sigma^2_\epsilon &\sim \mathcal{N}(\mu_\epsilon, \sigma^2_\epsilon). \end{aligned} \tag{6.2}$$

In Abbildung 6.2 ist der künstliche Datensatz mit einem Umfang von 2850 Datenpunkten in einem PKP dargestellt (siehe auch Tabelle 6.2). Bereits bei dieser Größe tritt ein deutliches Overdraw-Problem auf. Die interessierenden beiden Cluster vom Umfang 400 und 450 werden nahezu vollständig von den Datenpunkten des Clusters, der eng um die Nulllage schwingt (Baseline-Cluster), maskiert und lassen sich damit visuell kaum identifizieren.

In dieser Arbeit wird, basierend auf einer Zusammenarbeit mit Dirk Bartz vom *Innovation Center for Computer Assisted Surgery* (ICCAS) der Universität Leipzig und Julian Heinrich [Hei07], die Kombination von insgesamt vier Ansätzen zur Erweiterung des PKPs vorgeschlagen, um seinen hauptsächlichen Schwächen entgegen

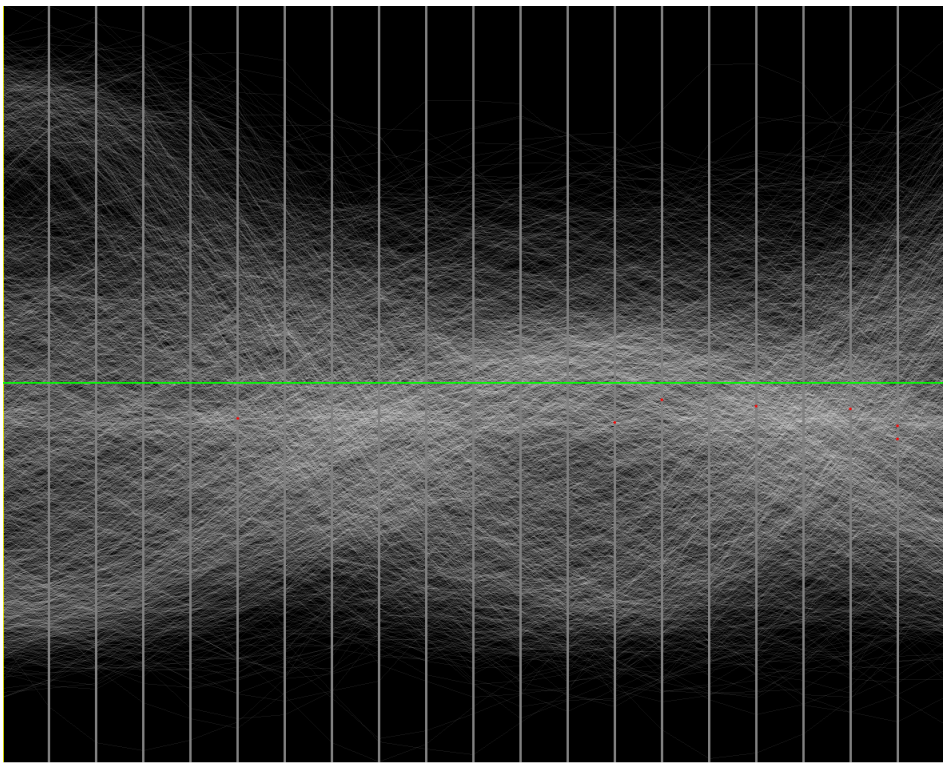


Abbildung 6.3: Allein die Erweiterung des PKPs um eine überdeckungsabhängige transparente Darstellung der Linienzüge lässt, im Gegensatz zu Abbildung 6.2, bereits einige strukturelle Details erkennen. Die Deckkraft der Farbdarstellung der Linienzüge liegt in dieser Darstellung bei ca. 7 %.

zu wirken:

1. die Variation der Transparenz der Linienzüge
2. eine Einfärbung der Linienzüge entsprechend der Anordnung der Datenelemente entlang einer wählbaren Dimension
3. die gleichzeitige und synchrone Darstellung weiterer Plots, wie z.B. Scatterplots und bzw. oder Histogrammen
4. die Erweiterung des dargestellten Raumes der ursprünglichen Daten, um einen aus diesen Daten abgeleiteten statistischen Parameterraum.

Variiert man die Transparenz der einzelnen Linienzüge, so offenbaren sich bereits erste Details der dateninhärenten Struktur entlang der einzelnen aufgetragenen Dimensionen (Abbildung 6.3). Es besteht die Möglichkeit, die Transparenz sowohl global für alle Linienzüge und damit für alle Datenpunkte zugleich zu variieren oder aber für jede Dimension, entsprechend der Dichte der Linienzüge in den einzelnen Bins (äquidistante lokale Intervalle, in die die Achse zerlegt wird) der jeweiligen Dimension bzw. Achse.

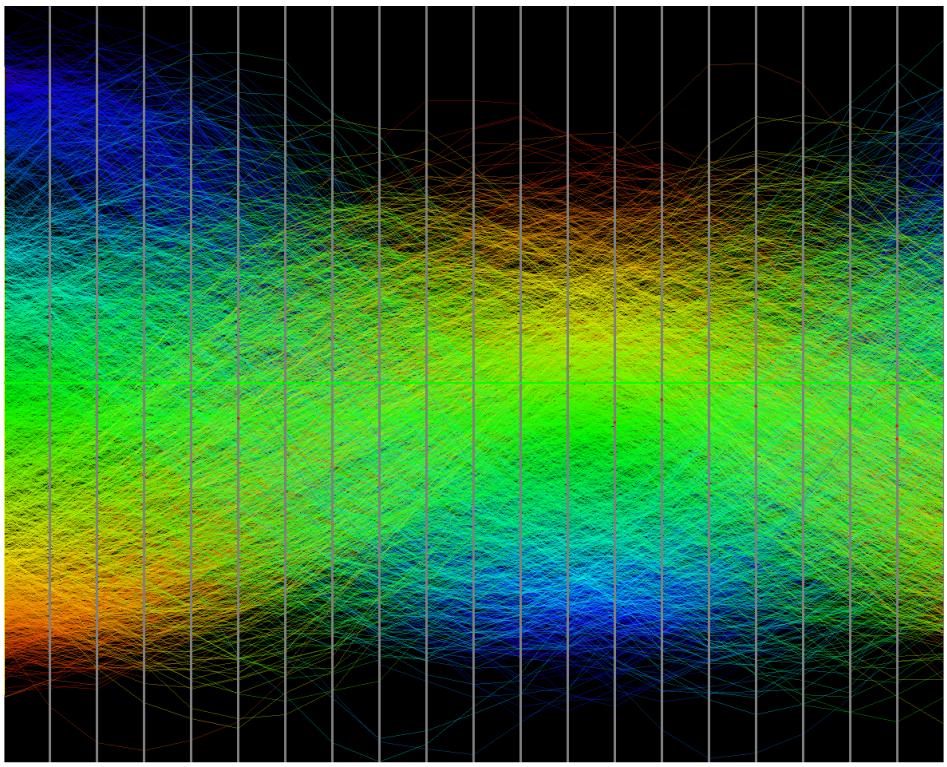


Abbildung 6.4: Die Einfärbung der Linienzüge im PKP nach einer wählbaren Dimension erleichtert zusätzlich die Wahrnehmung interner Strukturen der Daten. In der hier vorliegenden Darstellung wurden die Linienzüge anhand der ersten Dimension eingefärbt.

Kombiniert man den Einsatz der Transparenz im PKP mit einer achsenspezifischen Einfärbung, lassen sich bei einer geeigneten Wahl der Dimension, strukturelle Details noch deutlich besser visualisieren (Abbildung 6.4). Die dimensionsbezogene farbkodierte Darstellung der Linienzüge erlaubt es, die relativen Ordnungsverhältnisse innerhalb der farbgebenden Dimension in Bezug auf alle anderen Dimensionen abzubilden. Eingeschränkt wird dies allerdings durch das Overdraw-Problem des normalen PKPs und dem in den Daten enthaltenen Rauschen. Beides lässt sich aber durch das Anwenden der Transparenzdarstellung minimieren, wie in der Abbildung 6.4 zu sehen ist.

Von Vorteil ist es, dem Anwender mehrere Farbpaletten anzubieten und eine jeweils problemangepaßte von ihm wählen zu lassen. So unterstützt SpRay die weit verbreitete Rainbow-Map (spektrale Anordnung der Farbtöne von Blau über Zyan, Grün und Gelb nach Rot, siehe Abbildung 6.4), isometrischere Luminanz- und Sättigungspaletten, sowie Heat-Maps (Farbtemperatur). Die für die Wahrnehmung zu bevorzugende Grauwertpalette ist bei gleichzeitiger Nutzung der Transparenz nicht anwendbar.

Ähnlich dem Ansatz von Doleisch et al. [DGH03] in SimVis wird die visuelle Analyse von simultanen, integrierten und synchronen Sichten auf die gleichen Da-

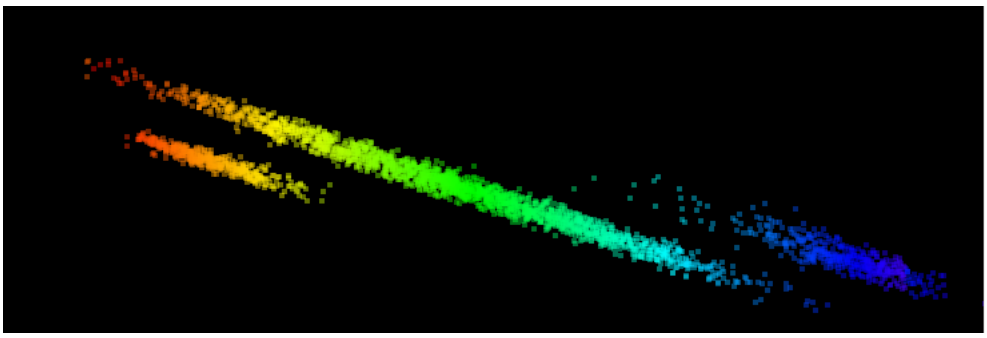


Abbildung 6.5: Assoziierter Scatterplot der 1. und 13. Dimension des PKPs von Abbildung 6.4, auch hier werden Färbung und Transparenz dargestellt (Punktgröße 4 Pixel). Bei geeigneter Wahl der Achsen sind die einzelnen Cluster gut zu identifizieren und zu trennen.

ten unterstützt. Getestet wurden insbesondere Scatterplots und Histogramme für vom Nutzer gewählte Dimensionenachsen. Dabei wurde für diese Plots die im PKP gewählte Farbkodierung benutzt. Abbildung 6.5 zeigt für den künstlich erzeugten Datensatz einen assoziierten Scatterplot zwischen der ersten und dreizehnten Dimension. Bei dieser Wahl sind die einzelnen Cluster der Daten visuell gut separiert wahrzunehmen. In einem weiteren Schritt könnten in diesem Plot einzelne Cluster ausgewählt und in der Darstellung ausgeblendet werden. Abbildung 6.6 zeigt eine nicht ganz vollständige Ausblendung des Clusters, der sehr eng um die Nulllage schwingt (Baseline-Cluster).

Im Unterschied zu den sonst üblichen Darstellungen des PKPs im Bereich der Visualisierung kommen in Microarraydatensätzen für manche Dimension bzw. Konditionen oder Experimenten invalide Daten vor. Dem sollte die Visualisierung Rechnung tragen. Um eine in dieser Hinsicht irreführende Darstellung der Linienzüge zu vermeiden, werden in SpRay die Linienabschnitte, die einen invaliden oder nicht vohandenen Wert einer bestimmten Dimension mit seinen benachbarten Dimensionen verbinden würde, nicht gezeichnet. Auf diese Weise kann ein invalider Wert nicht einen markanten und auffälligen Peak eines Linienzuges erzeugen.

Der Profilplot bleibt bei seiner Benutzung in der üblichen Weise auf die Darstellung der einzelnen Genexpressionswerte beschränkt. Ein wesentlich neuer Punkt des erweiterten PKPs ist die Kombination der einzelnen Expressionsdaten mit zusätzlich abgeleiteten statistischen Daten im PKP. Hierbei kann man sich die Eigenschaft des PKPs zunutze machen, zusammengehörige Daten in einem einzigen Linienzug darzustellen. Passt man den statistischen Parameterraum den interessierenden Fragestellungen an, so kann man damit die zugehörigen Expressionsprofile für eine genauere visuelle Exploration hervorheben. Damit ist die Wahl des statistischen Parameterraums sehr stark vom jeweiligen Anwendungsfall abhängig. Es lassen sich jedoch bestimmte Klassen von Anwendungsszenarien charakterisieren. Im Rahmen dieser Arbeit wurden drei Klassen identifiziert und für diese im nächsten Abschnitt (Abschnitt 6.5) jeweils eine zugehörige Fallstudie aus der Praxis

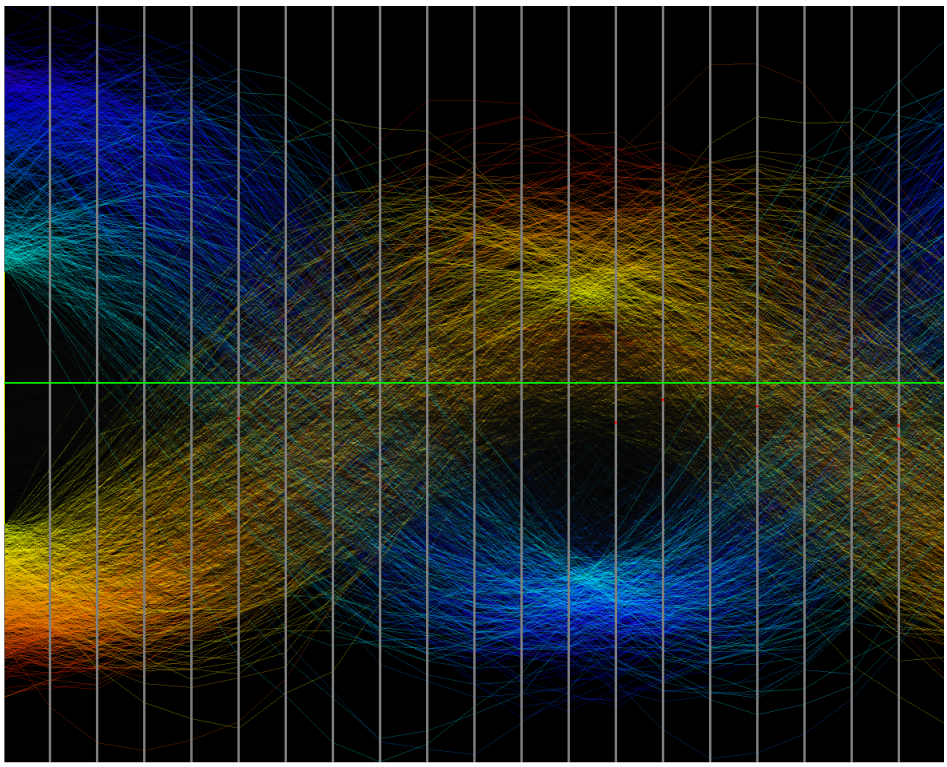


Abbildung 6.6: Darstellung der Daten im PKP nach dem Entfernen des Baseline-Clusters.

dargestellt und erläutert:

1. **Detektion zyklischer Expressionsmuster**:
   Die Detektion zyklischer Muster in Expressionsprofilen ist von großem Interesse, um sie mit den makroskopisch beobachtbaren Zyklen, denen eine Zelle, ein Gewebe oder ein ganzer Organismus unterworfen ist, in Verbindung zu bringen. Beispiele für solche zeitlichen Zyklen sind der Zellzyklus, der auch in der Fallstudie in Abschnitt 6.5.1 eine Rolle spielt oder endogene Rhythmen, wie beispielsweise der circardiane Rhythmus eines Organismus oder der Herzschlag des Herzgewebes.
   Zyklische Muster können jedoch auch räumlich betrachtet werden, wie beispielsweise die Steuerungsmechanismen bei der Morphogenese zur Ausdifferenzierung verschiedener Zellen, die Mechanismen, die der Fellzeichnung der Tiere zu Grunde liegen oder auch die Erregungsausbreitung in Nervenzellen oder dem Herzgewebe [Mur93]. Bei dieser Klasse von Fragestellung bietet sich die Kombination des Expressionsraums mit Parametern an, die die Zerlegung der Profile in ihre zyklischen Anteile (vgl. Abschnitt 6.5.1) oder ihre Darstellung in einer geeigneten zyklischen Basis ausreichend charakterisieren.

2. **Detektion gesuchter Expressionsprofile aufgrund geeigneter Signifikanzaussagen**:

Diese Klasse umfasst eine große Menge an verschiedensten Fragestellungen. Sie ist immer dann anwendbar, wenn sich spezifische Signifikanzkriterien für bestimmte Charakteristika der Expressionsprofile formulieren lassen. So besteht eine der häufigsten Zielstellungen eines Genexpressionsexperimentes darin, differentiell exprimierte Gene zwischen verschiedenen Zuständen (normal - pathologisch), Zeitpunkten oder Typen einer Zelle oder eines Gewebes zu finden. Da eine Vielzahl an Störgrößen die gemessenen Expressionswerte beeinträchtigen, ist die Anwendung statistischer Methoden zur Einschätzung der Signifikanz unumgänglich [GCH+05]. Hierbei kommt die Vielzahl an Methoden der statistischen Inferenz zum Einsatz [Tho07], wie beispielsweise parametrische und nichtparametrische Tests [Goo05, SG04, SH06]. Die Fallstudie in Abschnitt 6.5.2 stellt einen Anwendungsfall dieser Art dar und demonstriert zugleich, auf welche Weise die vorgeschlagene Visualisierung eine korrekte Auswahl des dem Problem am besten angepassten statistischen Evidenzparameters erlaubt.

Aber auch andere Arten von Evidenzen wären durchaus denkbar und bei entsprechender Wahl des Parameterraums anwendbar. So könnte zum Beispiel auch Sekundärwissen aus anderen Quellen, nicht nur numerischen, in geeignet quantifizierter Form als Dimension eingeführt werden.

3. **Qualitätssicherung anhand von Qualtitätsstatistiken**:
   Weil bei der Messung der Genexpression mittels Microarrays eine große Anzahl potentieller Fehlerquellen zu berücksichtigen sind, ist eine Qualitätssicherung unerlässlich. Hierbei kann auf verschiedenste bereits bekannte und etablierte Qualitätsmaße, wie sie beispielsweise in Form von Gewichten aus der Bildanalyse kommen [SS03, SYS03], zurückgegriffen werden oder auch auf selbst definierte, dem jeweiligen Problem angepasstere Parameter. Diese Qualitätsparameter bilden dann die Dimensionen des im erweiterten PKP verwendeten Raumes der statistischen Parameter. In Abschnitt 6.5.3 wird dies am Beispiel der Validierung eines neuen cDNA-Microarrays gezeigt.

Durch die vorgestellte visuelle Kombination statistisch gewonnener Parameter mit den Originaldaten lassen sich die perzeptiven und kognitiven Fähigkeiten des Betrachters bzw. Auswerters besser auf das Wesentliche konzentrieren. Dies ist auch eines der erklärten Ziele des erst kürzlich entstandenen neuen Visualisierungsfeldes *Visual Analytics* [Tho06]. In [NVA07] findet man die folgende kurze Definition von Visual Analytics, die ausführlicher in [Tho06] und insbesondere in [TC05] erklärt wird:

> Visual analytics is the science of analytical reasoning facilitated by interactive, visual interfaces. The goal of visual analytics is to obtain insight into massive, dynamic and often conflicting pieces and formats of information; to "detect the expected and to discover the unexpected"; and to yield timely assessments with evidence and confidence levels.

6.5 Erweiterte Parallelkoordinaten in der Genexpressionsanalyse

Um die Anwendbarkeit und Nützlichkeit der vorgeschlagenen Erweiterungen des PKPs in praxisnahen Anwendungen zu demonstrieren und die im vorangegangenen Abschnitt 6.4 identifizierten Anwendungsklassen zu belegen, wurde SpRay beispielhaft auf drei reale Microarray-Datensätze, die den jeweiligen Anwendungsklassen entsprechen, angewendet. Tabelle 6.3 gibt einen Überblick über die wesentlichsten Eckdaten der Datensätze, die visuell analysiert wurden. Die Ergebnisse selbst sollen im Folgenden näher dargestellt werden.

Datensatz	Datenpunkte insgesamt	Dimensionen	relevante Datenpunkte
Spellman Datensatz [SSZ+98]	6178	18+3	$\sim$ 800
Halbmarathon [ZZD+05]	345	8+10	13/6
Microarray Validierung	1921	6+3	17

Tabelle 6.3: Überblick über die Eckdaten der visuell analysierten realen Datensätze. Die Darstellung ist analog zu der für den künstlich erzeugten Datensatz in Tabelle 6.2. Auch hier bezeichnen die relevanten Datenpunkte die Menge der interessierenden Expressionsprofile und damit die Datenpunkte, die das Ergebnis einer Analyse darstellen sollten.

6.5.1 Zellzyklusgene von Saccharomyces cerevisiae

Ein Standarddatensatz zur Beurteilung neuer Verfahren in der Genexpressionsanalyse stellen die von Spellman et al. [SSZ+98] an der Hefe *Saccharomyces cerevisiae* erhobenen Microarraydaten dar. In einer umfangreichen Analyse der Genexpression versuchten Spellman et al., die periodischen Schwankungen der Transkriptionslevel von Genen mit dem Zellzyklus der Hefezellen in Verbindung zu bringen. Um ein verlässliches Expressionssignal sicherstellen zu können, mussten die Zellen einer Kultur erst mit einem *Arrest-Release-Zyklus* synchronisiert werden. Die Hefezellen wurden nach vier verschiedenen Methoden in einer bestimmten Zyklusphase mit Hilfe des α-Faktors, CDC15, CDC28 und per Elution festgehalten. Nach der Freigabe der Zellkulturen zum Zeitpuntk t_0 wurde an aufeinanderfolgenden Zeitpunkten mRNA entnommen und auf einem zweikanaligen cDNA-Chip mit mehr als 6000 Spots, bzw. Gensonden, hybridisiert. Im Falle des hier verwendeten α-Datensatzes handelte es sich um insgesamt 18 Zeitpunkte, die zwei komplette Perioden des Zellzyklus umfassen.

Die hybridisierten Arrays wurden per Scanner ausgelesen und mit den üblichen Methoden zur Bildanalyse, Spoterkennung, Hintergrundkorrektur, Normalisierung und Qualitätsfilterungen bearbeitet. Darauf aufbauend wurden Methoden zur Un-

tersuchung der Periodizität, der Korrelation und Clusterbilung der Expressionsprofile angewendet, um die Assoziation der Transkriptionsprofile zu bestimmten Zellzyklusphasen zu quantifizieren.

Spellman et al. identifizierten rund 800 Gene, die das definierte Minimalkriterium für eine Regulation durch den Zellzyklus erfüllten. Für eine Untermenge der gefundenen Gene konnte in anschließenden Untersuchungen weitere Belege einer Assoziation mit dem Zellzyklus gefunden werden, beispielsweise durch die Analyse von Promoterbindungsstellen oder die Variation des Transkriptionslevels durch gezielte Induktion bestimmter Zellzyklusphasen.

Aufgrund seiner Bedeutung war der Spellman Datensatz schon Gegenstand vieler Publikationen und wurde von einer ganzen Reihe von Autoren untersucht und auch kontrovers diskutiert. Shedden und Cooper [SC02] reanalysierten den kompletten Datensatz und kamen zu einem differenzierteren Schluß als Spellman et al.. Sie zeigten, dass eine Randomisierung der Daten eine viel geringere Periodizität aufweist als die Originaldaten des Experiments. Damit konnte ein Störrauschen bzw. zufällige Variationen der Messwerte als eigentliche Ursache der auftretenden Periodizität ausgeschlossen werden. Andererseits zeigen nur die α, CDC15 und CDC28 Experimente ein starkes Signal und eine gute Reproduzierbarkeit der Ergebnisse und nicht die Experimente mit Elution-Synchronisation. Shedden und Cooper schlagen für eine Erklärung der beobachteten Effekte vor, daß die Synchronisation über den α-Faktor, CDC15 und CDC28 einen weitaus dramatischerer Eingriff in den Zellhaushalt bedeutet als die Elution-Synchronisation. Die sich deutlich zeigenden zyklischen Expressionsmuster bei den ersten drei Synchronisationsmethoden seien demzufolge eher eine Stressantwort der Hefezelle als ihr natürliches Regulationsmuster bei ungestörter Entwicklung.

Für den Test des erweiterten PKPs sollen im Folgenden nur die Expressionswerte des α-Datensatzes betrachtet werden. Die Analyse basiert, ähnlich der von Shedden und Cooper [SC02], auf einer *Harmonischen Regressionsanalyse* (HRA), d.h. die Expressionswerte $y(t_j)$ eines Genes zum jeweiligen Zeitpunkt t_j werden per *Least-Square-Fit* an ein lineares Modell mit den beiden harmonischen Basisschwingungen angepasst:

$$y(t_j) = \beta_s \sin\left(\frac{2\pi}{T} t_j\right) + \beta_c \cos\left(\frac{2\pi}{T} t_j\right) + r(t_j). \tag{6.3}$$

Zur Detektion harmonisch schwingender Expressionsmuster, entsprechend des zeitlichen Rahmens der Zellentwicklung, wurde für die Periode T das von Shedden und Cooper publizierte nominale Zellteilungsintervall von 66 Minuten gewählt [SC02]. Mit Hilfe von Formel 6.3 kann der Expressionswert $y(t_j)$ in den interessierenden harmonischen Anteil

$$h(t_j) = \beta_s \sin\left(\frac{2\pi}{T} t_j\right) + \beta_c \cos\left(\frac{2\pi}{T} t_j\right), \tag{6.4}$$

und ein Restglied $r(t_j)$ zerlegt werden. $r(t_j)$ quantifiziert den nichtperiodischen Anteil von $y(t_j)$ bzw. den Anteil von Schwingungen mit Perioden, die sich signifikant von der für T gewählten unterscheiden. Um die Güte der Modellanpassung

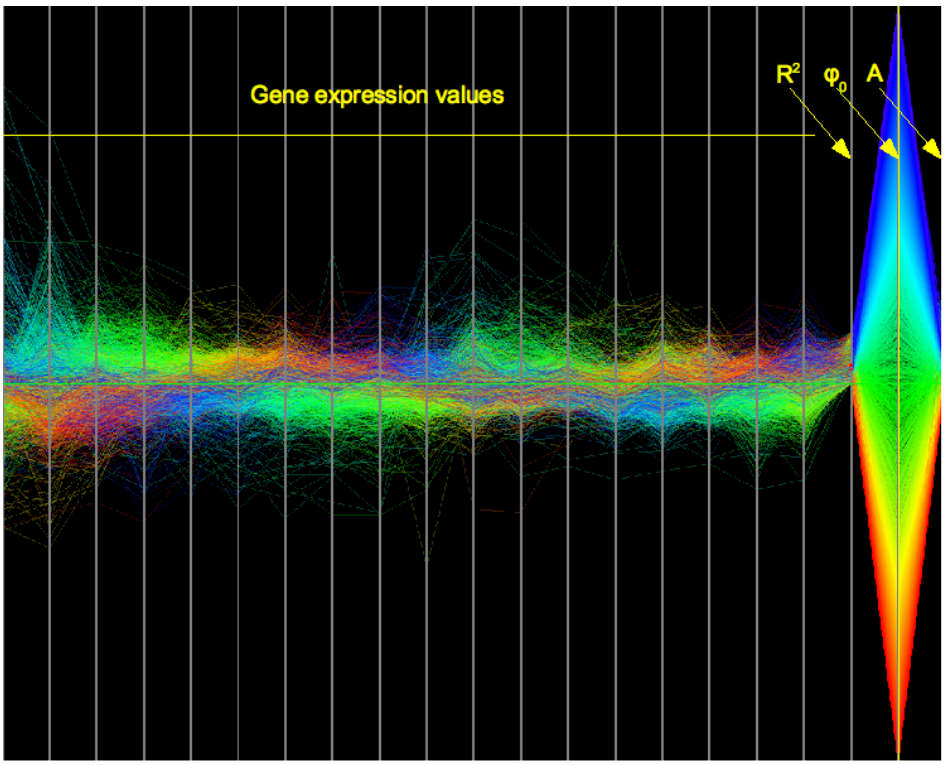


Abbildung 6.7: Erweiterter PKP der zellzyklusassoziierten Gene des Spellman α-Datensatzes [SSZ+98]. Die ersten 18 Dimensionen des PKPs repräsentieren die von Spellman et al. gemessenen Genexpressionswerte zu den 18 Zeitpunkten für die per α-Faktor synchronisierten Hefe-Zellen. Die letzten drei Dimensionen bilden einen anwendungsfallbezogenen statistischen Parameterraum und entsprechen dem Ergebnis einer *Harmonischen Regressionsanalyse* (HRA): das Bestimmtheitsmaß (oder auch Determinationskoeffizient) R^2, der Nullphasenwinkel φ_0 und die Amplitude A der geschätzten Schwingung (siehe Text für die Details). Die Linienzüge des PKPs sind entsprechend der Nullphasendimension eingefärbt, so dass die periodischen Änderungen des Genexpressionslevels verschiedener Gruppen von Genen sehr leicht erkannt werden können.

abschätzen zu können, erfolgt für jedes einzelne Gen die Berechnung des Bestimmtheitsmaßes R^2 des jeweiligen linearen Modells:

$$R^2 = \frac{SS_R}{SS_t} = \frac{\sum (\hat{y}_j - \bar{y})^2}{\sum (y_j - \bar{y})^2}, \tag{6.5}$$

das den Anteil der vom Modell erklärten Variation SS_R (*Regression Sum Of Squares*) im Verhältnis zur insgesamt in den Daten enthaltenen Variation SS_t (*Total Sum Of Squares*) angibt. Der Wertebereich von R^2 liegt zwischen 0 und 1 ($0 \leq R^2 \leq 1$) wobei $R^2 = 1$ ein vollständig passendes Modell bedeutet und $R^2 = 0$ das genaue Gegenteil.

Aufgrund der für den Betrachter besseren Vorstellbarkeit ist eine Darstellung des harmonischen Anteils $h(t_j)$ als einzelne phasenverschobene, skalierte Sinusschwin-

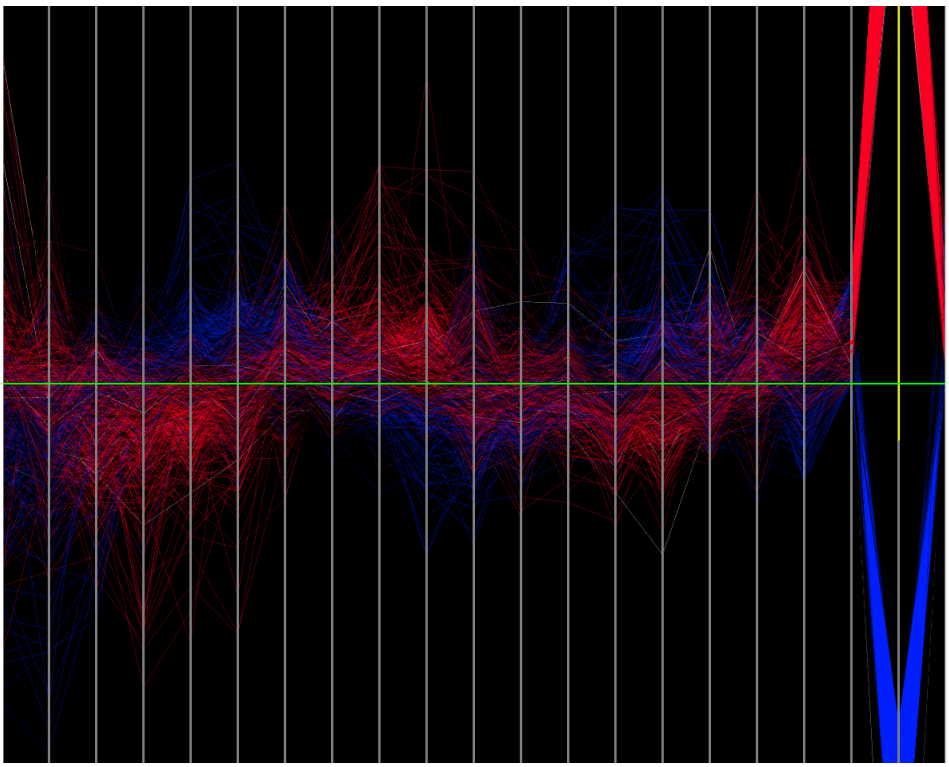


Abbildung 6.8: Selektion zweier Gengruppen mit antikorreliertem Genexpressionsmuster im α-Datensatz von Spellman.

gung für die Visualisierung viel geeigneter:

$$h(t_i) = A * \sin\left(\frac{2\pi}{T_k} t_j + \varphi_0\right). \tag{6.6}$$

Die Amplitude A und der Nullphasenwinkel φ_0 kann aus den Koeffizienten β_s und β_c von Formel 6.4 mit Hilfe der trigonometrischen Additionstheoreme gewonnen werden:

$$\begin{aligned} h(t_i) = A * \cos(\varphi_0) \sin\left(\frac{2\pi}{T_k} t_j\right) &+ A * \sin(\varphi_0) \cos\left(\frac{2\pi}{T_k} t_j\right) \\ \Rightarrow \beta_s = A * \cos(\varphi_0) &\wedge \beta_c = A * \sin(\varphi_0) \\ \Rightarrow A = \sqrt{\beta_s^2 + \beta_c^2} &\wedge \varphi_0 = \tan^{-1}\left(\frac{\beta_c}{\beta_s}\right). \end{aligned}$$

Abbildung 6.7 zeigt die Darstellung der Expressionswerte des α-Datensatzes, also des eigentlichen Werteraums, zusammen mit den Dimensionen für R^2, φ_0 und A, dem abgeleiteten (statistischen) Parameterraum, als erweiterten PKP in SpRay. Die Farbkodierung erfolgte mit Hilfe einer Rainbow-Map anhand der Dimension für den Nullphasenwinkel φ_0. Die Gruppierungen der periodischen Expressionsmuster verschiedener Gene können so sehr deutlich auch im ursprünglichen Werteraum

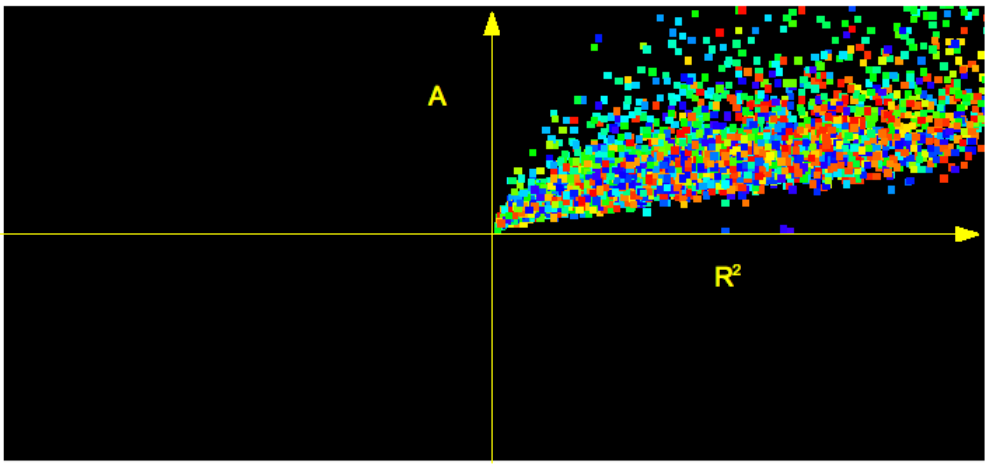


Abbildung 6.9: Darstellung des Scatterplots der Dimensionen für A, auf der y-Achse, gegen R^2, auf der x-Achse, wobei die jeweiligen Datenpunkte in Abhängigkeit von der Lage auf der Nullphasenachse φ_0 eingefärbt sind. Ein Datenpunkt entpricht der Projektion des Genprofils auf die Achsen R^2 und A. Die Form der räumlichen Clusterung der Expressionprofile in der $R^2 - A-$Ebene lässt eine vorhandene Relation zwischen der Amplitude und dem Bestimmtheitsmaß erkennen $A \sim R^2$, wohingegen die ungeordnete Gleichverteilung der Farbwerte gegen eine Relation bezüglich des Nullphasenwinkels φ_0 spricht.

der Beobachtungen visuell erkannt und untersucht werden. Die Visualisierung in Abhängigkeit des abgeleiteten Wertes φ_0 kann damit sehr gut zu einer besseren Einsicht in ursprünglich über- bzw. verdeckte Strukturierungsdetails der Daten führen. Beispielhaft sei hier die Auswahl zweier Gengruppen mit antikorreliertem Expressionsmuster in SpRay erwähnt. Anhand der φ_0-Dimension und basierend auf der Darstellung 6.7 lässt sich diese Aufgabe sehr leicht und schnell lösen, siehe Abbildung 6.8.

Komplexere und umfangreichere Fragen können durch visuelle Datenexploration mit Hilfe der zusätzlich vorhandenen Datensichten beantwortet werden. Abbildung 6.9 zeigt eine derartige Darstellung. Hier sind drei verschiedene Informationsdimensionen in einem Scatterplot abgebildet. Die Amplitude A ist auf der y-Achse gegen das Bestimmtheitsmaß R^2 auf der x-Achse abgebildet. Die Farbkodierung entspricht der im PKP gewählten Form nach der Lage auf der Achse für den Nullphasenwinkel φ_0. Es ist möglich abzulesen, ob hohe Werte für die Amplitude A mit einem hohen Bestimmtheitsmaß R^2 korreliert sind und ob zusätzlich ein Zusammenhang zum Nullphasenwinkel φ_0 existiert. Aus der räumlichen Lokalisierung der Punktwolke in der $R^2 - A-$Ebene lässt sich tatsächlich eine positive Korrelation zwischen A und R^2 erkennen, während kein geordnetes Muster in Bezug auf φ_0 identifizierbar ist.

Als Ursache für die vorliegenden Effekte wären verschiedene Erklärungen denkbar – methodische wie auch inhaltliche. So ist beispielsweise eine hohe Amplitude A mit

einem insgesamt stärkeren und deutlicheren Signal verbunden, das damit relativ betrachtet weniger von zufallsbedingten Störungen beeinflusst ist. Eine harmonische Regression eines qualitativ besseren Signals sollte auch eine bessere Zerlegung in die harmonische Basis ermöglichen (größeres R^2). Auf der anderen Seite spielt der Nullphasenwinkel φ_0 keine vergleichbare Rolle für die Qualität der Zerlegung des Signals.

Auch inhaltlich ließe sich argumentieren, beispielsweise wäre es möglich, dass gerade zellzyklusassoziierte Gene ein deutlicher ausgeprägtes Schwingungsmuster mit größeren statt geringeren Ausschlägen zeigen (größere Amplitude A) und sich deshalb diese Expressionsprofile besser auf die harmonischen Basisfunktionen abbilden lassen (höheres R^2). Die Unabhängigkeit vom Nullphasenwinkel φ_0 würde dafür sprechen, dass zellzyklusassoziierte Gene in ihrer Mehrheit nicht einer bestimmten Phase zugeordnet, sondern relativ konstant über alle Phasen verteilt sind.

6.5.2 Genexpressionsänderungen bei trainierten Halbmarathonläufern unter Ausdauerbelastung

Diese Fallstudie verdeutlicht nicht nur als Prototyp die identifizierte Aufgabenklasse 2 der evidenzbasierten Visualisierungen von Abschnitt 6.4, sondern zeigt zugleich sehr gut die wechselseitige Beziehung zwischen Visualisierung und Statistik auf. Während im vorhergehenden Abschnitt 6.5.1 mehr die Einflußnahme des statistischen Parameterraumes auf die Visualisierung dominierte, dient die Visualisierung in diesem Falle einer problemangepassten Auswahl der statistischen Parameter.

Um differentiell exprimierte Gene verlässlich detektieren zu können, hat es sich allgemein durchgesetzt, Methoden des statistischen Testens zu verwenden. Aufgrund der Vielzahl von Genen, die auf einem Microarray parallel in einem einzigen Durchgang gemessen werden, tritt das Problem des multiplen Testens massiv auf. Eine nach wie vor allgemein ungelöste Frage ist dabei die Wahl der geeignetsten Methode dem entgegenzuwirken. In der klassischen Statistik existiert bereits eine Reihe von Verfahren zur Korrektur des Problems des multiplen Testens [SH06]. Allerdings tritt dort diese Problematik nicht in einem solch massiven Umfang auf, weshalb auch speziell adaptierte Methoden auf dem Gebiet der Microarrayauswertung eingeführt wurden [Dra03, GCH+05, WM04]. Die Wahl der geeignetsten Korrekturmethode ist somit ein schwieriges Problem der Genexpressionsanalyse mit Hilfe von Microarrays, das je nach spezifischer Fragestellung neu gelöst werden muß. Ist die Korrekturmethode zu konservativ, so sichert sie eine gewünschte niedrige Quote an falschpositiven Resultaten (*Falschpositivrate* (FP)), aber andererseits werden möglicherweise wichtige und interessante Änderungen des Expressionslevels bestimmter Gene nicht erkannt, d.h. die Rate an falschnegativen Resultaten (*Falschnegativrate* (FN)) ist sehr hoch. Gerade bei Studien am Menschen bzw. höheren Lebewesen und insbesondere bei den äußerst interessanten Steuergenen zur Regulation vieler Prozesse sind eher kleinere Expressionsänderungen mit großen physiologischen Auswirkungen zu erwarten. Zusätzlich sorgt die große Menge von

Gensonden auf einem einzigen hochdichten Chip für sehr große Korrekturfaktoren bei den Korrekturmethoden, wodurch die Wahrscheinlichkeit für das Übersehen wichtiger Expressionsänderungen steigt. Andererseits würde eine hohe Falschpositivrate die anschließenden Untersuchungen zur Verifizierung der Ergebnisse, wie beispielsweise eine RT-PCR der einzelnen gefundenen Gene, sehr zeit- und materialaufwendig, umfangreich und teuer machen. Basierend auf den genauen Zielen der Studie muss ein jeweiliger Kompromiss zwischen diesen beiden Extremen gefunden werden. Hierbei kann eine geeignete Visualisierung gute Hinweise für das richtige Maß geben.

Der hier verwendete Datensatz entstammt einer Studie, die in enger Kooperation mit der Sport- und Transfusionsmedizin Tübingen entstand, siehe auch Kapitel 8 dieser Arbeit und [ZFD+05, ZZD+05]. Diese Studie untersuchte den Einfluss, den Ausdauerbelastungen auf das Immunsystem haben. Es ist allgemeiner Konsens, dass ein großer Effekt existiert, der auf zweierlei Vorgänge zurückzuführen ist - einerseits eine Verschiebung zwischen den verschiedenen Zellpopulationen des peripheren Blutes und andererseits eine Änderung der Genexpression.

Zur Messung der Genexpressionswerte wurde das in Kapitel 7 vorgestellte eigenentwickelte cDNA-Microarray verwendet. Dieses wurde speziell mit Sonden für Gene ausgestattet, die in die Regulation des Immunsystems und der Stressantwort involviert sind. Damit war es möglich, die verschiedenen Regulationsmechanismen in einer systematischen und umfassenden Weise zu untersuchen.

Grundlage der hier vorgestellten Daten ist die Untersuchung von Blutproben, die 8 gut trainierten Halbmarathonläufern (38 ± 11.8 Jahre, *Body-Mass-Index* (BMI) 23.6 ± 1.8) in Ruhe vor dem Lauf ($= t_0$), kurz nach dem Lauf ($= t_1$, bis zu 15 Minuten) und 24 Stunden danach ($= t_2$) entnommen wurden.

Die interessantesten und deutlichsten Effekte konnten zwischen den Zeitpunkten t_0, kurz vor, und t_1, kurz nach dem Lauf, beobachtet werden (für die genauere Analyse siehe Kapitel 8 und [ZFD+05, ZZD+05]). So wurden interessante Änderungen im Expressionslevel bestimmter inflammatorischer Gene gefunden und im Besonderen ein Bezug zur Regulation von anti-oxidativen Prozessen. Beides ist ein Indikator für den erhöhten Stresspegel, dem der Körper ausgesetzt ist. Eine ausführlichere Diskussion findet sich in Kapitel 8 bzw. [ZFD+05, ZZD+05].

Für die einzelnen Marathonläufer wird das logarithmierte Verhältnis der Expressionswerte zwischen den Zeitpunkten t_1 und t_0 bestimmt, der Log-Ratio M_r:

$$M_r = log\left(\frac{expr_{t_1}}{expr_{t_0}}\right). \tag{6.7}$$

Die Werte der 8 Marathonläufer stellen biologische Replikate dar, um auf die Verhältnisse in der zugrunde liegenden Basispopulation schließen zu können. Die 8 Replikate bilden den originalen Bildraum des erweiterten PKPs in Abbildung 6.10 und werden den ersten 8 Dimensionen zugewiesen. Die folgenden 10 Dimensionen entsprechen dem abgeleiteten statistischen Parameterraum, bestehend aus dem arithmetischen Mittel `m`, der Standardabweichung `sd`, dem unkorrigierten p-Value

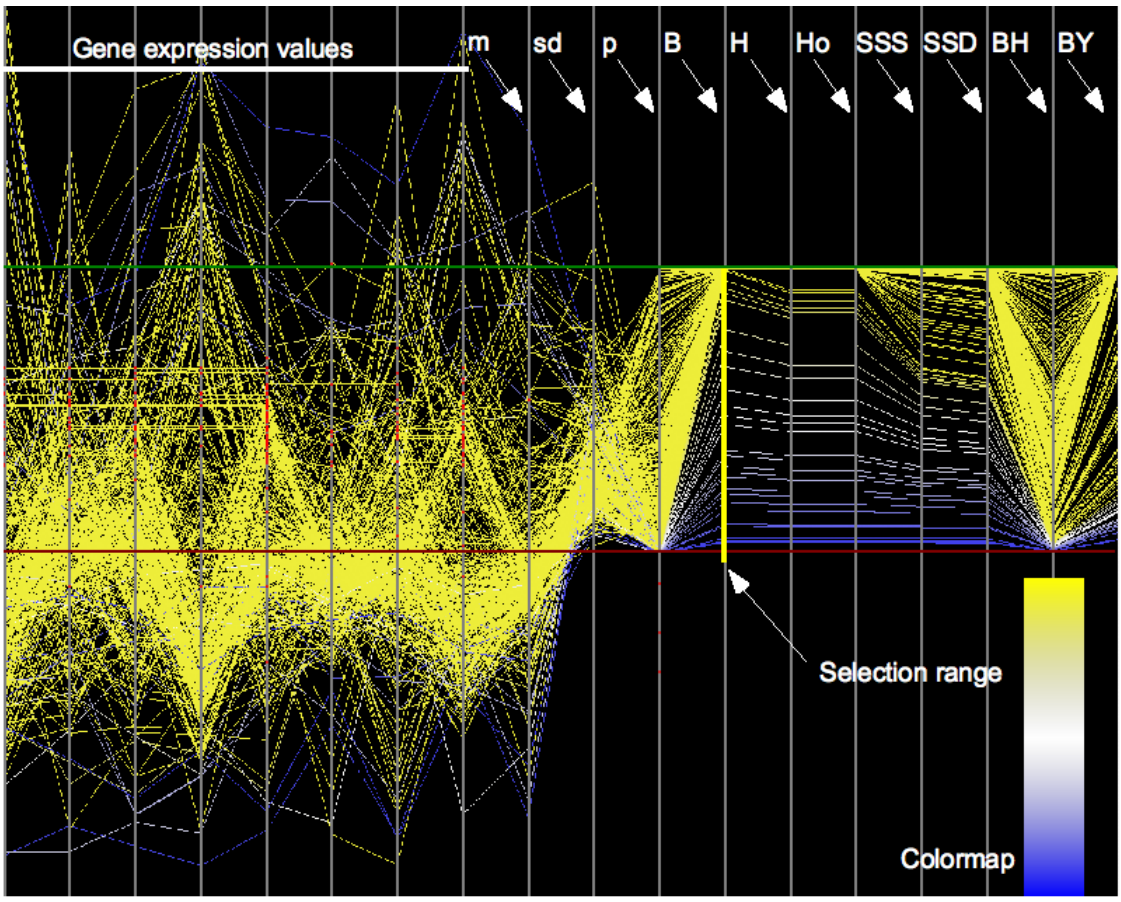


Abbildung 6.10: Der erweiterte PKP stellt in den ersten 8 Dimensionen die Log-Ratios der Genexpressionswerte von 8 gut trainierten Halbmarathonläufern dar. Die Log-Ratios geben das logarithmierte Verhältnis vom Zeitpunkt kurz vor dem Lauf (t_0) und direkt nach dem Lauf (t_1) an. Die sich daran anschließenden 10 Dimensionen entsprechen 10 aus den Originaldaten abgeleiteten statistischen Parametern (das arithmetische Mittel (`m`) sowie die Standardabweichung (`sd`) der Log-Ratios, dem unkorrigierten p-Value für den t-Test (`p`) und 7 p-Values, die nach jeweils unterschiedlichen Methoden gegen multiples Testen korrigiert wurden, siehe Text für weitere Details und die Bedeutung der restlichen Achsen). Das gelbe Ende des Farbkodierungsspektrums repräsentiert p-Values nahe 1, also letzlich keine Signifikanz für differentielle Genexpression, während das blaue Ende des Spektrums Werte nahe 0 und damit eine hohe Signifikanz repräsentieren.

eines Einstichproben-t-Tests gegen die Nullhypothese keiner differentiellen Expression zwischen den Zeitpunkten t_1 und t_0, d.h. H_0: $\mu_{M_r} = 0$ und sieben p-Werte die verschiedenen Korrekturverfahren für multiples Testen entstammen. Bei den verwendeten Verfahren handelt es sich um die Korrekturen nach Bonferroni `B` [Bon36], Holm `H` [Hol79], Hochberg `Ho` [Hoc88], Sidak `SSS`/`SSD` [Sid67], Benjamini-Hochberg `BH` [BH95] und Benjamini-Yekutieli `BY` [BY01]. Insgesamt werden 18 Dimensionen mit 345 Linienzügen für jedes einzelne Gen im erweiterten PKP dargestellt.

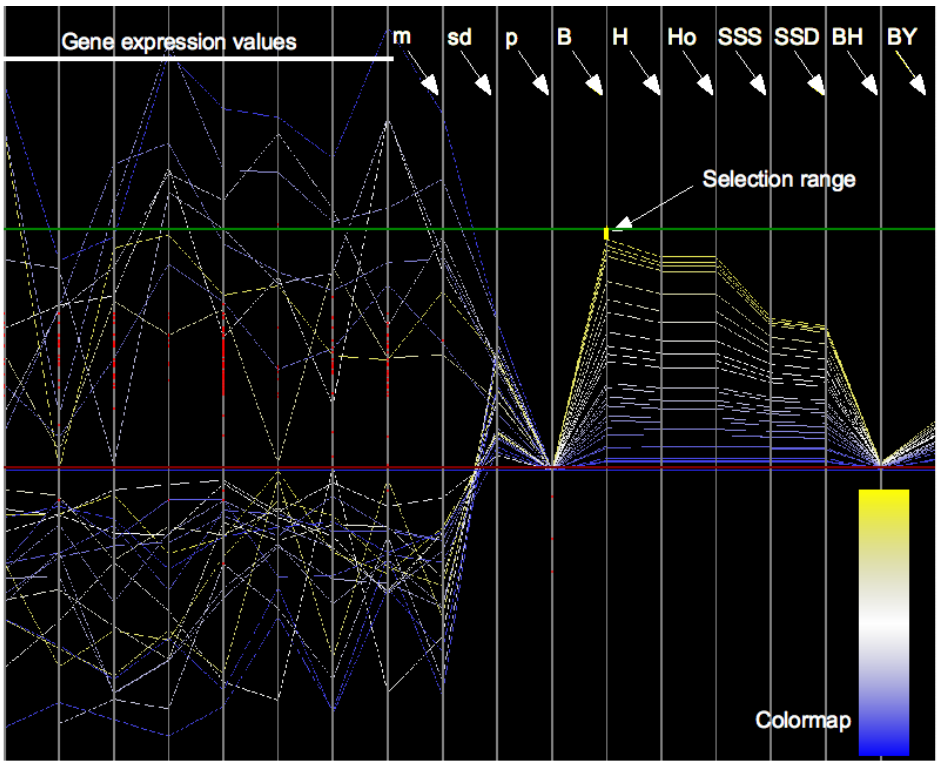


Abbildung 6.11: Visualisierung der gleichen Daten wie in Abbildung 6.10 unter Ausblendung der insignifikanten Gene ($p_{bonf} > 0.99$) durch entsprechende grafische Selektion auf der Dimensionsachse der Bonferroni-korrigierten p-Values (gelbe vertikale Linie auf der `B`-Achse).

Abbildung 6.10 vermittelt eine detaillierte Vorstellung der innerhalb dieser Studie gemessenen 8 Expressionswerte und der 10 abgeleiteten statistischen Parameter. Die Farbkodierung erfolgte entsprechend den Bonferroni-korrigierten p-Values. Diese Methode ist die konservativste und stringenteste. Sie wurde für diese Studie ausgewählt, um möglichst verlässliche Kandidatengene zu identifizieren, die Rate der Falschpositiven bewusst gering zu halten und die Anschlussuntersuchungen im zeitlichen und finanziellen Rahmen zu halten. Der zu erreichende Signifikanzlevel des t-Tests wurde vor dessen Durchführung auf 5 % festgelegt ($\alpha = 0.5$), d.h. alle Gene deren korrigierter p-Value unter 0.05 fiel, wurden als signifikant differentiell exprimiert angesehen. Betrachtet man Abbildung 6.10 so kann man darin nicht die Expressionsprofile der interessanteren Gene erkennen, also von Genen mit einem korrigierten p-Value kleiner als 1, da sie mehrheitlich von den Profilen für Gene mit keinem signifikant differentiellen Expressionsmuster ($p_{corr} \simeq 1$) verdeckt werden. Somit ist es notwendig, die Darstellung der irrelevanten Expressionsprofile in der Visualisierung zu unterdrücken und im Gegenzug die Linienzüge der interessierenden Gene hervorzuheben, wie in Abbildung 6.11 zu sehen ist. In Abbildung 6.12 sind speziell alle 13 Expressionsprofile von Genen mit einem korrigierten p-Wert ≤ 0.2 rot hervorgehoben. 6 dieser 13 haben einen korrigierten p-Wert von $\leq$

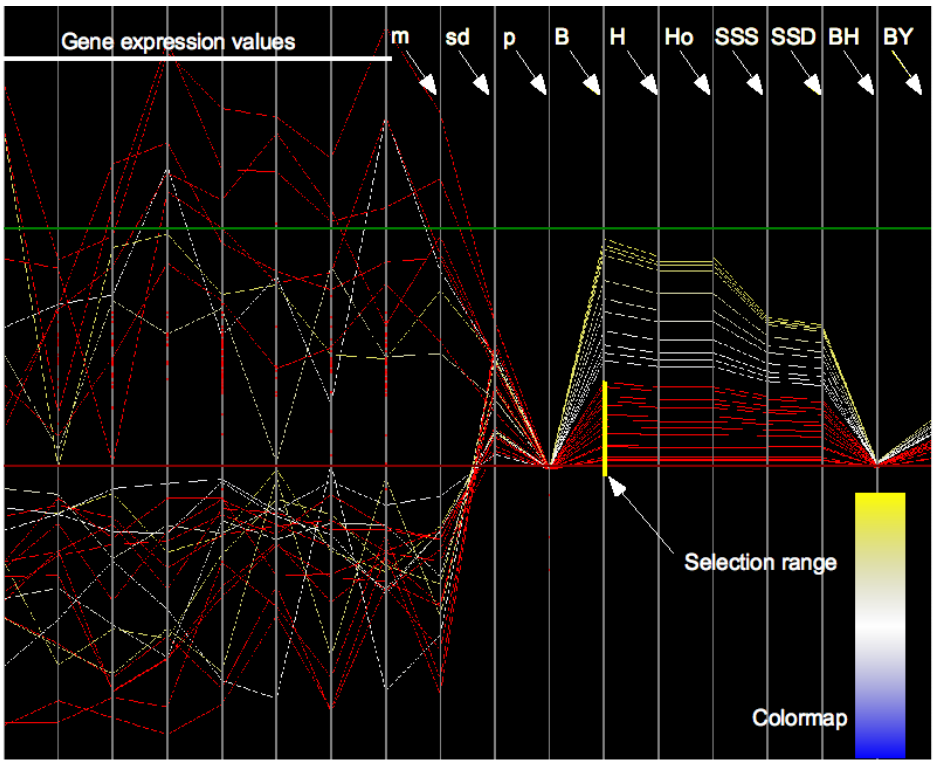


Abbildung 6.12: Markierung (rote Hervorhebung) der am Ende wichtigsten und signifikanten Genexpressionsprofile (p-Value kleiner als 0.2, der Auswahlbereich ist gelb auf der `B`-Achse hervorgehoben). Diese Gene wurden für eine weitere Validierung per RT-PCR vorgesehen.

0.05. Dies sind die, im Bereich der p-Werte, nahe der Nullinie liegenden markierten Profile. Lediglich eines dieser 6 Gene ist im Zeitpunkt t_1 signifikant hochreguliert und liegt deshalb im Bereich der Expressionswerte oberhalb der Nulllinie des erweiterten PKPs. Die anderen sind in t_1 signifikant herrunterreguliert und liegen mit ihren Expressionswerten deswegen unterhalb der Nulllinie (siehe auch Tabelle 8.2 auf Seite 169).

6.5.3 Validierung eines neu entworfenen cDNA-Microarrays

Das Beispiel zur Verdeutlichung der dritten identifizierten Anwendungsklasse in Abschnitt 6.4 besteht aus mehreren Experimenten zur Validierung eines neu entwickelten cDNA-Microarrays. Dieses Array wurde als ein umfangreicherer und erweiterter Nachfolger des Arrays entwickelt, das Grundlage für die Halbmarathon-Studie im letzten Abschnitt (Abschnitt 6.5.2), dem Kapitel 8 und dem Teil 7 war. Der Validierungsdatensatz (siehe auch Abbildung 6.13) besteht aus 5 sogenannten Self-Self-Experimenten $SE_1, \ldots, SE_5$ (die ersten 5 Dimensionen im PKP von Abbildung 6.13) und einem realen Experiment `PT` (sechste Dimension im PKP von Abbildung 6.13). Bei Self-Self-Experimenten befindet sich in beiden Kanälen des hier

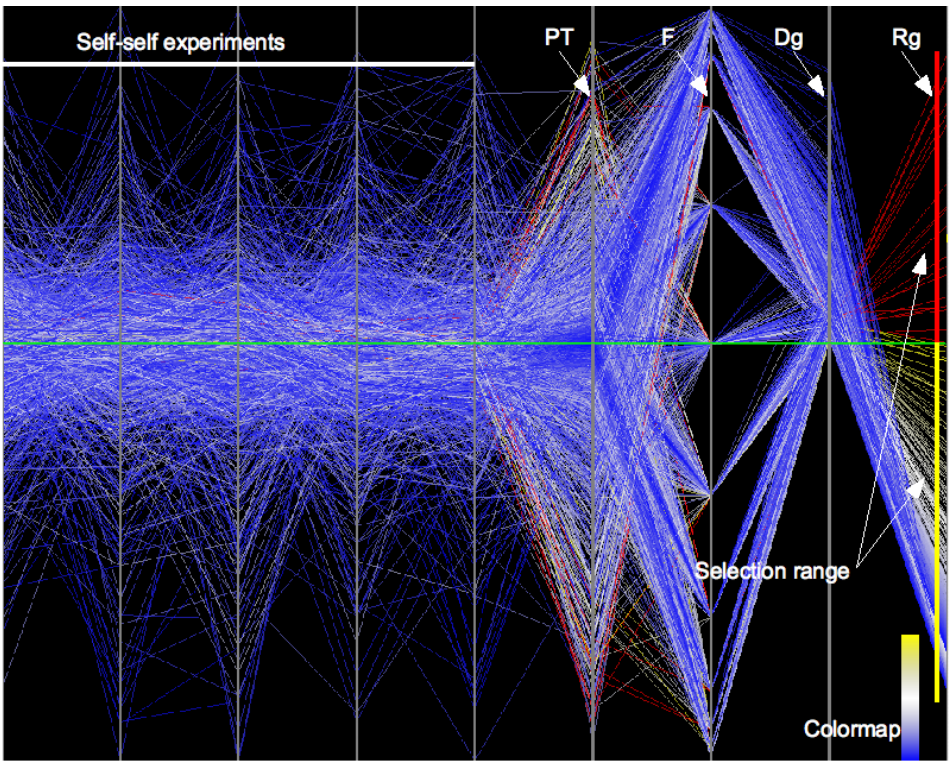


Abbildung 6.13: Darstellung des kompletten Datensatzes zur Validierung eines neuen cDNA-Microarrays. Die ersten 5 Dimensionen stellen die Log-Ratios (logarithmierter *Fold Change* (FC)) der Self-Self-Experimente dar ($SE_1, \ldots, SE_5$), die sechste den des realen Experiments (PT), daran schließen sich die Dimensionen des statistischen Parameterraumes an, bestehend aus der Anzahl der in der Bildanalyse gesetzten Qualitätsflags (F) und der beiden statistischen Parameter D_g und R_g (siehe Text für eine nähere Beschreibung der Parameter). Die grüne dünne Linie markiert die Nulllinie, während die dickere gelbe und rote Linie die Auswahlbereiche für die Einfärbung kennzeichnen. Der in Gelb ausgewählte Bereich wurde entsprechend der unten eingeblendeten Farbpalette eingefärbt, der in Rot wurde auf Grund seiner hohen Relevanz komplett mit Rot hervorgehoben. Zu beachten ist, dass die vertikale Achse logarithmisch skaliert ist.

genutzten zweikanaligen Microarrays das gleiche biologische Material. Aus diesem Grund wäre eigentlich zu erwarten, dass das Array keinen Expressionsunterschied detektieren kann. Abweichungen davon werden von den vorhandenen technischen Störquellen des Experimentes und des Meßprozesses hervorgerufen. Damit eignen sich Self-Self-Experimente sehr gut, die technischen Messfehler eines Microarrays einzuschätzen. Bei der Betrachtung der Daten in Abbildung 6.13 fallen auf den Achsen der Self-Self-Experimente extreme Ausreißer auf, die eigentlich, bei bloßer Betrachtung des Experiments, ohne den in der Abbildung gezeigten Kontext, eine differentielle Genexpression für das betrachtete Gen nahelegen.

Das reale Experiment bestand aus dem Vergleich zweier Speichelproben der gleichen Person. Die eine wurde kurz vor einer extremen physischen Belastung

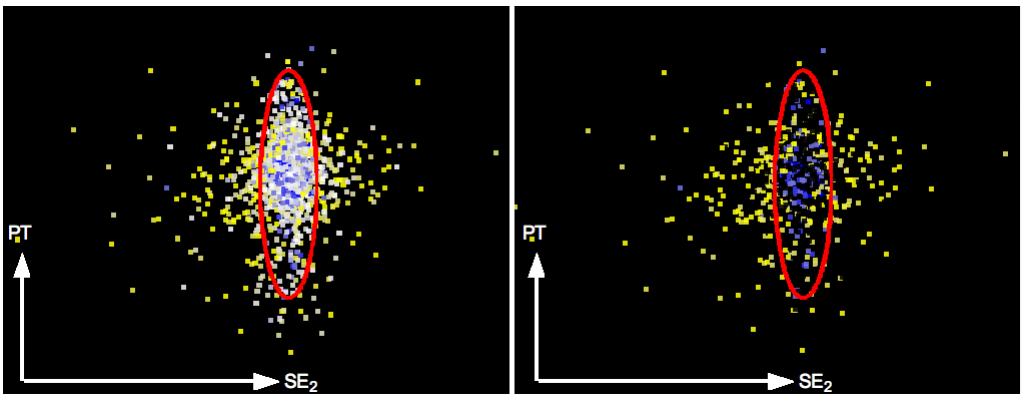


Abbildung 6.14: Scatterplot des realen Experimentes `PT` (y-Achse) gegen das Self-Self-Experiment SE_2 (x-Achse). Die x- und die y-Achsen besitzen die gleiche Skalierung. Im linken Scatterplot sind alle Daten dargestellt, während im rechten die Datenpunkte ausgeblendet wurden, die eine mittlere Anzahl an gesetzten Qualitätsflags aus der Bildverarbeitung aufweisen. Dies entspricht der PKP-Darstellung in Bild 6.16. Zu erkennen ist die kompaktere Verteilung der Log-Ratios des realen Experiments im Vergleich zum Self-Self-Experiment. Interessant ist insbesondere die Verteilung der vertrauenswürdigsten Datenpunkte, d.h. der Datenpunkte mit keinen bzw. wenig gesetzten Qualitätsflags in der Bildanalyse. Während sie im realen Experiment fast den vollen Wertebereich dieses Experimentes überdecken, konzentrieren sie sich beim Self-Self-Test in einem sehr engen Bereich um den zu erwartenden Messwert Null herum (rotes Ellipsoid). Das unterstreicht noch einmal die Vertrauenswürdigkeit dieser Datenpunkte und den prognostischen Wert der in der Bildanalyse gewonnenen Qualitätseinschätzung.

durch Sport entnommen und die andere umgehend danach. In diesem Fall ist eine Änderung des Expressionslevels verschiedener Gene zu erwarten.

Im Vergleich des realen Experiments mit den Self-Self-Experimenten ist zu sehen, dass die gemessenen Log-Ratios für ersteres einen kleineren Wertebereich überdecken, auf diesem jedoch breiter verteilt sind. Dies lässt sich in allen drei Plots, dem PKP (Abbildung 6.13), dem Scatterplot (Abbildung 6.14) und den Histogrammplots (Abbildung 6.15), gut erkennen. Im Vergleich dieser beiden Experimente wird der Unterschied zwischen rein technischen und biologischen Signalen deutlich. Im Falle des realen Experimentes sind Log-Ratios, verschieden von Null, für eine größere Anzahl von Genen zu erwarten. Überraschend ist jedoch, daß die Verteilung der Self-Self-Experimente extremere Ausreißer als die des realen Experiments aufweist.

Zum besseren Verständnis der zu beobachtenden Effekte, wurden drei statistische Parameter als Basis des statistischen Parameterraums für den erweiterten PKP berechnet und hinzugefügt (die letzten drei Dimensionen im PKP von Abbildung 6.13). Beim ersten handelt es sich um den Anteil `F` an gesetzten Flags für den Spot eines Genes über alle Experimente hinweg. Gewonnen werden diese Flags in der Bildverarbeitung, d.h. bei der Spotdetektion und Signalextraktion. Sie

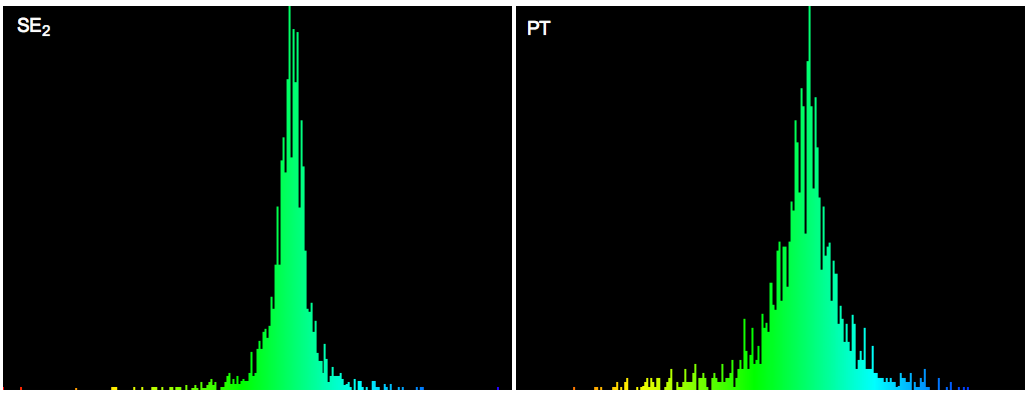


Abbildung 6.15: Histogramme der Log-Ratios des Self-Self-Experiments SE_2 (links) und des realen Experiments `PT` (rechts). Beide Histogrammdarstellungen weisen die gleiche Achsenskalierung auf. Das Histogramm des realen Experiments lässt im Unterschied zu allen Self-Self-Experimenten, hier beispielhaft nur SE_2 gezeigt, eine etwas gedrungenere, breitere Verteilung erkennen (heavy tails). Wobei allerdings der insgesamt überstreckte Bereich kleiner ist als der der Self-Self-Experimente, bei denen extremere Werte auftreten.

kennzeichnen einen Spot und damit auch ein zugehöriges Expressionssignal von fraglicher Qualität. Ein Gen für dessen Spot kein einziges Flag gesetzt ist, hat damit einen Wert von 0.0 und stellt ein Expressionsprofil maximaler Qualität dar, demgegenüber steht ein Wert von 1.0 für ein Gen, bei dem Flags in allen Experimenten gesetzt wurden. Da alle 6 Experimente in Form eines Dye-Swap-Experiments [SCPT+04, TOR+01, YDLS01] ausgeführt wurden, ergeben sich maximal 12 gesetzte Flags:

$$\mathrm{F} = \frac{\mathrm{Flags}}{12} \tag{6.8}$$

Erkennbar sind die 12 möglichen Wertestufen in der Darstellung der `F`-Dimension im PKP von Abbildung 6.13. Dabei muß jedoch die logarithmische Skalierung der y-Achsenrichtung berücksichtigt werden.

Um Unterschiede im Verhalten der jeweils qualitativ besten und schlechtesten Genexpressionsprofile einschätzen zu können, wurden in Abbildung 6.16 alle Profile bis auf die drei höchsten Qualitätstufen (niedrige Flagrate, blauer Farbton) und die drei niedrigsten Qualitätsstufen (hohe Flagrate, gelber Farbton) ausgeblendet. In dieser Visualisierung zeigt sich sehr deutlich die Nützlichkeit und der hohe prognostische Wert der Qualitätseinschätzung, die in der Bildanalyse gewonnen wurde. Die Profile von hoher und höchster Qualität zeigen innerhalb der Self-Self-Experimente SE_i eine sehr konsistente Messung des Log-Ratios in einem engen Bereich um die zu erwartende Null. Demgegenüber weichen im realen Experiment eine ganze Reihe der hochqualitativen Profile deutlich von Null ab. Diese Profile beschreiben die Messung eines wirklichen Genexpressionsunterschiedes im realen Experiment (PT) und damit den Nachweis eines tatsächlichen biologischen Effektes. Im Kontrast dazu weichen die Log-Ratios der Profile niedriger und niedrigster Qualität in

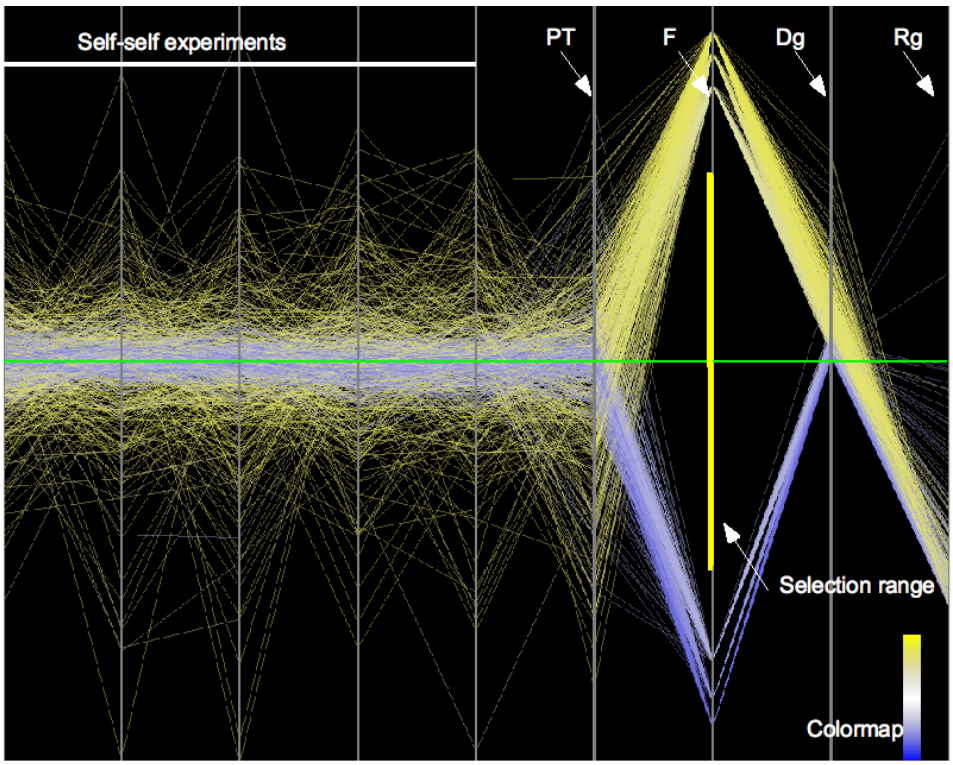


Abbildung 6.16: Gegenüberstellung der anhand der Qualitätsbewertung in der Bildanalyse als am verlässlichsten und unverlässlichsten eingeschätzten Genexpressionsprofile. Die Einfärbung der Linienzüge erfolgte mit Hilfe der rechts unten gezeigten Farbpalette. Profile mit einem sehr hohen bis maximalen Anteil an gesetzten Flags und damit von sehr schlechter Qualität, sind in verschiedenen Gelbtönen koloriert. Demgegenüber sind Profile mit minimalem Anteil bis gar keinem gesetzten Flag und damit hoher bis höchster Qualität, in Blautönen gefärbt. Profile von mittlerer Qualität wurden ausgeblendet (gelbe dicke Linie als Selektionsbereich). Sehr gut ist erkennbar, dass sich Expressionsprofile mit hoher bis höchster Qualität in den Self-Self-Experimenten in einem engen Bereich um Null herum bewegen, während die Verteilung im realen Experiment weiter gestreut ist.

den Self-Self-Experimenten SE_i signifikant weit vom zu erwartenden Nullniveau ab. Berücksichtigt man zusätzlich die logarithmische Skalierung der y-Achsenrichtung, so sind diese Unterschiede sogar noch höher zu bewerten.

Ebenfalls sehr gut zu sehen sind die Verteilungsunterschiede, sowohl für die unterschiedlichen Qualitätsstufen wie auch für die Verteilung der Log-Ratios insgesamt in den Scatterplots in Abbildung 6.14. Auch hier zeigt sich der enge Bereich um die Null herum, in dem die Expressionswerte hoher Qualität beim Self-Self-Experiment (SE_2 auf der x-Achse) liegen. Im Unterschied zu dem viel weiteren Intervall beim realen Experiment (`PT` auf der y-Achse). Als Resultat dieser beiden Effekte liegen die hochqualitativen Profile in dem rot markierten ellipsenförmigen Bereich des Scatterplots.

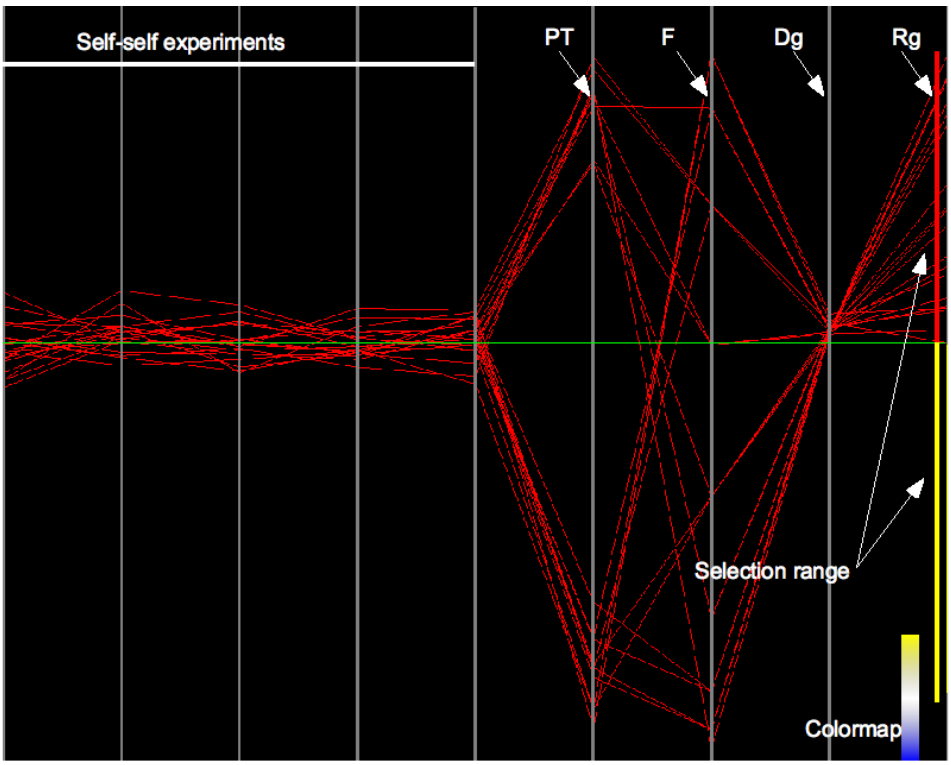


Abbildung 6.17: Visualisierung der gleichen Daten wie in Abbildung 6.13 unter Ausblendung aller wenig relevanten und unverlässlichen Genexpressionsprofile anhand der Dimension (R_g) für den statistischen Parameter der Relevanz R_g.

Die zweite statistische Größe D_g ist die mittlere Abweichung des Log-Ratios $M_{i,g}$ für ein Gen g innerhalb der 5 Self-Self-Experimente SE_i vom zu erwarteten Wert Null:

$$D_g = \frac{1}{N}\sum_{i=1}^{N}|M_{i,g} - 0| = \frac{1}{N}\sum_{i=1}^{N}|M_{i,g}| \quad \text{mit} \quad N = 5. \tag{6.9}$$

Expressionsprofile, die wesentlich vom zu erwartenden Log-Ratio von Null abweichen und damit einen hohen Wert D_g aufweisen, liefern ein deutliches Anzeichen für eine schlechte Qualität und für ein unverlässliches Expressionssignals.

Der dritte statistische Wert kombiniert die eben eingeführte Metrik D_g mit der Größe des Log-Ratios im realen Experiment `PT` $|M_g^{\mathrm{PT}}|$ derart, dass Gene mit einem hohen Expressionsunterschied in `PT` und einem niedrigen D_g auf einen hohen numerischen Wert abgebildet werden:

$$R_g = \frac{|M_g^{PT}|}{D_g}. \tag{6.10}$$

Damit kennzeichnen hohe Werte für R_g die interessierenden bzw. relevantesten Expressionsprofile. Das sind Profile, deren Detektion eines Expressionsunterschiedes

im realen Experiment durch eine konsistent erwartungsgemäße Messung in den Self-Self-Experimenten gestützt werden, wobei ein tatsächlich gemessener biologischer Effekt sehr warhscheinlich wird.

In Abbildung 6.17 wird die Relevanzmetrik R_g angewendet, um die interessantesten Expressionsprofile zu visualisieren. Profile, die zu dem mit der gelben, dicken Linie gekennzeichneten Auswahlbereich niedriger R_g-Werte gehören, wurden ausgeblendet und Profile mit einem hohen R_g-Wert (rote, dicke Linie) rot hervorgehoben. Erkennbar ist, dass die ausgefilterten Profile im realen Experiment `PT` eine hohe Distanz zum Nullniveau zeigen, während sie in den Self-Self-Experimenten nur eine sehr geringe Abweichung davon haben, insbesondere unter Berücksichtigung der logarithmischen Skalierung der y-Achsenrichtung. Allerdings werden einige dieser Expressionsprofile von der Qualitätseinschätzung der Bildanalyse als fraglich klassifiziert - hohe Werte für `F`. Dies stützt argumentativ das Vorgehen mancher Arbeitsgruppen, Spots, die durch die Bildanalyse als schlecht bewertet wurden, doch noch in entsprechend abgewertet gewichtetem Umfang in die Auswertung einfließen zu lassen [SS03, SYS03].

6.6 Diskussion der Erfahrungen aus der Anwendung

Sowohl das Beispiel mit künstlichen Daten, wie auch die drei realen Datensätze der entsprechenden Anwendungsklassen zeigen die nutzbringende Anwendung der Kombination aus PKP und anderen Visualisierungstechniken für die visuelle Exploration hochdimensionaler Daten, wie sie in der microarraybasierten Genexpressionsanalyse auftreten. Tendenzen und Gruppierungen, bzw. Cluster, innerhalb der Daten können mit Hilfe der Anwendung einer Farbkodierung nach verschiedenen Farbpaletten und dem Einsatz von Transparenz besser identifiziert werden.

Nicht immer bieten die Originaldaten eine ausreichende Strukturierung, um lediglich anhand einer geeigneten Form ihrer Darstellung eine hinreichend genaue visuelle Analyse durchführen zu können. Aus diesem Grund erweitert der Ansatz der visuellen Synthese des originalen Datenraumes mit dem Raum abgeleiteter statistischer Parameter den herkömmlichen PKP beträchtlich. Relationen, die nur schwer oder gar nicht mit Hilfe rein visueller Methoden in die Analyse einzubringen wären, können auf diesem indirekten Weg für die visuelle Datenexploration zugänglich gemacht werden. Beispielsweise lassen sich über den Raum der statistischen Parameter interaktiv Filter formulieren, um irrelevante Daten zu identifizieren und von der Darstellung auszuschließen, wie zum Beispiel in der Analyse des Halbmarathondatensatzes bei der Ausblendung der insignifikanten Gene (Übergang von Abbildung 6.10 zu 6.11). Auch die Darstellung der Spellman Daten in Abbildung 6.7 ist ein deutlicher Beleg für den Nutzen der kombinierten Darstellung. Erst durch die Auswahl der Farbkodierung, entsprechend der φ_0-Achse der HRA, können die verschiedenen zyklischen Expressionsmuster gut identifiziert und differenziert werden.

Von großem Vorteil bei der visuellen Datenexploration ist die Möglichkeit zu ei-

nem schnellen Feedback auf das „Spiel“ mit den einzelnen Parametern der Darstellungen hin. Das sich durch eine schnelle und gute Interaktion mit der Anwendung erreichen lässt. Die Wichtigkeit dieses Prozessablaufes ist bereits in verschiedenen Zusammenhängen erwähnt worden, beispielsweise für das Lernen neuer statistischer Zusammenhänge von Daten von Sanford Weisberg [FW06]. So lassen sich aus einer Visualisierung heraus neue Parameter oder Filter auf die bereits dargestellten und bearbeiteten Daten anwenden. Beispielhaft sei hier die Auswahl zweier Gengruppen mit antikorreliertem Expressionsmuster des Spellman-Datensatzes in SpRay erwähnt (Abbildung 6.8).

Ein weiterer großer Vorteil der kombinierten Darstellung im PKP ist die Möglichkeit, die Effekte verschiedener Auswertungsmethoden visualisieren zu können, wie es auch in der Exploration des Halbmarathondatensatz zu sehen ist. Im Hinblick auf die konkreten Ziele der durchgeführten Studie kann so die Verlässlichkeit und Stabilität von Ergebnissen, bezüglich zu Grunde liegender Modellannahmen, besser untersucht und eingeschätzt werden. Dies hilft, die einzelnen statistischen Methoden im Kontext der vorliegenden Analyse besser bewerten und verstehen zu können. Beispielsweise wird das Wesen der verschiedenen Korrekturmethoden in allen Darstellungen zum Halbmarathondatensatz sehr gut visualisiert (siehe Abbildung 6.10 bis 6.12). Es ist deutlich erkennbar, dass der untere Bereich der unkorrigierten p-Values (`p`) des t-Tests von allen Korrekturmethoden über einen größeren Bereich aufgespreizt wird. Die stringenteste Medhode ist die Korrektur nach Bonferroni (`B`), somit ist es auch diese Methode, die den unteren Bereich der p-Values auf nahezu den vollständigen zur Verfügung stehenden Wertebereich spreizt. Alle anderen Methoden spreizen weniger, insbesondere die Methoden nach Benjamini-Hochberg (`BH`) und Benjamini-Yekutieli (`BY`) skalieren auf ein kleineres Intervall. Die Relationen zwischen den Profilen auf den Achsen bleiben allerdings erhalten. Auch das ist im erweiterten PKP sofort zu erkennen, ohne dass eine nähere Kenntnis der Korrekturverfahren notwendig ist. Betrachtet man das für diese Studie festgelegte Signifikanzniveau von 5 %, so stellt man fest, dass die Variation der Methoden nicht groß ist und man sich auf das Ergebnis der strengen Bonferroni-Korrektur beziehen kann, ohne Gefahr zu laufen, durch diese Wahl zu viele interessante Expressionsprofile zu übersehen.

Auch das Beispiel der dritten Anwendungsklasse von Abschnitt 6.5 zur Validierung des neuen cDNA-Microarrays belegt den besonderen Mehrwert, den die kombinierte Darstellung aus Expressionswerten und geeigneter Statistik in SpRay bieten kann. So konnte gezeigt werden, dass alle Ausreißer der Self-Self-Experimente verlässlich durch die Flagrate `F` relativiert und alle wichtigen und verlässlichen Log-Ratios durch die Relevanzmetrik R_g detektiert werden konnten. Eine visuelle Exploration allein auf dem Datenraum der Log-Ratios hätte den Eindruck einer zu hohen Rate an technischen Störungen (Ausreißer in den Self-Self-Experimenten) erweckt.

Wie in Abschnitt 6.4 beschrieben, konnte mit der Implementierung des Ansatzes in SpRay eine Reihe von verschiedenen Farbpaletten auf frei wählbare Achsen des PKP angewendet werden. Zwar ist die Rainbow-Map in der Visualisierung sehr

diskutiert und umstritten, weil sie wahrnehmungsbedingt zum Teil unterschiedliche Distanzen bei Punkten gleichen Abstandes suggeriert, doch erwies sie sich in der Arbeit sowohl mit künstlichen wie auch mit realen Datensätzen als eine gute Option, unterschiedliche Expressionsmuster über mehrere Dimensionen, bzw. Konditionen, hinweg optisch zu trennen. Allerdings muss man einschränkend sagen, dass es vorerst vor allem um eine qualitative Differenzierung ging und nicht um eine quantitative. Dafür sollten möglicherweise andere Farbpaletten verwendet werden.

6.7 Ausblick - aussichtsreiche weitere Entwicklungsansätze

Insgesamt zeigt die Anwendung des in SpRay implementierten Ansatzes das große Potential des eingeschlagenen Weges für den Bereich der Microarrayanalyse auf. Die mit SpRay gemachten praktischen Erfahrungen bildeten die Grundlage für die erfolgreiche Einreichung eines Förderantrag im Rahmen des Schwerpunktprogramms „Scalable Visual Analytics: Interactive Visual Analysis Systems of Complex Information Spaces“ (SPP 1335) der DFG zur weiteren Entwicklung von SpRay und der damit verbundenen Konzepte.

SpRay zeigt in der vorliegenden Ausbaustufe, dass die bereits von anderen Autoren auf anderen Gebieten vorgeschlagene verbundene Darstellung von Parallelkoordinaten mit Scatterplots und Histogrammsichten ebenfalls im Bereich der Genexpressionsanalyse große Vorteile hat. Der größte Vorteil erwächst aus der engen Verbindung der visuellen Datenexploration mit einer statistischen Analyse und liegt damit ganz auf der Linie der neu entstandenen Forschungsrichtung Visual Analytics. Erreicht wird diese Kopplung im hier vorgeschlagenen Weg durch die integrierte Visualisierung von Originaldatenraum und abgeleitetem statistischen Parameterraum. Damit ist eine schnelle Validierung und Evaluierung der Anwendbarkeit der vorgesehenen statistischen Auswertungsmethoden möglich, das insgesamt zu einer besseren, standardisierteren und problemangepassteren Analyse, insbesondere von microarraybasierten Genexpressionsdaten, führen kann. Gestützt wird diese Argumentation auch durch sehr aktuelle Veröffentlichungen, wie [CHL+07], in der an einer Genexpressionsstudie beispielhaft das zwangsläufige Versagen einer rein statistischen Modellierung gezeigt wird, die erst durch den Einsatz begleitender visueller Datenexploration in eine adäquaten Lösung überführt werden konnte.

Es bieten sich eine Reihe konzeptioneller Weiterentwicklungen an, die auch Gegenstand des durch das Schwerpunktprogramm der DFG geförderten Projektes sind:

- Welche weiteren statistischen Methoden bzw. Parameter eignen sich am besten für eine visuelle Integration?
- Wie kann man diese Integration am besten gestalten?

- Lassen sich eventuell zusätzliche, anders geartet verbundene Sichten besser zur Visualisierung benutzen?

SpRay stellt die Implementierung und damit den prototypischen Testfall der in diesem Kapitel aufgeführten Konzepte und Ideen dar. SpRay ist somit ein guter Ausgangspunkt für die Entwicklung einer Anwendung für den breiteren und produktiven Einsatz. Bereits jetzt lassen sich zusätzliche synchrone Visualisierungssichten mit den Mitteln von SpRay simulieren, wie beispielsweise der verbreitete Volcano-Plot durch die Auswahl des Scatterplots, der entsprechenden Achsen des Parallekoordinatenplots und einer geeigneten Achsenskalierung. Als weitere Ausbaurichtungen von SpRay sollten im Rahmen des bewilligten Förderprojektes vor allem die folgenden in Angriff genommen werden:

- SpRay hat bislang einen begrenzten Umfang an statistischen Methoden. Es sollte deshalb so direkt wie möglich mit umfangreicheren Analysesystemen verbunden werden, wie z.B. Mayday oder R. Dadurch würde eine weitaus größere methodische Vielfalt für die Visualisierung zur Verfügung stehen.

- Eine Reihe weiterer Visualisierungsmethoden, insbesondere Verfahren aus dem Bereich der *Focus+Context* (F+C) Visualisierung - wie die bereits in SpRay implementierte angepasste Version einer *Table Lense*, sollten integriert werden.

- SpRay muß für eine funktionierende, reibungslose alltägliche Arbeit in Bezug auf die *Usability* verbessert werden. Das umfasst vor allem flexiblere Möglichkeiten zur Selektion von Datenpunkten, zum Export der in der Analyse gewonnenen Daten, zur Skalierung und Beschriftung der Achsen sowie verbesserten Zoomfunktionen in allen verfügbaren Views.

- Die Integration weiterer Farbpaletten mit anderen wahrnehmungsbedingten Eigenschaften steigert die Breite der Anwendbarkeit von SpRay. Insgesamt wäre ein Vereinfachung und Flexibilisierung des Umganges mit den Farbpaletten wünschenswert. So könnten zum Beispiel Farbpaletten nicht nur fest kodiert werden, sondern auch über entsprechende Dateien zur Laufzeit geladen werden. Damit können eigene, sehr speziell auf das Problem angepasste Abbildungen zwischen Werte- auf Farbraum verwendet werden.

- Interessant ist zu untersuchen, inwieweit eine Entkopplung der Farbkodierung zwischen den verschiedenen Datensichten, die visuelle Analyse unterstützen kann. Dies könnte noch weiter gehen bis hin zu einer separaten Farbkodierung lediglich für selektierte Datenpunkte bzw. Unterräume des Datenraumes.

Teil III

Fallstudien Array-basierter Genexpressionsanalysen aus der medizinischen Forschung

Kurzübersicht zu Teil III

Für den letzten Teil dieser Arbeit steht die praktische Anwendung der Genexpressionsanalyse innerhalb der medizinischen Forschung im Mittelpunkt. Gegenstand der Ausführungen ist ein langfristiges Projekt zur Schaffung einer Analyseplattform für inflammatorische und Streßantwort-assozierte Gene auf der Grundlage eines eigens entwickelten Spotted Arrays (vgl. Abschnitt 1.3 auf Seite 8 im Einführungsteil I der Arbeit). Die Entwicklung erfolgte in Kooperation mit der Transfusionsmedizin der Universität Tübingen, inbesondere mit Dr. Derek Zieker. Die Transfusionsmedizin war für die gerätetechnische, experimentelle und medizinisch-praktische Seite zuständig, während die Aufgabe des Autors dieser Arbeit darin bestand adäquate Daten-organisatorische, bioinformatische und mathematisch-statistische Methoden für eine valide Auswertung der Plattform zu entwickeln und anzuwenden.

Das gemeinsame Projekt begann bereits im Jahr 2004 mit den Planungen der Voruntersuchungen zur Validierung des ersten Entwurfs der Plattform (Kapitel 7 auf Seite 155) und den Planungen zu den ersten beiden großen Studien, die mit Hilfe dieser Plattform untersucht werden sollten – die Marathonstudie [ZZD+05, ZFD+05, Zie06] (Kapitel 8 auf Seite 165) und die Untersuchung von *Posttraumatic Stress Disorder* (PTSD)-Patienten [ZZJ+07, Zie07] (Kapitel 9 auf Seite 177). Diese erste Stufe des Entwicklungsprozesses konnte erfolgreich durchgeführt und abgeschlossen werden, wofür die neu erzielten, veröffentlichten Erkenntnisse zu den einzelnen Studien einen eindeutigen Beleg abgeben.

Die bei dieser praktischen Arbeit gemachten Erfahrungen des Autors hatten wichtige, bereichernde Auswirkungen auf die Entwicklungen der in Teil II beschriebenen Applikationen und Verfahren, konkret sei hier auf die Abschnitte 6.5.2 auf Seite 136 und 6.5.3 auf Seite 140 verwiesen, wobei im letzt genannten Abschnitt die Qualitätstests einer stark erweiterten neuen Entwicklungsstufe des Microarrays dargestellt wurden.

Neben den in dieser Arbeit erwähnten, abgeschlossenen Studien wurden eine Anzahl weitere Studien zur Genexpression mit dieser Plattform erfolgreich durchgeführt, teilweise noch mit dem in dieser Arbeit beschriebenen Entwicklungsstand, zum Beispiel zu zystischer Fibrose, zu Sepsis und zum Trainingszustand von Radrennfahrern. Mittlerweile gibt es eine neue Version, die um eine Reihe onkologisch relevanter Gene erweitert und bereits erfolgreich eingesetzt wurde [ZKT+08, Kna07, Sch07]. Diese Entwicklungsstufe fand nur in der oben erwähnten Form in Abschnitt 6.5.3 Eingang in die vorliegende Arbeit. Die mit diesem neuen Array erzielten Ergebnisse sind Ausdruck der erfolgversprechenden weiteren Entwicklung der Plattform jenseits des ursprünglichen Anwendungsbereichs.

7 Die Etablierung einer inflammatorischen, home-made c-DNA-Plattform für die medizinische Forschung

> The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data.
>
> *(J.W. Tukey)*

In enger Kooperation mit der Abteilung der Transfusionsmedizin der Universität Tübingen wurde ein Spotted cDNA-Array für die Gene inflammatorischer Faktoren entwickelt und für die medizinische Forschung etabliert. Hierbei bestand der Anteil, der in dieser Arbeit beschrieben werden soll, sowohl in der statistischen Evaluierung des Arrays, wie auch in einer statistisch fundierten Planung der durchgeführten Studien sowie der Auswahl geeigneter Verfahren für die Analyse und die Verwaltung der gewonnenen Daten.

Das entwickelte Array wurde im Vorfeld der mit diesem Array geplanten ersten Studien einer Evaluierung unterzogen. Entsprechend den daraus resultierenden Erfahrungen folgte dann der Einsatz in den beiden Studien:

- Expressionverhalten inflammatorischer Gene nach intensiver Ausdauerbelastung (beschrieben in Kapitel 8, sowie [ZZD+05, ZFD+05, Zie06])

- Genexpressionsunterschiede im peripheren Blut von Posttrauma-Patienten (beschrieben in Kapitel 9, sowie [ZZJ+07, Zie07])

Aufgrund der bereits guten und umfangreichen Vorstellung der Aspekte dieser Array-Plattform und ihrer Anwendung in den erwähnten Veröffentlichungen ist die Darstellung im Rahmen dieser Arbeit sehr gestrafft und nur so weit ausgeführt, wie für das grundlegende Verständnis notwendig. Speziell für die Einzelheiten rund um den labortechnischen Ablauf und die verwendeten Protokolle sei auf die umfassenden Ausführungen hierzu in [Zie06] verwiesen.

7.1 Eckdaten des inflammatorischen Arrays

Auch für die Plattform eines inflammatorsichen Genexpressionsarrays waren die in Abschnitt 1.2 ab Seite 6 skizzierten generellen Überlegungen und Fragen zur Durchführung eines Array-basierten Genexpressionsexperiments zu beantworten.

7.1.1 Technologische Grundlage

Aus der Zielrichtung der ganzen Plattform heraus auf die Untersuchung der Genexpression inflammatorischer, immunologisch entzündungsrelevanter Gene stand bereits ein begrenzter Kanon interessierender Gene fest. Rund 300 verschiedene Gene oder ESTs betrachtete die Transfusionsmedizin Tübingen als interessant bzw. als interessierend und damit als relevant für das Array. Die Liste der Gene ist ab Seite 92 in [Zie06] zu finden.

Die moderate Anzahl an zu untersuchenden Genen, kombiniert mit der Vorgabe zu einer möglichst kostengünstiger Produktion sowie einem kostengünstigen Einsatz der Plattform innerhalb von medizinischen Studien, legten als technologische Grundlage die Verwendung eines Spotted Arrays vor (vgl. Abschn. 1.3 auf Seite 8).

7.1.2 Sondenmaterial

Für das Sondenmaterial kommen bei Spotted Arrays entweder lange cDNA-Fragmente ($\sim 500 \ldots 1500$ b) oder wesentlich kürzere Long-Oligos ($\sim 50 \ldots 80$ b) zum Einsatz. Die Verwendung von Long-Oligos setzt allerdings ein genaues Design dieser Oligos voraus, d.h. die Auswahl des bestgeeignetsten Sequenzabschnittes innerhalb des Targetgens oder Target-ESTs, um eine ausreichend hohe Spezifität für diese Target sicherzustellen. Dabei sind vor allem mögliche Kreuzhybridisierungen zu verhindern. Bei langen cDNA-Fragmenten ist dieser Schritt nicht so aufwendig, da sie, aufgrund ihrer Länge, automatisch sehr Target-spezifisch sind.

In der für diese Arbeit relevanten, ersten Entwicklungsstufe des Arrays fiel die Entscheidung zugunsten der einfacheren Verwendung von cDNA-Fragmenten aus. Geeignete bakterielle Klone für die oben erwähnten vorgesehenen Gene wurden vom damaligen *Deutschen Ressourcenzentrum für Genomforschung GmbH* (RZPD) bezogen.

Das Sondenmaterial selbst wurde als *Polymerase Chain Reaction* (PCR)-Amplifikat von 300 humanen cDNA-Inserts bakterieller Klone aus 300 je 50 μl PCR-Ansätzen einer 384-Well Mikrotiterplatte gewonnen.

7.1.3 Trägermaterial

Üblicherweise bilden oberflächenbehandelte Glasträger das Trägermaterial von gespotteten Microarrays. Aufgrund der Entscheidung für längere cDNA-Fragmente als Sondenmaterial kamen Glasträger mit amino-modifizierter Oberfläche zum Einsatz. Die Beschichtung mit polaren Amino-Gruppen verstärkt bzw. verbessert die

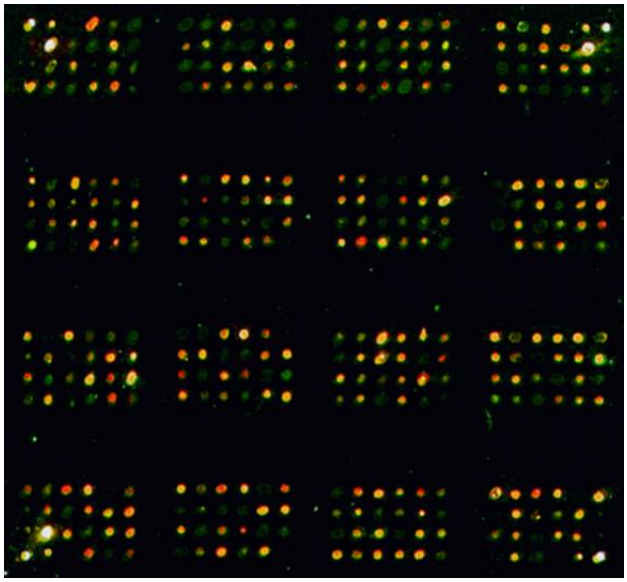


Abbildung 7.1: Typisches Ergebnis eines gescannten hybridisierten inflammatorischen Arrays. (Quelle [Zie06])

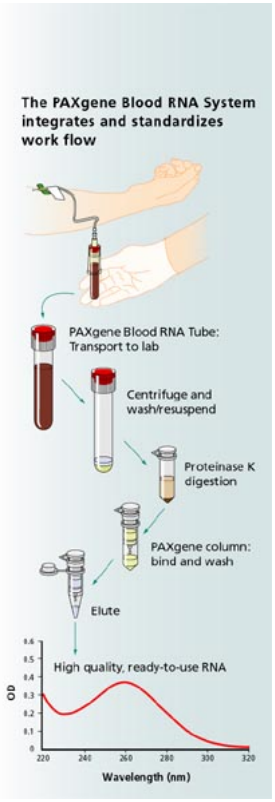


Abbildung 7.2: RNA-Gewinnung aus Vollblut mit dem PAXgene Blood RNA Kit von Qiagen (Quelle [Pre08])

Wechselwirkung mit den geladenen Gruppen des DNA-Phosphatanteils. Der Bindungsvorgang wird zusätzlich durch eine geeignete Behandlung mit UV-Licht und Wärme unterstützt.

7.1.4 Das Spottingverfahren

Für das Spotten von Microarrays, also das Aufbringen des Sondenmaterials auf dem Träger des Arrays, wird ein rechnergesteuerter Spottingroboter verwendet - der sogenannte *Arrayer*. Dieser überträgt das Sondenmaterial aus den 384-Well Mikrotiterplatten auf den Glasträger. Dabei sind zwei mechanische Varianten möglich – eine berührungslose, die das Sondenmaterial nach dem Inkjet-Prinzip aus Düsen auf die Oberfläche sprizt oder eine, die durch den Kontakt von Nadeln, sogenannten Pins, mit dem Trägermaterial die Sondenmoleküle auf dem Träger plaziert. Die Nadeln sind in Printtips organisiert, deren Organisationsstruktur sich zumeist im Layout des Arrays widerspiegelt.

Bei den Nadeln sind Formen zu unterscheiden, bei denen ein größerer Vorrat an Material mitgeführt wird, z.B. bei „Split-Pins“ in einer Kapillare der Nadel oder

nur der durch Adhäsion nach dem Eintauchen anhaftetende, wie bei „Solid-Pins“. Letzteres begrenzt die Menge an Arrays, die nach einem einmaligen Eintauchen gespottet werden können und beschränkt somit den Durchsatz.

Die Arrays, die in den für diese Arbeit relevanten Studien zum Einsatz kamen, wurden mit Hilfe von Solid-Pins im Kontaktverfahren gespottet. Zusätzlich zu den Sonden der eigentlich interessierenden Gene wurden cDNA-Kontrollen der Firma Stratagene, sowie Negativ-Kontrollen in Form von Leerspots und reinem Spottingpuffer gespottet. Ein solches komplettes Printtip („384er-Layout“) wurde dreimal auf einen einzelnen Glasträger gespottet, so dass jede Sonde als Triplikat vorlag. Die Eckpunkte des Printtips wurden mit Universal-cDNA von Stratagene markiert. Abbildung 7.1 zeigt ein Printtip des so gespotteten Chips nach der Hybridisierung und dem Scannen.

7.2 Kompletter Ablauf eines Experiments

7.2.1 Probenentnahme und Aufbereitung des Targetmaterials

Die Gewinnung qualitativ hochwertigen Targetmaterials ist für die Durchführung von Microarrayexperimenten von entscheidender Bedeutung. RNA ist, im Gegensatz zu DNA, nicht sehr stabil. Insbesondere mRNA wird sehr schnell durch *Ribonukleasen* (RNase) zersetzt. RNase ist allgemein weit verbreitet und es ist schwierig absolut RNase-freie Umgebungen zu erzeugen. Somit ist eine schnelle und saubere Arbeitsweise unbedingt notwendig. Aus der Gesamt-RNA einer Zelle wird mRNA mit Hilfe sogenannter Oligo-dT-Primer in stabilere cDNA umgeschrieben.

Da für die geplanten Studien die Gewinnung von RNA aus Vollblut vorgesehen war, erfolgte die Evaluierung zweier spezieller Verfahren im Hinblick darauf [Zie06]. Dabei setzte sich die RNA-Isolation mit dem PAXgene Blood RNA Kit von Qiagen [Pre08] (siehe Abbildung 7.2) aufgrund der damit erzielten guten Qualitätswerte durch [Zie06]. Liegt nicht genügend Probenmaterial vor, so ist eine vorhergehende linearen Amplifikation per PCR notwendig.

Nach der Reinigung und Aufkonzentration der gewonnenen cDNA erfolgt die Markierung mit den Fluoreszenzfarbstoffen Cyanine-3 und Cyanine-5 (Cy3 und Cy5) von Amersham Bioscience. Cy3 wird mit Licht von $\sim$ 550 nm angeregt und fluoresziert mit einem Maximum bei einer Linie von $\sim$ 570 nm (helles Rot), während Cy5 bei $\sim$ 650 nm angeregt wird und sein Emissionsmaximum bei $\sim$ 670 nm (dunkles Rot) besitzt.

7.2.2 Hybridisierung des Arrays

Da das Sondenmaterial auf dem Array noch doppelsträngig vorliegt, müssen diese Bindungen in einer entsprechenden chemischen und thermischen Vorbehandlung aufgebrochen werden. Anschließend erfolgt die Trocknung der Chips, die dann für die Hybridisierung vorbereitet sind.

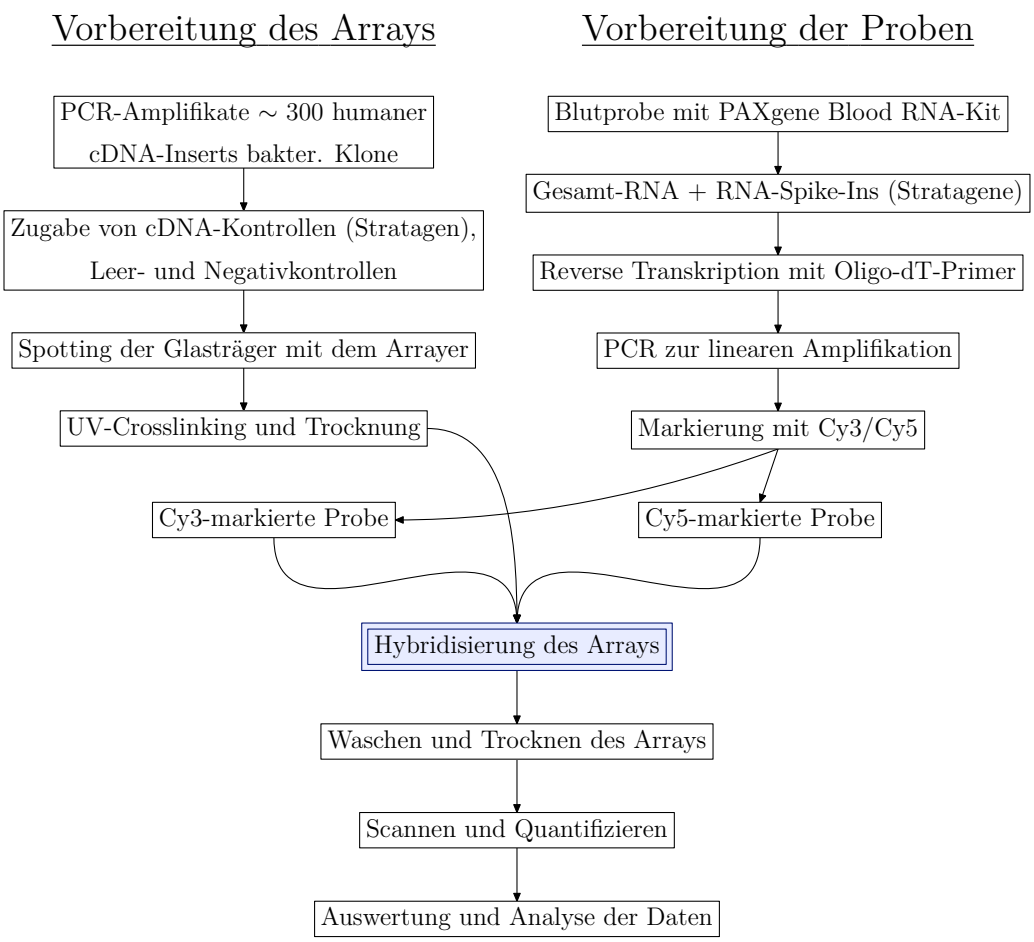


Abbildung 7.3: Kompletter Ablauf des experimentellen Teils einer Genexpressionsanalyse mit dem inflammatorischen Array

Die Hybridisierung der fluoreszenzfarbstoffmarkierten Proben mit dem Array erfolgt in einer wassergefüllten Hybridisierungskammer. Nach einer Inkubationszeit von ca. 14 Stunden bei einer Wassertemperatur von $\sim 55^{o}$C ist die Hybridisierung abgeschlossen und das Array kann gewaschen werden.

7.2.3 Gewinnung der Rohdaten des Arrays

Nach dem Waschen muss das Array schließlich quantifiziert werden, d.h. die Fluoreszenzintensitäten der verschiedenen Kanäle (Cy3 $\mapsto$ 570 nm und Cy5 $\mapsto$ 670 nm) werden ausgelesen und quantifiziert. Üblicherweise werden die beiden Farbkanäle einer zweikanaligen Genexpressionsanalyse mit Cy3 und Cy5 in der Form Cy3 $\mapsto$ Grün und Cy5 $\mapsto$ Rot kodiert. Die Intensitätsdaten beider Kanäle stehen dann als „Grün-“ bzw. „Rotstufenbilder“ (oft im 16-bit TIFF-Format) für die weitere Analyse zur Verfügung.

Die sich anschließende Bildanalyse der ausgelesenen Bilder zur Spotdetektion und

Spotquantifizierung wurde mit ImaGene 5 der Firma BioDiscovery durchgeführt.

7.2.4 Verwaltung der Rohdaten

ImaGene liefert neben einem XML-basierten auch ein ASCII-basiertes Dateiformat, um die mit ihm erhobenen Rohdaten auszuwerten. Für die Verwaltung der Rohdaten wurde eine spezifisch auf diese Plattform zugeschnittene SQL-Datenbank entworfen und realisiert.

Applikationsseitig kamen dabei Ruby-Skripte zum Parsen der ImaGene-Datendateien und dem Befüllen der Datenbank zum Einsatz. Als DBMS fiel die Wahl auf PostgreSQL [Pos07], da es einerseits frei verfügbar ist und eine umfangreiche Implementierung des SQL-Standards bietet und andererseits eine gute Anbindung an die statistische Programmierumgebung R [R D07] besitzt. R war von Anfang an als statistisch analytische Auswertungsplattform vorgesehen.

Die Datenbank selbst wird in dieser Arbeit nicht weiter vorgestellt, da sie mittlerweile weiterentwickelt und an die neuen Entwicklungstufen des inflammatorischen Arrays im Rahmen der Arbeit [Kna07] angepasst wurde.

7.2.5 Normalisierung

Die für die Auswertung eines Microarrays notwendige Normalisierung wurde in drei Schritten durchgeführt (siehe Abbildung 7.4). Um einen potentiell vorhandenen lokalen Bias der Intensitätswerte zu verringern, erfolgte eine separate *Locally Weighted Scatterplot Smoother* (LOWESS)- bzw. LOESS-Korrektur der gemessenen Intensitäten für jedes einzelne Subarray [BIÅS03, Cle79, YBDS02]. Die so behandelten Intensitätswerte des grünen (Ch^{green}) und roten Kanals (Ch^{red}) dienten als Ausgangspunkt, um den Log-Ratio M und das logarithmierte geometrische Mittel der Intensitäten A für jeden einzelnen Spot zu berechnen [BIÅS03, DYCS02]:

$$M = \log_2 \left(\frac{Ch^{red}}{Ch^{green}} \right), A = \log_2 \sqrt{Ch^{red} * Ch^{green}} \tag{7.1}$$

Da es sich bei der gewählten Hybridisierungsvariante um ein Dye-Swap-Design handelte, wurden zwei Arrays hybridisiert. Auf dem einen Array befanden sich die Referenz im roten (r^{red}) und die Probe im grünen Kanal (s^{green}) und auf dem anderen Array wurde genau umgekehrt markiert (s^{red},r^{green}). Somit ergaben sich die zwei folgenden Schätzer für die Log-Ratios der einzelnen Spots

$$M^I = \log_2 \left(\frac{r^{red}}{s^{green}} \right), M^{II} = \log_2 \left(\frac{s^{red}}{r^{green}} \right). \tag{7.2}$$

Die Verwendung der logarithmierten *Fold Change*s (FC) hat verschiedene Gründe. So wirkt sich die logarithmische Transformation varianz- und symmetriestabilisierend auf die Verteilung der Spotintensitätswerte aus. Schiefsymmetrische Verteilungen werden in ihrer Schiefe reduziert [DHHR02, Spe00]. Außerdem ermöglicht

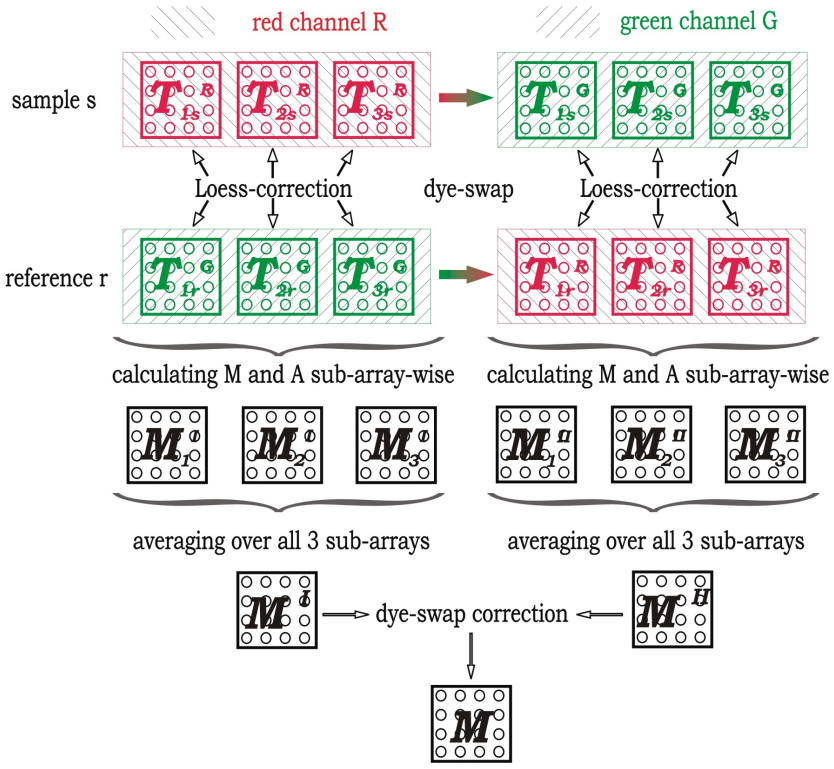


Abbildung 7.4: Ablaufdiagramm des Normalisierungsverfahrens für die Daten des inflammatorischen Arrays. Die Dye-Swap-Arrays sind in der oberen Reihe zu sehen. Auf der linken Seite befindet sich die zu untersuchende Probe im roten und die zugehörige Referenz im grünen Kanal. Auf der linken Seite sind die Kanäle entsprechend des Dye-Swap-Designs vertauscht. Nach einer *Locally Weighted Scatterplot Smoother* (LOWESS)- bzw. LOESS-Korrektur für jedes einzelne Subarray wurden die Log-Ratios M_x^I, M_x^{II} der Intensitäten der Probe und der Referenz für jeden einzelnen Spot berechnet. Die Werte für M^I, M^{II} und A^I, A^{II} waren das Ergebnis einer Mittelung über alle drei Subarrays hinweg. Schließlich wurde der endgültige Schätzer für M aus M^I und M^{II} mit Hilfe einer Dye-Swap-Korrektur gewonnen.

die Wahl der Basis 2 eine intuitive Interpretation von $M = \log_2(FC)$. Alle drei Replikate (Subarrays) einer Hybridisierung werden arithmetisch gemittelt, so dass für jedes Transkript jeweils zwei Schätzer vorliegen M^I, M^{II} und A^I, A^{II} (vergl. mit Bild 7.4). I und II kennzeichnen die beiden komplementären Dye-Konfigurationen. Als letzter Schritt der Normalisierung werden die Dye-Effekte mit Hilfe der beiden Schätzer M^I und M^{II} nach der Formel 7.3 herausgerechnet [SCPT+04, TOR+01, YDLS01].

$$\begin{aligned} M &= \log_2\left(\frac{s}{r}\right) \cong \frac{1}{2}(M^I - M^{II}) \qquad (7.3) \\ &= \frac{1}{2}\left[\log_2\left(\frac{r^{red}}{s^{green}}\right) - \log_2\left(\frac{s^{red}}{r^{green}}\right)\right] = \frac{1}{2}\left[\log_2\left(\frac{s^{red}}{r^{red}}\right) + \log_2\left(\frac{s^{green}}{r^{green}}\right)\right] \end{aligned}$$

2304	100%	Spots auf A und B insgesamt
1399	∼ 60.72%	Spots ohne Flag
188	∼ 8.16%	Empty- & Negative-Spots
717	∼ 31.12%	Spots mit gesetztem Flag-Code

Tabelle 7.1: Spot-Statistik für den Vorversuch des inflammatorischen Arrays. Übersicht über alle vorhandene Spots auf den beiden Dye-Swap-Arrays mit jeweils 3 Subarrays von 384 Spots (2∗3∗384 = 2304). Die nominell recht hohe Fehlerquote von 31.12 % wird wesentlich durch die Replikate auf dem gleichen Array (Triplikate in den Subarrays) und zusätzlich durch das Replikat aus dem Dye-Swap abgemildert, die für fehlerbehaftete, entprechend geflagte Spots des einen Subarrays bzw. Arrays einen korrekten Spot aufweisen können.

7.3 Erste Evaluierung des inflammatorischen Arrays

7.3.1 Self-Self-Experimente

Die Evaluierung des Arrays erfolgte in mehreren Stufen. Im Selbstversuch wurden Blutproben entnommen und die Genexpressionsanalyse als Self-Self-Experiment durchgeführt [Zie06] (vgl. Abschnitt 6.5.3 auf Seite 140 für die Evaluierung einer weiter entwickelten Stufe des Arrays). Die Beurteilung dieser Ergebnisse erfolgte während jeder einzelnen Stufe des experimentellen Teils der Analyse, sowie anhand der gescannten Images und deren Auswertung in ImaGene (siehe [Zie06]). Insbesondere die Beurteilung der Spotmorphologie und der Signalverteilung im Hintergrund erfolgte in ImaGene.

7.3.2 Stimuliert vs. Unstimuliert

In einem weiteren Versuch wurden *Lipopolysaccharid* (LPS)-stimulierte Blutproben der selben Person mit ihren unstimulierten verglichen. Die LPS-Stimulation erfolgte 2 Stunden lang [Zie06]. Der Versuch wurde in einem direkten Design, d.h. beide Proben werden in den beiden Kanälen auf einem Array direkt innerhalb einer Hybridisierung verglichen und als Dye-Swap durchgeführt (vgl. hierzu Abbildung 7.4).

Nach dem Scannen und der bildanalytischen Aus- und Bewertung mit ImaGene waren die in Tabelle 7.1 angegebenen Qualitätsanteile an den insgesamt vorhandenen Spots zu verzeichnen. Die nominell hohe Fehlerquote von ∼ 31.12 % Spots wird im weiteren Auswertungsablauf durch die Verrechnung der Triplikate auf dem Array und die Verrechnung mit dem zweiten Replikat aus dem Dye-Swap abgemildert.

Für die Gewinnung der Intensitätsrohdaten wurden die gleichen Schätzer benutzt wie in Abschnitt 8.3.1 auf Seite 167 - infolge der Gründe, die in Abschnitt 8.3.1 erläutert werden.

Abbildung 7.5 zeigt schrittweise die Auswertung der Voruntersuchung. Dabei lagen zwei hybridisierte Arrays A und B vor, bei denen es sich um ein Dye-Swap-Paar

der selben Probe handelte. Für die Normalisierung kam das dreistufige Verfahren zur Anwendung, das in Abschnitt 7.2.5 erläutert wurde:

1. die Subarray-bezogenen LOWESS-Korrektur, (obere Reihe von Abbildung 7.5 für die Subarrays jeweils von `A` und `B`)

2. die anschließenden Mittelung über die Subarrays hinweg (mittlere Reihe von Abbildung 7.5 für Array `A` und `B`)

3. und der darauffolgenden Dye-Swap-Korrektur entsprechend Gleichung 7.3. (Bild unten links von Abbildung 7.5)

M und A wurden entsprechend der Definition von Gleichung 7.1 bestimmt und für Schritt 3 Gleichung 7.3.

Bei den Voruntersuchungen war lediglich eine überblicksartige Einschätzung der Signaleigenschaften der Array-Plattform notwendig und nicht das konkrete Ergebnis einzelner differentiell exprimierter Gene, so dass eine *Slice-Analyse* zum Einsatz kam. Die Slice-Analyse stellt eine intensitätsabhängige z-Transformation (Transformation zur Standardnormalverteilung) von M_S und A_S der Spots dar. Dabei wird, ähnlich einer Kernelmethode, die Intensitätsachse A schrittweise abgelaufen und für jedes einzelne Fenster h_i ($\widehat{=}$ Slice) die M_S-bezogenen Schätzer für $\hat{\mu}_{h_i}$ und $\hat{\sigma}_{h_i}$ bestimmt. Die Fenstergröße von h_i ist konstant und wird vorher festgelegt.

$$\forall M_S \text{ mit } A_S \in h_i : \quad z(M_S) = \frac{M_S - \hat{\mu}_{h_i}}{\hat{\sigma}_{h_i}} \tag{7.4}$$

Interessant sind dabei besonders die ersten ganzzahligen Stufen der Standardnormalverteilung:

$$\begin{aligned} P(|Z| \leq 1) &\sim 68.3\,\% \\ P(|Z| \leq 2) &\sim 95.5\,\% \\ P(|Z| \leq 3) &\sim 99.7\,\% \end{aligned} \tag{7.5}$$

Bild unten rechts von Abbildung 7.5 zeigt das Ergebnis der Slice-Analyse in einer farbkodierten Form entsprechend den Stufen von Gleichung 7.5. Dabei wird ersichtlich, dass nur eine geringe Anzahl von Genen blau markiert ist und damit im Bereich einer signifikanten Abweichung („p-Value“$< 100\% - 95.5\% = 4.5\%$, wenn man das unterstellte Modell der Normalverteilung annimmt) des Expressionsniveaus zwischen LPS-stimuliertem und unstimuliertem Blut liegt.

Insgesamt war damit, als Ergebnis der Evaluationsphase, der Nachweis eines reproduzierbaren Messens mit Hilfe der Arrayplattform erreicht. Dabei konnte die Einhaltung eines akzeptablen „Signal-Rausch-Abstands“ des Array-basierten Messvorgangs verzeichnet werden.

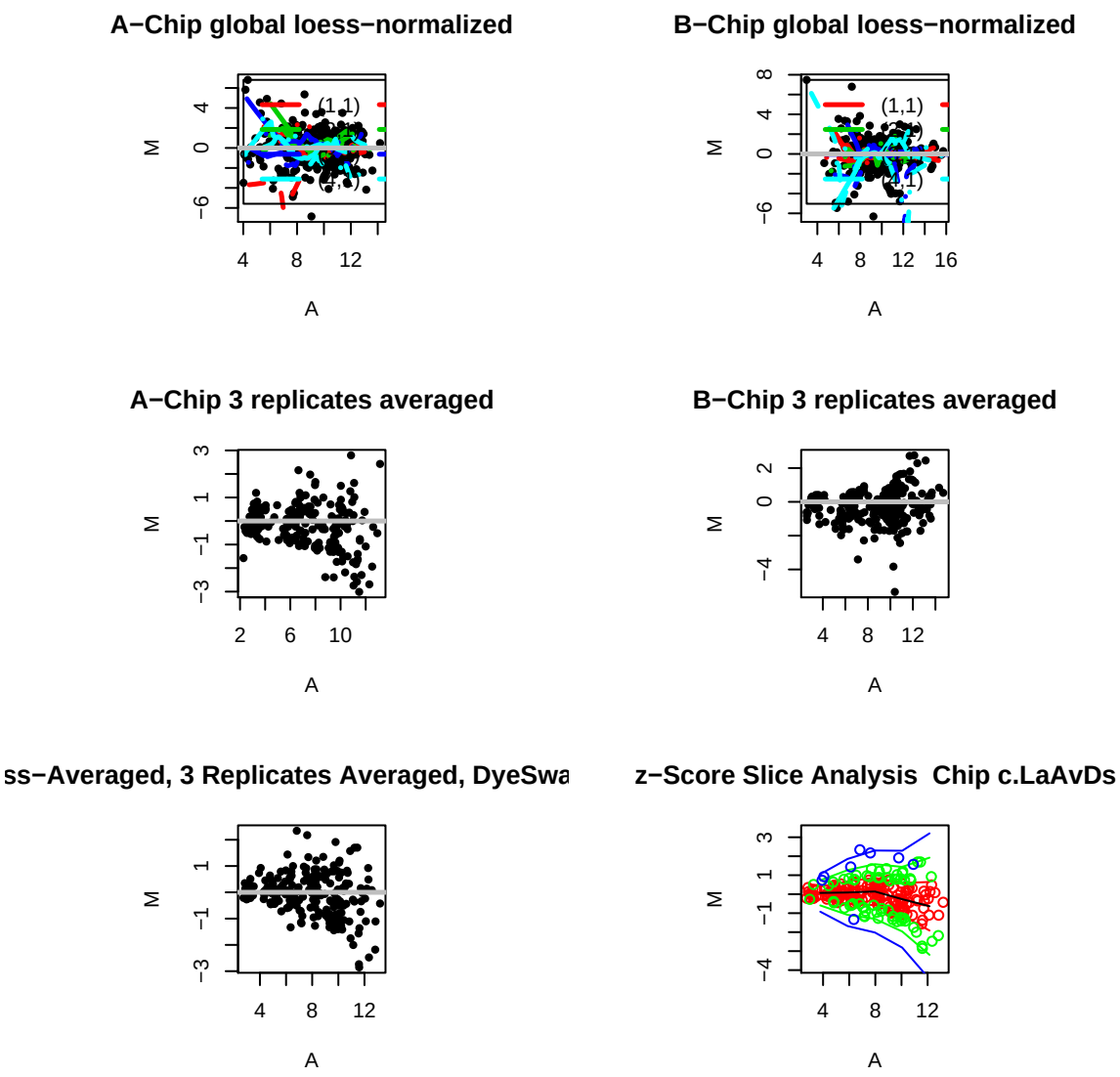


Abbildung 7.5: Auswertung der Voruntersuchung bestehend aus einer Dye-Swap-Hybridisierung (Array `A` und `B`) von LPS-stimuliertem und unstimuliertem Blut. Alle Grafiken tragen M gegenüber A ab (Definition von M und A entsprechend den Gleichungen 7.1). In der oberen Reihe ist eine Subarray-bezogene LOWESS-Korrektur für beide Arrays dargestellt (siehe Text und Abschnitt 7.2.5). Die mittlere Reihe zeigt das Ergebnis nach einer Mittelung über alle Triplikate hinweg. Das Bild links unten zeigt das Resultat nach der Dye-Swap-Korrektur enstprechend Formel 7.3. Rechts unten ist schließlich das Ergebnis einer intensitätsabhängigen z-Transformation bzw. einer Slice-Analyse zu sehen. Dabei bedeutet die Farbkodierung: $|M_S| \leq \hat{\sigma}_{h_i} \mapsto$ Rot, $\hat{\sigma}_{h_i} < |M_S| \leq 2\hat{\sigma}_{h_i} \mapsto$ Grün und $2\hat{\sigma}_{h_i} < |M_S| \mapsto$ Blau

8 Expressionsverhalten inflammatorischer Gene nach intensiver Ausdauerbelastung

> While nothing is more uncertain than a single life, nothing is more certain than the average duration of a thousand lives.
>
> *(E. Wright)*

In einer weitergefassten Kooperation der bereits beteiligten Kooperationspartner mit der Sportmedizin Tübingen wurde ein Einsatz des inflammatorischen Arrays für die Sportmedizin vorgesehen.

Sportliche Belastungen stehen bekanntermaßen im Zusammenhang mit einer Beeinflussung des Immunsystems. Insbesondere eine hohe Trainings- und Wettkampfbelastung provoziert wiederholt eine Immun- bzw. Stressantwort des Immunsystems, das eine zumindest zeitweise Schwächung desselben bedeutet. Im Gegensatz dazu gelten moderate und regelmäßige Ausdauerbelastungen als eher förderlich [NNC94]. Dabei ist eine Reaktion aller Organsysteme auf die sportliche Belastung zu beobachten [Ber03]. Für die Untersuchung in einem systemischen Ansatz erschien gerade die Untersuchung von Vollblut, für die die Array-Plattform ausgelegt und bereits validiert war, als sehr geeignet.

8.1 Studienentwurf

Primäres Ziel der Studie war das grundsätzliche Auffinden von Zusammenhängen zwischen einer unmittelbaren starken sportlichen Ausdauerbelastung und zeitnahen Veränderungen in der Expression inflammatorischer Gene [ZZD$^+$05]. Als sekundäres Ziel galt die Extraktion belastungscharakteristischer Expressionsprofile mehrerer inflammatorischer Gene [ZFD$^+$05]. Als geradezu prototypisch für hohe sportliche Ausdauerbelastung gilt der Marathonlauf und deshalb wurde das zu untersuchende Kollektiv aus dieser Sportart ausgewählt.

Zielgemäß war ein Vergleich der Genexpression unmittelbar nach einer großen sportlichen Ausdauerbelastung im Vergleich zur Ruhephase davor und danach vorgesehen.

8.1.1 Probandenkollektiv

Das Probandenkollektiv wurde aus 8 trainierten männlichen Marathonläufer zusammengestellt. Das Durchschnittsalter der Läufer betrug 38.9 ± 11.8 Jahre und ihr durchschnittlicher BMI 23.6 ± 1.8. Alle Läufer nahmen an dem gleichen öffentlichen Halbmarathon (21.1 km) teil, der als unmittelbare sportliche Ausdauerbelastung diente. Der Halbmarathon wurde an einem kühlen Dezembertag (1° C) gelaufen und führte über eine steigungsreiche anspruchsvolle Strecke.

Die Läufer befanden sich seit mindestens 2 Jahren im Ausdauertraining mit einer durchschnittlichen wöchentlichen Laufleistung von 52.2 ± 25.5 km. Das Probandenkollektiv wurde auch im Hinblick auf ihren aktuellen gesundheitlichen Zustand untersucht. Keiner der Beteiligten litt unter einer akuten oder chronischen Erkrankung. Auch die Einnahme von Medikamenten, sowie Antioxidantien und Nikotinmissbrauch wurde vorher ausgeschlossen, um eine Beeinflussung der Ergebnisse dadurch zu verhindern.

8.1.2 Probengewinnung

Die Probenentnahme erfolgte in Form von 5 mal 2.5 ml Vollblut, das zur weiteren Verarbeitung sofort in PAXgene Blood RNA Röhrchen von Qiagen abgefüllt wurde (siehe Abschnitt 7.2.1 auf Seite 158). Die Probenentnahme wurde entsprechend der Zielstellung zu den folgenden drei unterschiedlichen Zeitpunkten vorgenommen:

- t_0: 24 Stunden vor dem Lauf, in Ruhe
- t_1: unmittelbar nach Abschluss des Laufs (bis maximal 15 Minuten nach dem Lauf)
- t_2: 24 Stunden nach Abschluss des Laufes, in Ruhe

8.2 Hybridisierungsstrategie

Für die Hybridisierung empfahl sich eine direkte Strategie bei der der Zeitpunkt t_0 als „bedeutungsvolle“ Referenz innerhalb eines jeden Probanden i verwendbar war (siehe Abbildung 8.1):

- Hybridisierung von $t_1^i \Leftrightarrow t_0^i$
- Hybridisierung von $t_2^i \Leftrightarrow t_0^i$

Der Vorteil einer direkten Strategie mit einem Probenvergleich auf demselben Array besteht gegenüber einer indirekten darin, eine Reihe von Zufallsfehlern im statistischen Sinne blocken zu können (Blockdesign) und deshalb eine geringere Streuung aufzuweisen. Der bestehenden Nachteil der hohen Inflexibilität gegenüber später aufgeworfenen Fragestellungen, Erweiterungen oder Neukombination schien im Zusammenhang mit dieser Studie nicht sehr relevant. Alle „Zeitvergleiche“ waren

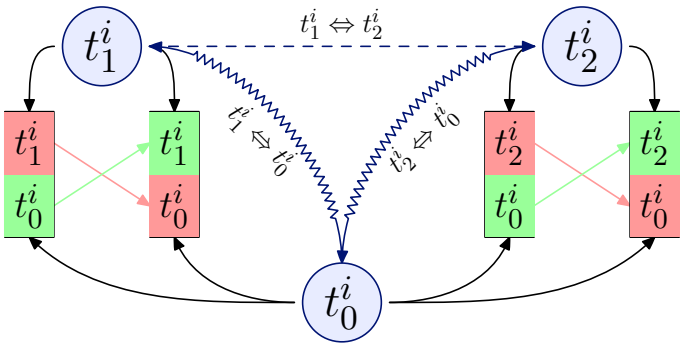


Abbildung 8.1: Da die einzelnen Zeitpunkte t_0, t_1 und t_2 intraindividuell verglichen werden müssen, reicht eine Beschränkung der Darstellung auf den jeweiligen Probanden i. Die Vergleiche $t_1^i \Leftrightarrow t_0^i$ und $t_2^i \Leftrightarrow t_0^i$ werden direkt auf zwei Dye-Swap-Arrays vorgenommen und können so bei minimaler Varianz ermittelt werden. Der Vergleich $t_1^i \Leftrightarrow t_2^i$ muss im Gegensatz dazu rechnerisch auf der Grundlage von t_0^i als gemeinsame Referenz ermittelt werden und würde deshalb ein größere Varianz besitzen.

intraindividuell zu ziehen und eine spätere Erweiterung der Fragestellungen mit neuen Daten hätte deshalb auch das Heranziehen der gleichen Personen bedeutet. Das Probandenkollektiv war somit geschlossen und die Fragestellung nach den intraindividuellen Genexpressionsunterschieden zwischen den Zeitpunkten t_0, t_1 und t_2 sehr klar umrissen.

Der Vergleich $t_1^i \Leftrightarrow t_2^i$, der hätte rechnerisch ermittelt werden müssen, stellte in der Planung eine untergeordnete Option dar. Da er nicht notwendig wird, wenn der Vergleich $t_2^i \Leftrightarrow t_0^i$ bereits eine wesentliche Übereinstimmung der beiden „Ruhezeitpunkte“ t_0 und t_2 ergibt, wofür es Anhaltspunkte gab.

8.3 Statistische Auswertung

Insgesamt wurden damit für 8 Probanden mit jeweils 2 Zeitvergleichen ($t_1^i \Leftrightarrow t_0^i$, $t_2^i \Leftrightarrow t_0^i$) 2 Hybridisierungen im Dye-Swap durchgeführt, d.h. es lagen $8 * 2 * 2 = 32$ Arrays für die Auswertung vor. Dabei ist zu beachten, dass Subgrids dieser Arrays aufgrund von durchgehend schlechter Qualität gar nicht erst Eingang in die Auswertung fanden. Dies umfasste bei dieser Studie 3 von 96 Subgrids. Die fehlenden Subgrids verteilten sich entsprechend, so dass die Information durch die Replikate dieser Subgrids verfügbar waren.

8.3.1 Gewinnung der Intensitätsrohdaten

Die Arrays wurden mit einem Scanner der Firma Genetix Limited, Hamphshire, UK eingescannt und anschließend mit ImaGene 5 (BioDiscovery) für die Bildbearbeitung ausgewertet. Für die nachfolgende Analyse wurden die Daten aller Spots verworfen, die von ImaGene 5 als *Poor* markiert wurden. Spots erhalten

35712	100%	verfügbare Spots für die Auswertung auf allen Arrays
29448	$\sim$ 82.46%	Spots ohne Flag
754	$\sim$ 2.11%	Empty- & Negative-Spots
5510	$\sim$ 15.43%	Spots mit gesetztem Flag-Code

Tabelle 8.1: Spot-Statistik für alle Arrays der Marathonstudie. Übersicht über alle vorhandenen Spots – 8 Probanden, 2 Zeitpunktvergleiche ($t_0 \Leftrightarrow t_1$, $t_0 \Leftrightarrow t_2$), 2 Dye-Swap-Arrays mit jeweils 3 Subarrays von 384 Spots, d.h. theoretisch $8*2*2*3*384 = 36864$ Spots. Da 3 Subarrays von zu schlechter Qualität waren, wurden sie bereits beim Scannen verworfen und nur 35712 Spots lagen überhaupt zur Auswertung vor. Die, im Vergleich zum Vorversuch (vgl. mit Tabelle 7.1) bessere Fehlerquote von 15.43 % wird auch hier noch durch die vorhandenen Replikate auf dem gleichen Array (Triplikate in den Subarrays) und dem Replikat auf dem Dye-Swap abgemildert, wie schon im Vorversuch in Tabelle 7.1 erläutert.

diese Markierung bei aufgetretener Kontamination des Vorder- oder Hintergrundsignals, bei irregulärer Form des Spots oder ganz allgemein bei einer detektierten schlechten Qualität der Spotsignale. Tabelle 8.1 gibt einen Überblick über die Qualitätsvalidierung nach der Bildverarbeitung aller für die Auswertung vorgesehener Spots.

Als ein robuster Schätzer für die Intensitätswerte $\hat{I}_{Ch^x}$ beider Kanäle Ch^x des Spots wurde jeweils der Modus der Verteilungen der Vorder- und Hintergrundpixel benutzt.

8.3.2 Normalisierung

Infolge einer gleichen Dye-Swap-Struktur der einzelnen Hybridisierungen, konnte die Normalisierung der Array-Daten vollkommen analog zu dem Normalisierungsverfahren erfolgen, das in Abschnitt 7.2.5 auf Seite 160 dargestellt und beschrieben wurde.

8.3.3 Detektion differentiell exprimierter Gene

Aufgrund des gewählten direkten Designs für die Hybridisierung, d.h. der Hybridisierung von t_0 als Referenz und t_1 sowie t_2 als Probe auf dem gleichen zweikanaligen Array, kennzeichnen die Log-Ratios M, die statistisch signifikant von Null abweichend sind, differentiell exprimierte Gene. Zur Filterung der Gene und Charakterisierung der statistischen Signifikanz ihrer Messwerte kam ein Ein-Stichproben-t-Test mit der Annahme $\mu = 0$ zur Anwendung [DYCS02] (vgl. Abschnitt 5.4 ab Seite 96). Das dabei auftretende Problem des multiplen Testens wurde durch die Verwendung verschiedener Methoden zur Kontrolle der *Familywise Error Rate* (FWER) und *False Discovery Rate* (FDR) angegangen [DYCS02, GDS03] (vgl. Abschnitt 6.6 ab Seite 146 und Abschnitt 6.5.2 ab Seite 136). Um eine möglichst hohe statis-

Gen	M	FC	p-Value (Bonf.-korr.)
CD 81 (cd81)	-0.9	0.5	0.00
CD 244 (cd244)	-0.6	0.7	0.01
Integrin Alpha X (itgax)	-0.2	0.9	0.02
Selectin L (sell)	1.7	3.2	0.04
Glutathione S-transferase M3 (gstm3)	-0.3	0.8	0.04
ICAM 2 (icam2)	-0.9	0.5	0.04

Tabelle 8.2: Übersicht der als signifikant differentiell exprimiert detektierte Gene ($p < 0.05$) des Vergleichs $t_1 \Leftrightarrow t_0$. Zur Anwendug kam dabei ein Einstichproben-t-Test mit der Annahme $\mu = 0$ und anschließender Bonferroni-Korrektur für multiples Testen über alle Gene hinweg. FC bezeichnet dabei den Fold Change FC $= \frac{Ch^{red}}{Ch^{green}}$ und M den Logratio und damit dessen Logarithmus $M = \log_2(\text{FC})$ (vgl. Gleichung 7.1).

tische Signifikanz für die Auswahl an differentiell exprimierten Genen zu erreichen, wurde dieser die Bonferroni-Korrektur [Bon36] zugrunde gelegt. Ein Gen wurde dann als differentiell exprimiert charakterisiert, wenn sein Bonferroni-korrigierter p-Value unter 5 % lag ($p - Value < 0.05$). Zusätzlich wurden auch die Korrekturverfahren nach Benjamini-Hochberg [BH95], Benjamini-Yekutieli [BY01], Holm [Hol79], sowie Sidak [Sid67] durchgeführt und deren Ergebnisse verglichen, um eine eventuell auftretende große Diskrepanz zwischen den einzelnen Verfahren nicht unberücksichtigt zu lassen. Derartig große Abweichungen ergaben sich jedoch nicht (vergleiche hierzu auch mit Abschnitt 6.5.2 auf Seite 136).

Als Ergebnis wurden für den Vergleich $t_1 \Leftrightarrow t_0$ 6 Gene als signifikant differentiell exprimiert zum Niveau $\alpha = 0.05$ detektiert (siehe Tabelle 8.2). Im Vergleich $t_2 \Leftrightarrow t_0$ erreichten zwei Gene die vordefinierte Signifikanzschwelle, die jedoch aus medizinisch biologischer Sicht nicht von großem Interesse waren.

Aufgrund der sehr konservativen Eigenschaften der Bonferroni-Korrektur, insbesondere bei sehr großem N, ist die Wahrscheinlichkeit für falsch positiv detektierte Gene zwar gering, aber dafür die für falsch negativ selektierte hoch. Aus diesem Grund wurden für die nachfolgende Validierung der Messergebnisse per Real-Time-PCR neben den 6 gefundenen Genen noch 4 weitere ausgewählt, deren p-Values zwar nicht die definierte Signifikanzschwelle unterschritten, die aber aus medizinisch biologischer Sicht sehr interessant waren.

In der Validierung durch die Real-Time-PCR bestätigten, bis auf CD 244 und ICAM 2, alle überprüften Gene eine Änderung und auch ihre Änderungsrichtung, die in der Microarrayauswertung ermittelt wurde (siehe Tabelle 8.3). Zur Beurteilung der Signifikanz wurde hier ein gepaarter Zweistichproben-t-Test verwendet. Eine genauere Betrachtung der Werteverteilung von CD 244 und ICAM 2 in den zugehörigen Boxplots der Abbildung 8.2 legt jedoch auch eine prinzipielle Bestätigung der Änderungsrichtung für diese beiden Gene nahe. Jedoch konnte die Nullhypothese hier nicht signifikant abgelehnt werden (siehe Tabelle 8.3).

Gen	p-Value
CD1c (cd1c)	0.03
CD244 (cd244)	0.27
CD81 (cd81)	0.02
Glutathione S-transferase M3 (gstm3)	0.02
ICAM2 (icam2)	0.57
IL 1 Receptor Antagonist (il-1ra)	0.0002
Integrin Alpha X (itgax)	0.03
Selectin L (sell)	0.004
Mapkap K2 (mapkapK2)	0.02
Thioredoxin (trx)	0.03

Tabelle 8.3: Ergebnis der validierenden Real-Time-PCR für die als differentiell exprimiert detektierten Gene im Vergleich $t_1 \Leftrightarrow t_0$ und 4 weiterer biologisch und medizinisch interessanter Gene. Die Ergebnisse der Real-Time-PCR wurden mit einem gepaarten Zweistichproben-t-Test analysiert.

8.3.4 Genexpression und Zellverschiebung

Ein allgemein bekanntes Problem bei der Benutzung von Vollblut zur Genexpressionsanalyse besteht darin, dass sich der Anteil der einzelnen Zellpopulationen zwischen den Zeitpunkten verschieben kann. Allgemein wird der Hinweis gegeben, möglichst homogenes Gewebe bzw. gezielt nur einzelne Zelltypen zu untersuchen. Bei Array-basierten Genexpressionsanalysen im Zusammenhang mit Verbrennungen wurde bereits darauf hingewiesen, dass festgestellte Änderungen möglicherweise mehr die Verschiebung der Anteile zwischen den verschiedenen Zelltypen reflektieren als eine grundsätzliche Änderung der Genexpression [SWF+04]. Aus diesem Grund wurden die Zusammensetzungen des Blutes mit erfasst (siehe Tabelle 8.4) und versucht, mit Hilfe eines linearen Modells den potentiellen Effekt der Zellverschiebung auf die Genexpressionsveränderung abzuschätzen.

Zeitpunkt	Leukozyten	Granulozyten	Lymphozyten	Monozyten
t_0	6237 ± 1520	3498 ± 939	2050 ± 565	445 ± 84
t_1	16825 ± 3930	14846 ± 3780	1140 ± 412	741 ± 335
t_2	6270 ± 1205	3682 ± 799	1865 ± 493	517 ± 134

Tabelle 8.4: Durchschnittliche Anzahl an Zellen des jeweiligen Typs in einem μl Blut für die verschiedenen Zeitpunkte.

In Tabelle 8.4 ist eine deutliche Zellverschiebung zwischen dem Zeitpunkt unmittelbar nach der Belastung t_1 und den beiden anderen t_0 und t_2 24 Stunden vor und nach der Belastung zu erkennen. Besonders Marker, die eng mit bestimmten Zelltypen in Verbindung stehen, sollten diese Verschiebung auch in ihren gemessenen

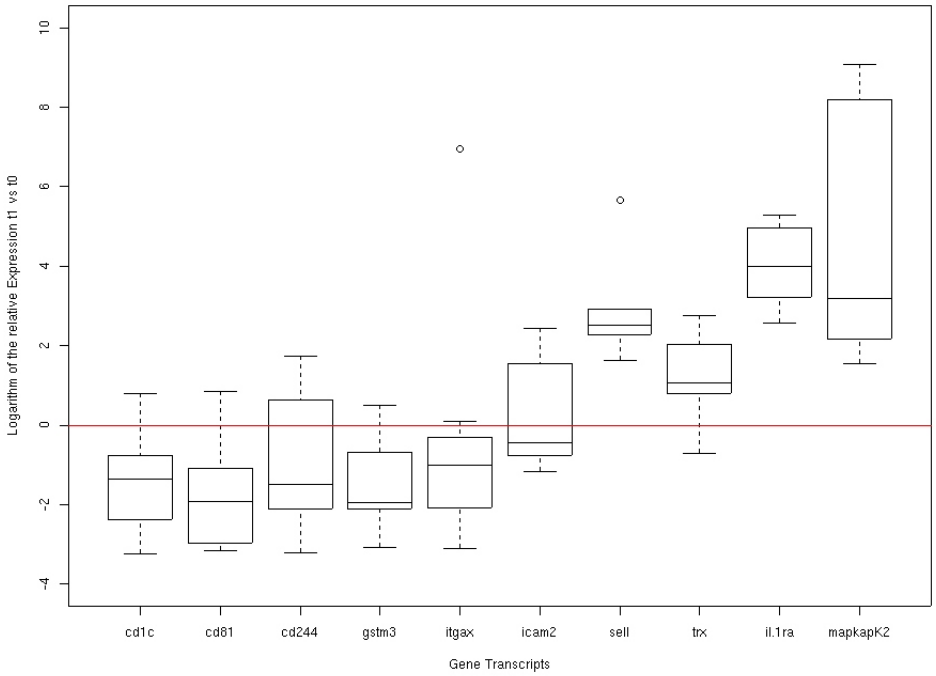


Abbildung 8.2: Boxplots für die Verteilung der durch Real-Time-PCR für den Vergleich $t_1 \Leftrightarrow t_0$ gewonnenen M_{rtPCR}-Werte. Für die nicht signifikanten Gene CD 244 und ICAM 2 ist zu erkennen, dass die Verteilungen schiefsymmetrisch sind und in ihrer statistischen „Verteilungsmasse“ die Analyseergebnisse des Arrays eher bestätigen.

Expressionswerten widerspiegeln. Tabelle 8.5 stellt einen aus den Zellpopulationen errechneten, für die Expressionswerte zu erwartenden FC dem in der Microarrayanalyse ermittelten FC des Gens gegenüber.

Die bereits in Tabelle 8.5 zu erkennende gute Korrelation der FCs aus der Expressionsanalyse und der Zellverschiebung wurde mit einem einfachen linearen Modell weiter untermauert, um die Relation beider Werte exakter zu spezifizieren:

$$\text{FC}_{Expr} \sim \text{FC}_{Zell} \tag{8.1}$$

Die hohe Korrelation zwischen beiden Werten wurde für das Modell

$$\text{FC}_{Expr} = 0.72 * \text{FC}_{Zell} + 0.37 \tag{8.2}$$

mit einem sehr hohen Bestimmtheitsmaß $R^2 = 0.97$ und einem sehr niedrigen, signifikanten p-Value von 0.0003 bestätigt. Insgesamt muss deshalb bei der Interpretation bzw. Diskussion der Ergebnisse die Zellverschiebung für zellabhängige Gene unbedingt berücksichtigt werden. Insbesondere Zell-assoziierte Gene deren Expression sich entgegen den durch die Zellverschiebung nahegelegten Werten ändern, verdienen eine besondere Beachtung bzw. Betrachtung, da hier entweder eine besonders

Marker	Zelltyp	FC (Zellverschiebung)	FC (Expression)
CD62L	Granulozyten	4.32375 ± 0.88021	$3.40987263 \pm 1.45615536$
CD14	Monozyten	1.6975 ± 0.72005456	1.887124 ± 0.52366769
CD1c	Lymphozyten	0.55875 ± 0.14456215	$0.81017089 \pm 0.07123074$
CD2	Lymphozyten	0.55875 ± 0.14456215	$0.47236313 \pm 0.18727041$
CD3e	Lymphozyten	0.55875 ± 0.14456215	$0.83402369 \pm 0.08292171$
CD19	Lymphozyten	0.55875 ± 0.14456215	0.78488713 ± 0.0926738

Tabelle 8.5: Gegenüberstellung der FCs aus der Zellverschiebung und der FCs von Markergenen aus der Array-basierten Expressionsanalyse für den Vergleich $t_1 \Leftrightarrow t_0$. Angegeben ist der Mittelwert $\pm$ die Standardabweichung für jeden Zelltyp und jedes Markergen.

große Genexpressionsänderung vorliegt, die die Zellverschiebung sogar überkompensiert, oder ein Messfehler.

Allerdings ist die oben gefundene klare Widerspiegelung der Zellverschiebung in den Genexpressionsdaten der gezeigten zugehörigen Markergene eine weiteres Indiz für die Validität der Messung und Auswertung mit Hilfe der inflammatorischen Array-Plattform.

8.3.5 Charakteristisches inflammatorisches Expressionsprofil für $t_1 \Leftrightarrow t_0$ und $t_2 \Leftrightarrow t_0$

Um eine Art typisches inflammatorisches Expressionsprofil zur Charakterisierung der Zustände $t_1 \Leftrightarrow t_0$ und $t_2 \Leftrightarrow t_0$ zu erhalten, wurden die Expressionsdaten noch einmal dahingehend untersucht, Gene mit hoher übereinstimmender Expressionsänderung zu finden. Dabei wurden für $t_1 \Leftrightarrow t_0$ 36 Gene gefunden, die in allen Läufern eine gleichsinnige Änderung der Expression aufwiesen. Bei $t_2 \Leftrightarrow t_0$ handelte es sich um 21 solcher Gene. Die [ZFD+05] entnommenen Abbildungen 8.3 und 8.4 zeigen die gefundenen Gene für $t_1 \Leftrightarrow t_0$ und $t_2 \Leftrightarrow t_0$. In [ZFD+05] sowie [Zie06] findet sich eine Auflistung der betroffenen Gene zusammen mit den assoziierten physiologischen Funktionen. Aus Platzgründen wurde hier auf eine Darstellung davon verzichtet.

8.3.6 Offene Verfügbarkeit der Daten

Die in dieser Studie erhobenen Daten sind auch über die Datenbank *Gene Expression Omnibus* (GEO) [NCB08, BTW+07] des *National Center for Biotechnology* (NCBI) unter der *GEO Series Accession Number* GSE2532 für die Öffentlichkeit allgemein zugänglich.

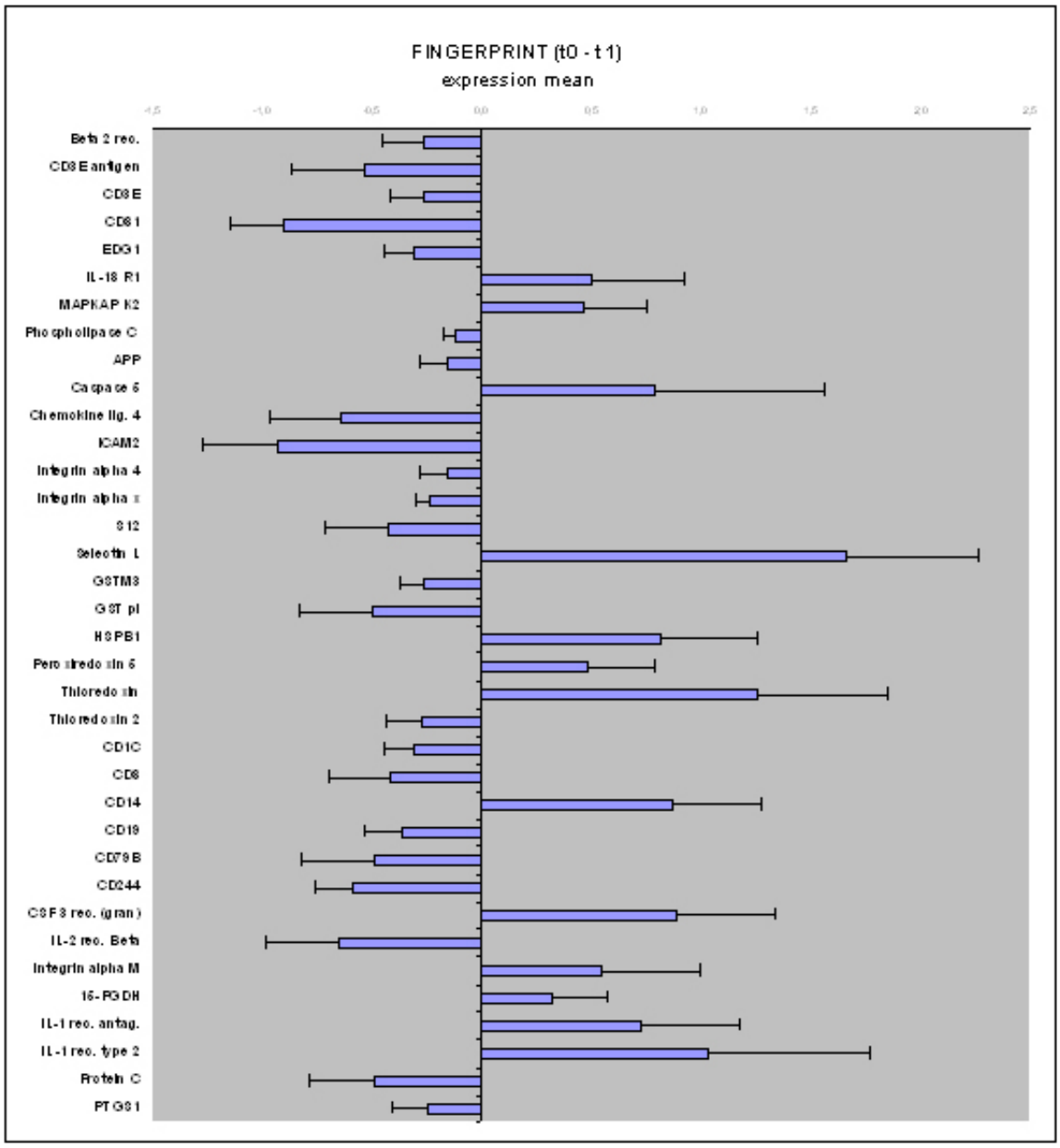


Abbildung 8.3: Das charakteristische Expressionsprofil für $t_1 \Leftrightarrow t_0$ umfasst 36 Gene, die ihre Expression bei allen Läufern in die gleiche Richtung ändern. Angegeben ist $\bar{M}$ samt Fehlerbalken über alle Läufer hinweg. Das Grafik ist [ZFD+05] entnommen.

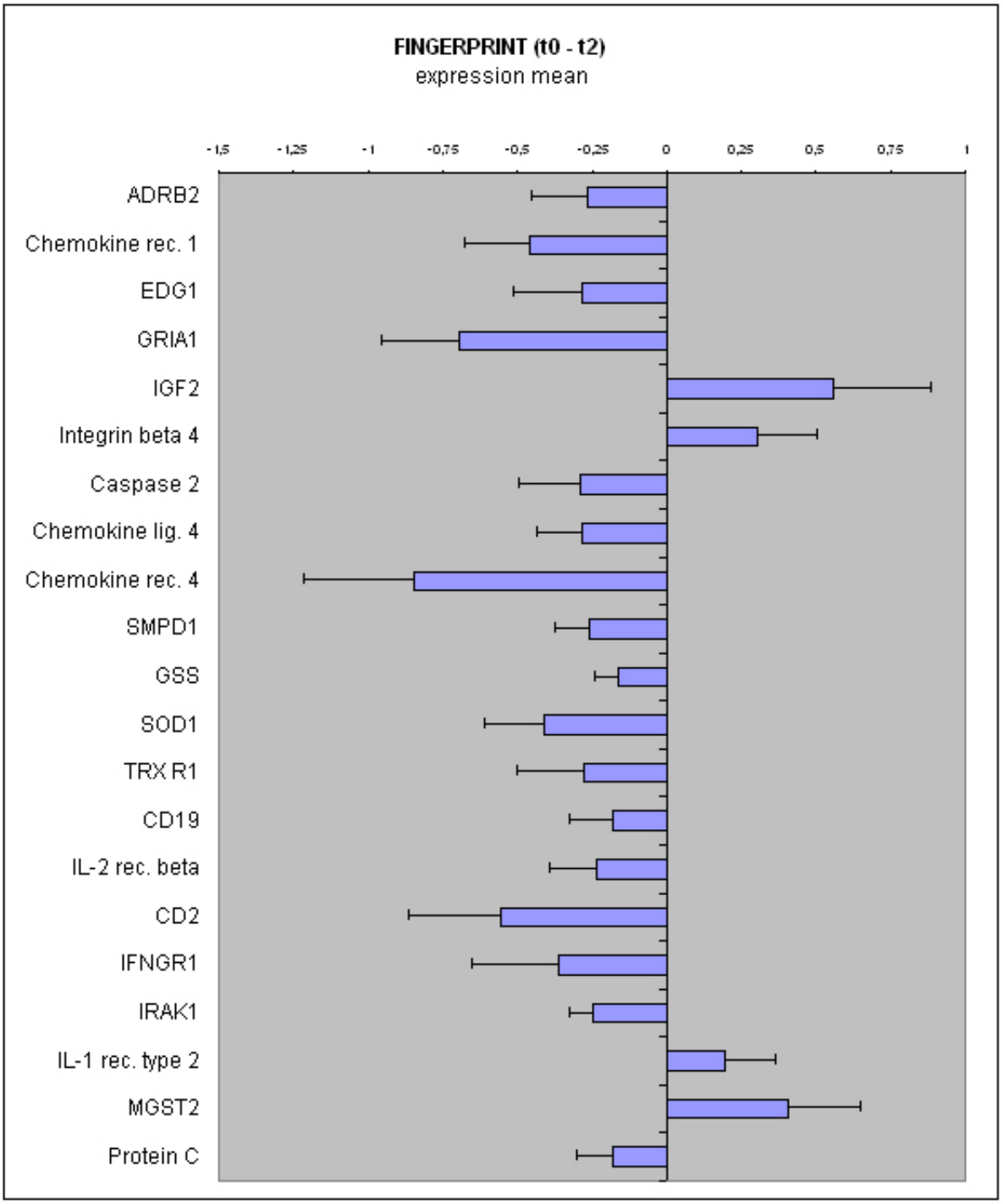


Abbildung 8.4: Das charakteristische Expressionsprofil für $t_2 \Leftrightarrow t_0$ umfasst 21 Gene, die ihre Expression bei allen Läufern in die gleiche Richtung ändern. Angegeben ist $\bar{M}$ samt Fehlerbalken über alle Läufer hinweg. Die Grafik ist [ZFD+05] entnommen.

8.4 Diskussion der Ergebnisse

Für eine ausführliche Diskussion der medizinischen und physiologischen Implikationen der vorgestellten Ergebnisse sei auch hier auf die umfangreichen Ausführungen in dieser Richtung innerhalb von [ZZD+05, ZFD+05, Zie06] verwiesen.

Zusammenfassend lässt sich feststellen, dass diese erste, mit dieser inflammatorischen Array-Plattform durchgeführte Studie ihre primären Ziele erreicht hat und damit die komplette Pipeline, von der Durchführung der Probenentnahme über die Messungen mit dem Array bis hin zur Auswertung, in ihrer Form sehr gut bestätigt hat.

Einerseits konnten Hinweise auf ganz neue Zusammenhänge zwischen Hochleistungssport und inflammatorischem, immunologischen Geschehen im peripheren Blut gefunden werden. Beispielsweise erbrachte es den erstmaligen Nachweis der verminderten Expression von CD 81 und des antikarzinogen wirkenden GSTM3 sowie die erhöhte Expression von Thioredoxin, das innerhalb antioxidativer Prozesse wichtig ist. Dies zeigt, dass die Array-Pipeline die notwendige Sensitivität aufweist, um relevante Informationen zu finden.

Andererseits konnten bekannte Zusammenhänge bestätigt werden, wie die erhöhte Expression von MAPKAP-K2, L-Selectin und IL1-ra im Zusammenhang mit sportlichen Belastungen, was ein gutes Indiz für eine ausreichende Validität liefert.

Erstmalig wurde im Rahmen dieser Studie der Versuch durchgeführt, die gemessenen Genexpressionswerte des Arrays mit Hilfe eines statistischen Modells um die bei sportlicher Belastung auftretende Zellverschiebungsrelationen im peripheren Blut zu korrigieren bzw. diese zu beachten. Dabei zeigte sich für die ermittelten Daten der Expressionsarrays bei zelltypischen Oberflächenmakern eine hochsignifikante Widerspiegelung der Zellverschiebung im untersuchten Vollblut. Auch diese ist ein weiteres Indiz für die Korrektheit der ganzen Array-Pipeline.

Die in Abschnitt 8.3.5 gezeigten charakteristischen Expressionsprofile liefern einen ersten Hinweis in Richtung eines spezifischen „Fingerabdrucks“ für bestimmte physiologische Zustände. Solche charakterisierende Profile könnten sich als ein sehr nützliches Hilfsmittel für die Detektion von zu großen Trainingsumfängen und Stressimpulsen erweisen und sich damit als ein wertvolles Instrument zur Trainingssteuerung sowie zur Entzündungs- und Stressbewältigung etablieren. Der Stichprobenumfang dieser Studie lässt hier leider keine ausreichende Verallgemeinerung zu, zeigt aber die Machbarkeit und legt nachfolgende Studien mit größeren und repräsentativeren Probandenkollektiven nahe.

9 Untersuchung der Genexpression im peripheren Blut von Posttrauma-Patienten

> The only relevant test of the validity of a hypothesis is comparison of its predictions with experience.
>
> *(M. Friedman)*

Nahezu gleichzeitig zur Marathon-Studie wurde in einer zusätzlichen Kooperation mit der Abteilung Psychopharmakologie des *Zentralinstituts für seelische Gesundheit Mannheim* (ZI) der Einsatz des inflammatorischen Arrays für die Untersuchung von Patienten vorgesehen, die an einer postraumatischen Belastungsstörung (PTSD) litten. Darunter versteht man die Entwicklung einer Vielzahl von psychischen und psychosomatischen Symptomen als Langzeitfolge eines Traumas oder einer Reihe von Traumata. Das oder die auslösenden Traumata stellen ein Ereignis von außergewöhnlich großer psychischer Belastung dar, wie die Erfahrung von Lebensgefahr, starker Körperverletzung oder tiefster Verzweiflung.

Im Allgemeinen gestaltet sich die Diagnose nicht einfach, da es keine direkten „harten“ Messwerte gibt, an denen man das Vorliegen eines PTSD festmachen kann. Dass jedoch ein Zusammenhang von Stresszuständen mit physiologischen Prozessen und Parametern besteht, ist bereits bekannt. So beeinträchtigt Stress die Leistungsfähigkeit des Immunsystems. Aus diesem Grund liegt die Annahme nahe, dass auch PTSD einen Einfluss auf das Immunsystem ausübt. Die Ergebnisse bisheriger Arbeiten hierzu sind aber nicht eindeutig und werden deshalb kontrovers diskutiert. Beispielsweise liegen sowohl Arbeiten vor die über eine Verminderung der Cortisolkonzentration [MGK$^+$86, YSN$^+$90] im Zusammenhang mit PTSD berichten, wie auch Arbeiten [GRY05, MLB$^+$98, PO90], die gegenteilig eine Erhöhung gefunden haben. Generell liegt momentan kein eindeutiges Bild in Bezug auf den Einfluss von PTSD auf das Immunsystem vor.

9.1 Studienentwurf

Ziel dieser Studie war, neben einer weiteren Validierung der inflammatorischen Array-Plattform, den im letzten Abschnitt angesprochenen unklaren Zusammenhang zwischen PTSD und dem Immunsystems auf der Ebene der Genexpression

zu untersuchen. Die im Rahmen dieser Studie durchgeführten Untersuchungen einzelner prädestinierter Proteine konnten mit Hilfe der Array-basierten Genexpressionsanalyse sehr gut um ein Hochdurchsatzverfahren zur breiten Identifikation aussichtsreicher Gene bzw. Zielproteine ergänzt werden. Hierzu sollten Vollblutproben eines Kollektivs von PTSD-Patienten mit denen eines Kollektivs passender Kontrollpersonen verglichen werden.

Für die Probengewinnung wurde genauso wie bei der Marathon-Studie vorgegangen (siehe Abschnitt 8.1.2 auf Seite 166) und 5 mal 2.5 ml Vollblut von jedem Patienten und jeder Kontrollperson entnommen. Diese Proben wurden für die weitere Verarbeitung sofort in PAXgene Blood RNA Röhrchen von Qiagen abgefüllt (siehe Abschnitt 7.2.1 auf Seite 158).

Auch bei dieser Studie sei für eine ausführliche Darstellung, insbesondere in medizinisch biologischer Hinsicht, auf die umfangreichen Erläuterungen in [ZZJ+07] und [Zie07] verwiesen.

9.1.1 Das Patientenkollektiv

Das Patientenkollektiv bestand aus 8 Personen, die alle durch das gleiche Ereignis traumatisiert waren. Alle 8 Personen waren entweder Opfer oder Helfer beim Unglück auf dem Fliegerhorst Rammstein im Jahre 1988. Dabei verunglückten drei Maschinen einer Kunstflugstaffel, stürzten brennend in die Zuschauermenge (30000), rissen 70 Menschen in den Tod und verletzten eine Vielzahl der weiteren anwesenden Zuschauer. In Bezug auf das traumatisierende Ereignis stellt die Gruppe damit eine homogene Struktur dar. Eine weitere Homogenität bestand in der Auswahl ausschließlich männlicher Patienten, womit auch die Beeinflussung des peripheren Blutes durch den Hormonzyklus der Frau nicht weiter beachtet werden musste.

Für alle Patienten wurde das Vorliegen eines PTSD eindeutig diagnostiziert und sie wiesen zum Zeitpunkt der Blutentnahme typische Symptome auf. Weiterhin wurde darauf geachtet, dass bei den Patienten möglichst keine akut und chronisch entzündlichen Vorgänge vorlagen. Keiner der Patienten nahm Psychopharmaka ein. Der Altersdurchschnitt betrug 51.1 Jahre. Die Tabelle gibt einen genaueren Überblick des untersuchten Patientenkollektiv.

9.1.2 Die Kontrollgruppe

Die Kontrollgruppe bestand aus 8 männlichen Probanden, die bezüglich des Alters gegenüber den Mitgliedern der Patientengruppe abgeglichen wurden. Alle Probanden wurden auf die gleiche Weise auf PTSD untersucht wie die Patientengruppe. Bei keiner Kontrollperson wurde ein PTSD festgestellt, wobei teilweise traumatische Erfahrungen in der Kontrollgruppe vorlagen. Kein Mitglied der Kontrollgruppe nahm Psychopharmaka ein. Allerdings litten zwei der Kontrollpersonen an einer chronischen Erkrankung, die medikamentös behandelt wurde. Akute entzündliche

Patient	Alter	Verbrennungen	vorliegende Symptome	Chron. Erkr.
p_1	64	0	Seelische Veränderungen	0
p_2	61	0	Einschränkung im Alltag	0
p_3	50	-	Seelische Veränderungen	-
p_4	52	0	-	-
p_5	36	1	1	0
p_6	47	1	1	0
p_7	55	0	Seelische Veränderungen	0
p_8	44	0	Einschränkung im Alltag	-

Tabelle 9.1: Überblick über das Patientenkollektiv der PTSD-Studie, dabei kodiert „1“ das Vorliegen einer Eigenschaft, „0“ deren Abwesenheit und „-“ das Fehlen von Angaben.„Chron. Erkr.“ steht für „Chronische Erkrankungen“.

Kontrolle	Alter	Trauma	Chronische Erkrankung
k_1	60	0	0
k_2	61	1	0
k_3	50	0	Hyperurikämie
k_4	53	1	Hypertonie, Hyperlipidämie
k_5	40	0	0
k_6	46	1	0
k_7	56	1	0
k_8	43	1	0

Tabelle 9.2: Überblick über die Kontrollgruppe der PTSD-Studie, dabei kodiert „1“ das Vorliegen einer Eigenschaft und „0“ deren Abwesenheit.

Prozesse kamen bei keiner Person vor. Tabelle 9.2 gibt einen genaueren Überblick über die Kontrollgruppe.

9.2 Hybridisierungsstrategie

Im Gegensatz zur direkten Hybridisierungsstrategie für die Expressionsanalyse der Marathonläufer (vgl. Abschnitt 8.2 auf Seite 8.2) wurde hier auf ein indirektes Design zurückgegriffen. Die Referenz r_0 des indirekten Designs stellt eine kommerzielle, standardisierte Referenzprobe dar, Universal Human Reference RNA der Firma Stratagene. Diese Referenz wird aus 10 verschiedenen Zelllinien gefertigt, die folgende Gewebetypen abdecken: Gehirn, Cervix, Lunge, Leber, B-Lymphozyten, Y-Lymphozyten, Liposarkom, Makrophagen, Hoden und Haut.

Ein indirektes Design mit einer standardisierten, immer verfügbaren Referenz ist in diesem Fall aufgrund seiner Flexibilität von großem Vorteil. Einerseits bleibt so die Möglichkeit, die Daten neu untersuchter Personen mit allen bisher ermittelten

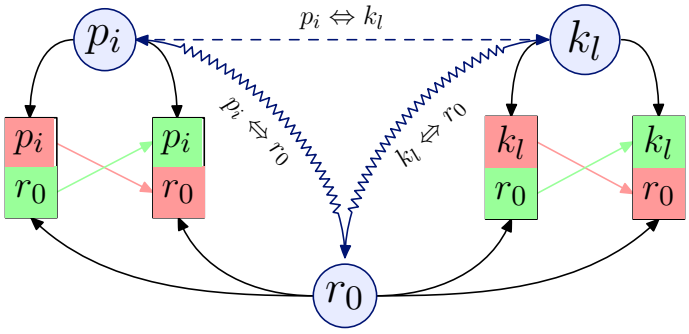


Abbildung 9.1: Die Darstellung der Hybridisierungsstrategie für die PTSD-Studie läßt sich auf die jeweilige Messung der Expression des Probanden p_i und seiner genau passenden Kontrollperson k_l beschränken und ist dabei der in Abbildung 8.1 gezeigten Darstellung für die Marathon-Studie sehr ähnlich. Die Vergleiche $k_l \Leftrightarrow r_0$ und $p_i \Leftrightarrow r_0$ werden direkt auf zwei Dye-Swap-Arrays gegenüber der Referenz mRNA r_0 vorgenommen. Der eigentlich interessierende Vergleich $p_i \Leftrightarrow k_l$ zwischen Patient und zugehöriger Kontrollperson muß dann rechnerisch, auf der Grundlage der gemeinsamen Referenz r_0 ermittelt werden.

Versuchswerten vergleichen bzw. verrechnen zu können. Andererseits besteht so die Freiheit für den Vergleich zwischen Patienten und Kontrollen auch Daten neu erhobener Kontrollpersonen zu verwenden, die eventuell besser abgestimmt zu dem jeweiligen Patienten passen.

Abbildung 9.1 stellt die Hybridisierungsstrategie grafisch dar. Von eigentlichem Interesse ist lediglich der Vergleich $p_i \Leftrightarrow k_l$ zwischen Patient p_i und seiner Kontrollperson k_i, der über die beiden direkten Dye-Swap-Replikate der Vergleiche $k_l \Leftrightarrow r_0$ und $p_i \Leftrightarrow r_0$ errechnet werden muss.

9.3 Statistische Auswertung

Als Ergebnis der Hybridisierung lagen für die 2 Kollektive von 8 Patienten ($p_i \Leftrightarrow r_0$) und 8 Kontrollen ($k_l \Leftrightarrow r_0$) jeweils 2 Dye-Swap-Replikate vor, d.h. insgesamt $2 * 8 * 2 = 32$ Arrays. Auch bei dieser Studie wurden jedoch 17 Subarrays von 96 möglichen aufgrund von durchgehend schlechter Qualität schon vor der Analyse verworfen. Die fehlenden Subgrids verteilten sich aber auch hier entsprechend, so dass die Information von den Replikaten der jeweiligen Messung erbracht wurde.

9.3.1 Gewinnung der Intensitätsrohdaten

Die Arrays wurden mit einem Scanner der Firma Genetix Limited, Hamphshire, UK eingescannt und anschließend mit ImaGene 5 (BioDiscovery, Link) in Bezug auf die Bildbearbeitung ausgewertet. Für die nachfolgende Analyse wurden die Daten aller Spots verworfen, die von ImaGene 5 als *Poor* markiert wurden. Spots erhalten diese Markierung bei aufgetretener Kontamination des Vorder- oder Hintergrundsignals,

30336	100%	verfügbare Spots für die Auswertung auf allen Arrays
24924	$\sim$ 82.16%	Spots ohne Flag
519	$\sim$ 1.71%	Empty- & Negative-Spots
4893	$\sim$ 16.13%	Spots mit gesetztem Flag-Code

Tabelle 9.3: Spot-Statistik für alle Arrays der PTSD-Studie. Übersicht über alle vorhanden Spots – 8 Patienten und 8 zugehörige Kontrollprobanden ($p \Leftrightarrow r$, $k \Leftrightarrow r$), zwei Dye-Swap-Arrays mit jeweils 3 Subarrays von 384 Spots, d.h. theoretisch $8*2*2*3*384 = 36864$ Spots. Da auch hier 17 Subarrays von zu schlechter Qualität waren, wurden sie bereits beim Scannen verworfen und nur 30336 Spots lagen insgesamt zur Auswertung vor. Berücksichtigt man diese verwofenen Spots, so kommt man mit $\sim$ 31.0 % bereits in den Bereich der Fehlerquote des Vorversuchs (vgl. mit Tabelle 7.1).

bei irregulärer Form des Spots oder ganz allgemein bei einer detektierten schlechten Qualität der Spotsignale. Insgesamt wurden so $\sim$ 17.8 % aller gemessenen Spots, einschließlich der Replikate, von der Auswertung ausgenommen (vgl. Tabelle 9.3).

Als Schätzer für die Intensitätswerte $\hat{I}_{Ch^x}$ beider Kanäle Ch^x eines Spots wurden jeweils der arithmetische Mittelwert der Verteilung der Vordergrundpixel und der Median der Verteilung der Hintergrundpixel verwendet.

9.3.2 Normalisierung

Infolge der auch hier gleichen Dye-Swap-Struktur der einzelnen Hybridisierungen, konnte die Normalisierung der Array-Daten vollkommen analog zu dem Normalisierungsverfahren erfolgen, das in Abschnitt 7.2.5 auf Seite 160 dargestellt wurde und zugleich bei der Marathon Studie Anwendung fand.

9.3.3 Detektion differentiell exprimierter Gene

Aus dem ersten Teil der Analyse der Hybridisierungen des indirekten Designs dieser Studie $k_l \Leftrightarrow r_0$ und $k_l \Leftrightarrow r_0$ lassen sich die Log-Ratios M_{p_i} und M_{k_l} gewinnen. M_{p_i} und M_{k_l} geben die Expressionsniveaus für den jeweiligen Patienten und der zugehörigen Kontrollperson in Bezug auf die gemeinsame Referenz r_0 (UniRNA) an. Damit sind differentiell expremierte Gene an statistisch signifikant unterschiedlichen Log-Ratios M_{p_i} und M_{k_l} eines Patienten und ihrer zugeordneten Kontrollperson in Bezug auf die Referenz r_0 zu erkennen. Drei verschiedene statistische Testmethoden dienten der Einschätzung der statistischen Signifikanz von Unterschieden zwischen diesen Log-Ratios.

Als erstes wurde ein gepaarter Zwei-Stichproben-t-Test als parametrischer und ein Wilcoxon-Test als nichtparametrischer Test durchgeführt. Die Signifikanzschwelle wurde auf 5 % festgelegt ($\alpha = 0.05$), d.h. ein Gen wurde als differentiell exprimiert bezeichnet, wenn der P-Value des Tests unter 5 % lag. Zusätzlich erfolgte auch die Anwendung von Methoden zur Korrektur für multiples Testen, wie Bonferroni

Gen	pfp-Value
Insulin-like Growth Factor 2	0.00078
IL 8 Receptor α	0.00401
Integrin β 4	0.02113
High Affinity Aspartate/Glutamate Transporter, Member 6	0.03260
Endoth. Diff. Lysophosphat. Acid G-Prot. Coupled Rec. 4 (EDG4)	0.08566

Tabelle 9.4: Übersicht über die vom RP als in ihrer Expression hochreguliert detektierten Gene

[Bon36], Benjamini-Yekutieli [BY01], Benjamini-Hochberg [BH95], Holm [Hol79] und Sidak [Sid67].

Unter Berücksichtigung des dateninhärenten recht hohen Rauschens kam zusätzlich das RP zum Einsatz. Das RP ist eine relativ neue nichtparametrische Testmethode, die in [BAAH04] vorgeschlagen wurde. Hierbei werden die Gene detektiert, die über die einzelnen Replikate eines Experimentes hinweg, konsistent zu den am stärksten hoch- oder runterregulierten Genen zählen. Methodisch wird dabei für jedes Gen bzw. Transkript und jede Hybridisierung der Log-Ratio (M_{p_i,k_l}) aus den beiden Expressionswerten des Patienten und der Kontrollperson bestimmt. Innerhalb einer solchen Liste für eine einzelne Hybridisierung werden die Gene entsprechend ihres Log-Ratio-Wertes rangsortiert, so dass das Gen mit dem höchsten positiven Log-Ratio den Rang 1 besitzt und das mit dem niedrigsten negativen den letzten. Dann wird über alle Hybridisierungen hinweg das geometrische Mittel der Ränge für jedes Gen bestimmt. Am Ende werden die Rangprodukte gebildet, und es ergeben sich zwei sortierte Listen in der Reihenfolge, mit der die Gene am wahrscheinlichsten hoch- bzw. runterreguliert sind. Das heißt, je kleiner der Wert des RPs eines Genes ist, desto wahrscheinlicher ist seine Position in der Liste nicht zufälligen Faktoren zuzuschreiben. Um die Signifikanz eines Rangproduktes zu charakterisieren, wird die *Percentage of False Positive Preditions* (pfp), die der FDR vergleichbar ist, in einem permutationsbasierten Ansatz berechnet. Im vorliegenden Fall sind 100000 Permutationen durchgeführt worden. Da das RP die gesamte „Genkollektion“ für den Permutationstest benutzt, berücksichtigt das pfp bereits das Problem des multiplen Testens und bedarf somit keiner weiteren Korrektur. Gene mit einem pfp-Wert unter 5 % wurden als statistisch signifikant betrachtet.

9.4 Ergebnisse

Die sehr strengen statistischen Methoden, bestehend aus der Kombination von Wilcoxon- oder t-Test mit den Korrekturen für multiples Testen bezogen auf das ganze Array, zeigten sich als zu rigoros und zu wenig sensitiv. Alternativ detektierte der Ansatz aus unkorrigiertem Wilcoxon- und t-Test sowie dem RP 4 hoch- und 14 herunterregulierte Gene (Tabellen 9.4 und 9.5), die in mindestens einem dieser

Gen	pfp-Value
Colony Stim. Factor 2 Receptor β Low Aff. (Gran.-Macroph.)	0.00034
Thioredoxin Reductase 1 (TXR)	0.00031
Interleukin 18 (Interferon γ Inducing Factor) (IL18)	0.00110
Chemokine Receptor 1	0.00104
3-Phosphoglycerate Dehydrogenase (PHGDH)	0.00207
Casein Kinase 1 γ 3	0.00704
Caspase 2	0.00668
Superoxide Dismutase 1 (SOD)	0.00972
CD 3Z	0.01077
Interleukin 16 (IL16)	0.02826
Endoth. Different. Sphingolipid G-Prot.-Coupledrecept. 1 (EDG1)	0.02658
G-protein Coupled Receptor 65	0.03056
Calnexin	0.03283
CD 81	0.04736

Tabelle 9.5: Übersicht über die vom RP als in ihrer Expression herunterreguliert detektierten Gene.

Tests auch statistische Signifikanz erreichten.

Aus den statistisch signifikant differentiell exprimierten Genen wurden 7 Gene und ein weiteres, nicht auf dem Array gespottetes Gen für die Validierung bzw. Analyse durch eine Real-Time-PCR ausgewählt. Diese 8 Gene standen in besonderem Interesse, da sie im Zusammenhang mit PTSD aufschlussreiche Wechselwirkungen aufzeigen können. Bei dem achten zusätzlichen Gen handelte es sich um den XC(-) Glutamat-Cystin-Antiporter (XC).

EDG4 und PHGDH erreichten in der Real-Time-PCR kein signifikantes Ergebnis, womit sich das Ergebnis der Microarray-Analyse bei 5 der 7 Gene bestätigte (siehe Tabelle 9.6). Der Wert für XC wurde ausschließlich über die Real-Time-PCR bestimmt und ergab eine verringerte Expression. Abbildung 9.2 gibt einen besseren Überblick über die genaue Verteilung der PCR-Ergebnisse in Form von Boxplots. Betrachtet man diese genauer, so bleibt als einziges, wirklich unentscheidbares Ergebnis das für EDG4 bestehen.

9.5 Diskussion

Auch bei dieser Studie sei für eine ausführliche Diskussion der medizinischen und physiologischen Implikationen der vorgestellten Ergebnisse auf die umfangreichen Ausführungen in dieser Richtung innerhalb von [ZZJ+07] und [Zie07] verwiesen.

In Bezug auf den Einsatz des inflammatorischen Arrays ist festzuhalten, dass die gefundenen Ergebnisse auch hier in sich stimmig sind und durch die Validierung mit einem zweiten Verfahren (Real-Time-PCR) im Wesentlichen bestätigt wur-

Gen	$\bar{M}$	$\tilde{M}$	p-Value
EDG4	-0.3044	-0.0056	0.7790
IL18	-1.4407	-1.7337	0.0047
SOD	-0.7834	-0.7213	0.0156
TXR	-1.0474	-1.4807	0.0265
EDG1	-1.208	-2.4014	0.0377
IL16	-0.4975	-0.4157	0.0407
PHGDH	-0.4454	-0.5422	0.2738
XC	-2.2625	-2.45	0.0490

Tabelle 9.6: Übersicht über die Analysewerte der Gene, deren Verhalten mit quantitativer Real-Time-PCR zusätzlich validiert wurde. Neben dem aus der PCR-Analyse bestimmten Mittelwert $\bar{M}$ ist für das Gen auch der Median $\tilde{M}$ angegeben, sowie der p-Value des t-Tests.

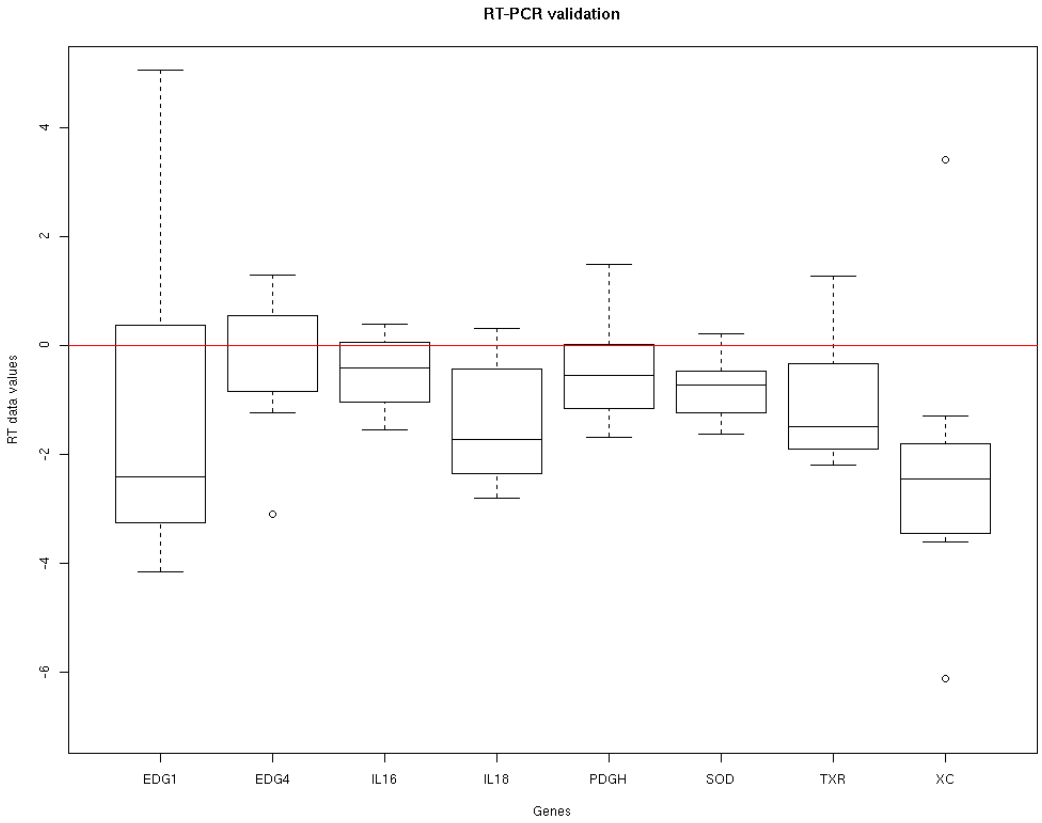


Abbildung 9.2: Boxplots der Ergebnisdaten der quantitativen Real-Time-PCR.

den. Lediglich im Falle von EDG4 kann dem Ergebnis der Real-Time-PCR keine bestätigende Information entnommen werden – allerdings auch keine widerlegende.

Betrachtet man beide Studien Marathon und PTSD im Vergleich, so ist die Auswertung der PTSD-Studie aus statistischer Sicht erheblich schwieriger. Der beobachtete Effekt ist kleiner und damit auch der Abstand zwischen dem zu detektierenden Signal und dem allgemeinen „Rauschen“. Aus diesem Grund ergab sich auch die Einschränkung bei der Benutzung sehr stringenter, konservativer Korrekturmethoden für das multiple Testen. Im Gegensatz dazu sind die Signale der Marathonstudie klarer, was angesichts des stärkeren körperlichen Stressfaktors durch die intensive körperliche Belastung auch wahrscheinlicher erscheint.

Sehr positiv im statistischen Sinne ist die Homogenität der Patientengruppe in Bezug auf das auslösende traumatische Ereignis. Als schwieriger bzw. diskusionsfähiger einzuschätzen ist die Abstimmung zwischen Patienten und ihren jeweiligen Kontrollen. Das Matching ist für die Validität der Ergebnisse und ihrem allgemeinen Wert von sehr großer Bedeutung, findet aber nur nach dem vergleichsweise einfachen Kriterium des Alters statt. Hier sollten möglicherweise weitere Kriterien Einfluss finden. Abgesehen davon bestände die passende Kontrolle aus einer Person, die nicht nur in den ausgewählten Kriterien zu einem Patienten passt, sondern auch das gleiche auslösende Trauma erlebt hat, ohne ein PTSD zu entwickeln. Diese Kontrollgruppe ist allerdings schwer bis unmöglich zu rekrutieren.

Trotz dieser kritischen Punkte sind die Resultate der Genexpressionsanalyse sehr interessant und vielversprechend, so erscheint das PTSD mit einer verringerten Expression von Genen, die mit der Stress- und Immunantwort assoziiert sind, zusammenzuhängen. Die eine Gruppe dieser Gene ist mit dem oxidativen Stressgeschehen verbunden und die andere mit dem Entzündungsgeschehen. Speziell die verminderte Expression von ersteren könnte für eine verminderte Menge von antioxidativen Enzymen sorgen und somit für eine Anhäufung von reaktiven Sauerstoffradikalen. Das wiederum könnte einen Grund dafür darstellen, dass PTSD-Patienten ein höheres Risiko für die Entwicklung von Autoimmunerkrankungen oder kardiovaskulären Krankheiten haben. Hinweise auf einen möglichen Zusammenhang zwischen PTSD und oxidativem Stress existierten bereits [PS01].

Veröffentlichungen und Preise im Zusammenhang mit dieser Arbeit

Veröffentlichungen in Fachzeitschriften

2008:

- J. Dietzsch, J. Heinrich, K. Nieselt, D. Bartz, „Visual Analysis of Microarray Data from Bioinformatics Applications“, WSI-Report, ISSN 0946-3852, WSI-2008-1, 2008

2007:

- J. Zieker, D. Zieker, A. Jatzko, J. Dietzsch, K. Nieselt, A. Schmitt, T. Bertsch, K. Fassbender, R. Spanagel, H. Northoff and PJ. Gebicke-Haerter: „Differential gene expression in peripheral blood of patients suffering from Posttraumatic Stress Disorder“, Molecular Psychiatry 12(2): 116-118, 2007

2006:

- E. Fehrenbach, D. Zieker, J. Dietzsch, A. M Niess, H. Northoff, „Microarray-based Gene Expression Fingerprint in Leukocytes of Half-marathon Runners“, Med Sci Sports Exerc. 2006 May, 38:S76
- J. Dietzsch, N. Gehlenborg, K. Nieselt: „Mayday - a Microarray Data Analysis Workbench“, Bioinformatics 22(8):1010-1012, 2006

2005:

- D. Zieker, J. Zieker, J. Dietzsch, M. Burnet, H. Northoff, E. Fehrenbach: „cDNA-microarray analysis as a research tool for expression profiling in human peripheral blood following exercise“, Exerc Immunol Rev. 2005, 11:86-96
- D. Zieker, E. Fehrenbach, J. Dietzsch, J. Fliegner, M. Waidmann, K. Nieselt, P. Gebicke-Haerter, R. Spanagel, P. Simon, A. Niess, H. Northoff: „cDNA microarray analysis reveals novel candidate genes expressed in human peripheral blood following exhaustive exercise“ Physiol Genomics 23:287-294, 2005
- N. Gehlenborg, J. Dietzsch, K. Nieselt: „A Framework for Visualization of Microarray Data and Integrated Meta Information“, Information Visualization, Vol. 4(3), 164-175, 2005. Special issue „Bioinformatics Visualization“

2004:

- P. Khaitovich,B. Muetzel, X. She, M. Lachmann, I. Hellmann, J. Dietzsch, S. Steigele, H-H. Do, G. Weiss, W. Enard, F. Heissig, T. Arendt, K. Nieselt-Struwe, E. Eichler, S. Pääbo: „Regional Patterns of Gene Expression in Human and Chimpanzee Brains“, Genome Research 14, 2004

Preise

2006:

- bwcon Sonderpreis IT & Life Sciences 2006 des Landes Baden-Württemberg für die Plattform *Mayday* zur Auswertung von Microarray Daten

Konferenzbeiträge

2008:

- J. Heinrich, J. Dietzsch, D. Bartz, K. Nieselt, „SpRay – an Visual Analytics Plattform“, Präsentation auf der „useR! 2008“

2004:

- J. Dietzsch, M. Zschunke, K. Nieselt, „R inside - let R drive your bioinformatic application“, Präsentation innerhalb einer Island-Session der „useR! 2004“

Posterbeiträge zu Konferenzen

2007:

- S. Symons, C. Schillinger, J. Dietzsch, F. Battke, K. Nieselt „GeneMining in Mayday, a feature selection framework for binary classification“ auf der „GCB 2007“ in Potsdam
- S. Symons, J. Dietzsch, K. Nieselt „Flexible Microarray data analysis using Mayday“ auf der „European BioPerspectives 2007“ in Köln

2006:

- J. Dietzsch, J. Heinrich, K. Nieselt, D. Bartz, „Extended Parallel Coordinates for Bioinformatical Applications“ auf der „GCB 2006“ in Tübingen
- M. Zschunke, K. Deubel, S. Symons, J. Dietzsch, K. Nieselt, „FAGE and VEGA: two new tools for functional and epigenomics expression analysis in Mayday“ auf der „GCB 2006“ in Tübingen
- S. Symons, F. Battke, J. Dietzsch, M. Zschunke, K. Nieselt, „Automated Processing and Machine Learning Tools for Mayday“ auf der „GCB 2006“ in Tübingen

2005:

- D. Zieker, J. Zieker, E. Fehrenbach, J. Dietzsch, P. Gebicke, H. Northoff, „Design, manufacture and use of a cDNA microarray for analysis of inflammatory status of human peripheral blood or other tissue samples“ auf dem 7. ISEI Symposium 2005 in Monaco

2004:

- J. Dietzsch, M. Zschunke, K. Nieselt, „R inside - let R drive your bioinformatic application“ auf der „useR! 2004“ in Wien

2003:

- J. Dietzsch, N. Gehlenborg, K. Nieselt, „MAYDAY“ auf der „GCB 2003“ in München

2002:

- S. Steigele, J. Dietzsch, P. Khaitovich, D. Huson, S. Pääbo, K. Nieselt-Struwe, „Clustering of differentially expressed genes in the human genome“ auf der „ECCB 2002“ in Saarbrücken

Erfolgreiche DFG-Förderungsanträge im Zusammenhang mit dieser Arbeit

2008:

- Erfolgreicher Förderantrag im Rahmen des Schwerpunktprogramms „Scalable Visual Analytics: Interactive Visual Analysis Systems of Complex Information Spaces“ (SPP 1335) der DFG

Abkürzungsverzeichnis

Abstrakter Datentyp (ADT) Unter einem Abstrakten Datentyp versteht man in der Informatik einen Datentyp, der ausschließlich über die, auf Exemplare diesen Typs, anwendbaren (Zugriffs-)Operationen definiert ist. Die interne Implementierung ist nach außen hin nicht sichtbar - von ihr wird abstrahiert (Geheiminsprinzip der Informatik). Der ADT war ursprünglich ein Konzept des Entwurfs und ist im Rahmen der OOP mit dem in modernen Programmiersprachen integrierten Klassenkonzept näher an die direkte Implementierung geführt worden (vgl. S. 533 in [Bal04] und S. 196 in [Sch01]).

Analysis of Variance (ANOVA) Die ANOVA (oder auch Varianzanalyse) stellt ein statistisches Verfahren für den Mittelwertvergleich von verschiedenen Gruppen dar. Die vorliegenden Daten müssen dabei mindestens einer Intervallskala entstammen. Eine ANOVA ist somit ein sogenannter Omnibustest für den simultanen Vergleich von Mittelwerten mehrerer Stichproben (vgl. Post-Hoc Test) [SH06].

Application Programming Interface (API) Unter einem API versteht man eine in einer Programmiersprache definierte Zugriffsschnittstelle auf die Funktionalität des Betriebssystems, einer Anwendung oder einer Software-Bibliothek.

Body-Mass-Index (BMI) Der Body-Mass-Index (auch Körpermassenzahl KMZ) dient als Maßzahl zur Einschätzung des Körpergewichtes eines Menschen. Er berechnet sich nach der folgenden Formel $\text{BMI} = \frac{m}{l^2}$, wobei m das Körpergewicht in Kilogramm und l die Körpergröße des Menschen in Metern angibt. Der BMI als Indikator für Übergewicht ist nicht unumstritten, da er die Gewebeverteilung des Körpergewichtes unberücksichtigt lässt.

komplementäre DNA (cDNA) cDNA (complementary DNA) ist eine Abschrift von mRNA, die durch reverse Transkription erzeugt wurde. Im Vergleich zu genomischer DNA enthält sie deshalb keine Introns mehr. Als cDNA-Klon wird DNA bezeichnet, die von cDNA kloniert wurde. cDNA-Bibliotheken (cDNA-Libraries) bezeichnen eine Sammlung solcher Klone, die alle in einer Zelle eines bestimmten Typs exprimierten Gene repräsentieren (G:6 in [AJL+02]).

Comparative Genomic Hybridization (CGH) Bei einer CGH erfolgt eine komparative bzw. kompetetive Hybridisierung fluoreszenzmarkierter, genomischer DNA der zu untersuchenden Zellen und der Referenz an einer speziell präparierten

Sonden-DNA. Im traditionellen Fall stellt diese Sonden-DNA ein Präperat der Chromosomen in der Metaphase dar und bei Array-CGH die einzelnen Sonden eines Microarrays (siehe auch Abschnitt 2.1.3 auf Seite 19).

Chromatin-Immunopräzipitation (ChIP) Bei der Chromatin-Immunopräzipitation handelt es sich um eine Methode, bei der die Bindesstelle DNA-bindender Proteine bestimmt werden. Hierzu werden die DNA-gebundenen Proteine mit dem Chromatin am Bindeort per Cross-Linking gebunden und diese Chromatin-Protein-Komplexe danach in Fragmente geschnitten. Anschließend erfolgt die Immunopräzipitation der interessierenden Protein-Komplexe mit Hilfe spezifischer Antikörper für die zu untersuchenden Proteine. Nach der Aufspaltung der gereinigten Protein-Komplexe wird deren DNA-Anteil analysiert und die Bindestelle identifiziert. Letzteres kann durch PCR mit spezifischen DNA-Primern durchgeführt werden oder per Microarray (siehe S. 394f in [AJL+02], ChIP-on-Chip im Glossar und Abschnitt 2.1.1).

Copy Number Variation (CNV) Die CNV, manchmal auch als Copy Number Polymorphism - CNP bezeichnet, beschreibt die Bandbreite an Genduplikationen, die im Genom der Individuen einer Population vorkommen können. Sie stellt neben den SNPs einen weiteren wesentlichen Grundpfeiler der genetischen Vielfalt einer Population dar und ist damit zugleich eng mit potentiellen Prädispositionen eines Individuums für Krankheiten oder bestimmte phänotypische Merkmale verbunden (vgl. Abshnitt 2.1.3).

Central Processing Unit (CPU) Die Central Processing Unit (Hauptprozessor, Mikroprozessor oder Prozessor) ist die zentrale Steuer- und Recheneinheit eines Computers.

Datenbankmanagementsystem (DBMS) Unter dem Datenbankmanagementsystem versteht man die verwaltende Software eines Datenbanksystems. Sie organisiert, strukturiert und verwaltet die Daten nach einem vorgegeben Datenmodell. Das DBMS koordiniert und organisiert alle schreibenden und lesenden Zugriffe auf die Daten des Datenbanksystems. (vgl. [CBS02])

Density Based Spatial Clustering of Applications with Noise (DBSCAN) DBSCAN ist ein dichtebasiertes Clusterverfahren, das in seiner Ausführung einem Region Growing Verfahren, dem Ausdehnen eines Clusters von einem Startelement oder einer Startkollektion aus, mit einem Dichtekriterium gleicht [EKSX96]. Die beiden Parameter ϵ (Durchmesser der ϵ-Umgebung eines Elements) und $minPts$ (minimale Anzahl an Elementen innerhalb der ϵ-Umgebung eines Elements im Clusterinneren) des Algorithmus werden heuristisch aus der Dichteverteilung des Datensatzes geschätzt.

Digital Micromirror Device (DMD) Ein Mikrospiegelarray (DMD) ist ein Bauelement, das mit Hilfe von matrixförmig verteilten, einzeln schwenkbaren

Mikrospiegeln optische Bilder projezieren kann. Es findet als grundlegender Baustein in Beamern breite Verwendung.

Desoxyribonukleinsäure (DNA) DNA ist eine Nukleinsäure. Sie bildet in Form einer gewundenen Doppelhelix im Kern von Zellen den Träger der Erbinformation. Die Erbinformation ist in der Abfolge der Nukleotide (Basen- oder Nukleotidsequenz) codiert (vgl. Abschnitt 1.1 auf S. 3). Durch Replikation der DNA bei der Zellteilung wird die Erbinformation von Generation zu Generation weiter gereicht.

Dynamic Queries (DQ) Dynamic Queries werden von Programmen zur Visualisierung angeboten und gestatten die dynamische, grafische Selektion von zu visualisierenden Objekten bzw. Informationen. Die Ergebnisse dieser Selektion müssen mit einer sehr geringen Reaktionszeit der Anwendung wieder visualisiert werden (vgl. hierzu auch das Glossar von [Inf07]).

Expressed Sequence Tag (EST) Als ESTs bezeichnet man Nukleotidsequenzen, die aus der Sequenzierung von Klonen der in der Zelle gefundenen *m*RNA gewonnen wurden. Technologisch bedingt stellen sie oft nur Fragmente der urschprünglichen mRNA dar, bei der es sich um eine kodierende oder nichtkodierende Sequenz handeln kann.

Focus+Context (F+C) Mit Focus+Context werden in der Informationsvisualisierung Techniken bezeichnet, die es erlauben, das jeweilige interessierende Objekt, den Fokus, in allen Details darzustellen, während gleichzeitig dessen „Einbettung“ in seine informationelle Umgebung, den Kontext, im Überlick erkennbar ist. F+C Techniken gewähren einen Kompromiss zwischen grober Globalübersicht und detaillierter Ausschnittssicht (vgl. hierzu auch das Glossar von [Inf07]).

Fold Change (FC) Der Fold Change gibt das Verhältnis der Expressionswerte EV_1 und EV_2 an: $\mathrm{FC} = \frac{\mathrm{EV}_2}{\mathrm{EV}_1}$

False Discovery Rate (FDR) Die FDR ist eine statistische Methode, mit deren Hilfe das Problem des multiplen Testens bei Mehrfachvergleichen in Hypothesentests angegangen werden kann. Die FDR kontrolliert den Anteil an fälschlicherweise abgelehnten Nullhypothesen (Typ-I-Fehler). Sie ist weniger konservativ und besitzt eine größere statistische Power als die FWER, birgt dafür aber ein höheres Risiko für einen Typ-I-Fehler.

Falschnegativrate (FN) Als Falschnegativrate FN bezeichnet man das Verhältnis $\mathrm{FN} = \frac{\#\text{ falsch negativ}}{\#\text{ richtig negativ}+\#\text{ falsch negativ}}$. Sie gibt den Anteil der durch ein Klassifikationsverfahren, wie z.B. einen statistischen Test, fälschlicher Weise als negativ eingestuften Objekte in Bezug auf die in der Ausgangsmenge insgesamt vorhandenen, positiv zu klassifizierenden Objekte an. Die Falschnegativrate

ergänzt sich mit der Sensitivität (Richtigpositivrate RP) zu 1 bzw. 100 % FN + RP = 1.

Falschpositivrate (FP) Als Falschpositivrate FP bezeichnet man das Verhältnis $\mathrm{FP} = \frac{\#\text{ falsch positiv}}{\#\text{ falsch positiv}+\#\text{ richtig negativ}}$. Sie gibt den Anteil der durch ein Klassifikationsverfahren, wie z.B. einen statistischen Test, fälschlicher Weise als positiv eingestuften Objekte in Bezug auf die in der Ausgangsmenge insgesamt vorhandenen, negativ zu klassifizierenden Objekte an. Die Falschpositivrate ergänzt sich mit der Spezifität (Richtignegativrate RN) zu 1 bzw. 100 % FP + RN = 1.

Floating Point Unit (FPU) Die Floating Point Unit (Gleitkommaeinheit) oder auch Floating Point Processing Unit ist eine Recheneinheit innerhalb oder außerhalb der CPU eines Rechners, der auf die Ausführung von mathematischen Funktionen und Operationen auf Gleitkommazahlen (Floating Point Numbers) spezialisiert ist (mathematischer (Ko-)Prozessor).

Familywise Error Rate (FWER) Mit Hilfe der FWER lässt sich das Problem des multiplen Testens eingrenzen. Unter der FWER versteht man die Wahrscheinlichkeit die Nullhypothese bei paarweisen Mehrfachtests mindestens einmal fälschlicherweise abzulehnen.

Gene Expression Omnibus (GEO) GEO stellt eine der beiden wichtigsten unter [NCB08] öffentlich zugänglichen Datenbanken für Expressionsdaten dar und wird am NCBI administriert und angeboten (siehe [BTW+07]).

Grafikprozessor (GPU) Als Grafikprozessor (Graphics Processing Unit, seltener Visual Processing Unit) bezeichnet man den speziell auf die Grafikausgabe ausgelegten Prozessor eines Computers oder einer Spielkonsole.

Gene Set Enrichment Analysis (GSEA) Bei GSEA handelt sich es um ein Verfahren, phänotypische Unterschiede auf Veränderungen in der Expression einer Gruppe von Genen, dem Gene Set, zurückzuführen. Dabei wird festgestellt, ob die Gene dieses Gene Sets in einer, nach der Korrelation zu einem bestimmten Phänotyp, geordneten Liste aller Gene auf einer Seite überrepräsentiert (enriched) sind. Die Signifikanz des Enrichment Scores (ES) wird mit Hilfe von Permutationstests abgeschätzt [STM+05, SKG+07].

Graphical User Interface (GUI) Als GUI bezeichnet man die grafische Benutzerschnittstelle für die Kommunikation des Nutzers mit der Anwendung

High Performance Computing (HPC) High Performance Computing - Hochleistungsrechnen für umfangreiche computerintensive wissenschaftliche oder ingenieurtechnische Fragestellungen

Harmonische Regressionsanalyse (HRA) Unter einer Harmonischen Regressionsanalyse versteht man eine lineare Regressionsanalyse unter Benutzung der Trigonometrischen Basisfunktionen (näheres hierzu auf S. 132 ff).

Innovation Center for Computer Assisted Surgery (ICCAS) Das „Innovation Center for Computer Assisted Surgery“ der Medizinischen Fakultät an der Universität Leipzig (`www.iccas.de`) ist ein interdisziplinäres Zentrum für die Kooperation von Medizin, Informatik und Ingenieuerwesen zur Entwicklung und Implementation neuer technologischer Methoden für die klinische Praxis.

Integrated Development Environment (IDE) Integrierte Enstwicklungsumgebungen sind Anwendungen, die den Entwickler bei der Programmierung unterstützen. Sie integrieren eine mehr oder weniger umfangreiche Menge an Komponenten, wie Texteditoren, Compiler oder Interpreter, Debugger, Linker, Profiler, Projektmanagement, UML-Modellierer oder Versionsverwaltung.

Information Gain (IG) IG auch als Kullback-Leibler-Divergenz, -Information oder -Entropie bekannt [KL51], stellt ein Maß für die Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen F_1 und F_2 dar. Für diskrete Verteilungen wird sie nach $KL(F_1, F_2) = \sum F_1(x) * \log \frac{F_1(x)}{F_2(x)}$ berechnet und für stetige Verteilungen mit den Dichtefunktionen f_1 und f_2 nach $KL(F_1, F_2) = \int f_1(x) * \log \frac{f_1(x)}{f_2(x)} dx$.

Java Foundation Classes (JFC) Die JFC bilden eine Sammlung von Java-Klassen zur Erstellung Betriebssystem-übergreifender GUIs.

Just-In-Time-Compiler (JIT-Compiler) Ein Just-In-Time-Compiler übersetzt den Zwischencode (Bytecode) einer virtuellen Maschine während der Laufzeit in den nativen Maschinencode der ausführenden Hardwarearchitektur und gewährt damit einen leistungsmäßigen Vorteil im Vergleich zu einem reinen Interpreter.

Java Native Interface (JNI) Das JNI ist eine standardisierte Schnittstelle (API) von Java für den Zugriff auf native Funktionen und Bibliotheken [Sun04a].

Java Specification Requests (JSR) JSRs sind wesentlicher Teil des Java Community Process (JCP), der die Evolution der Java-Plattform bestimmt bzw. voranbringt. JSRs spezifizieren einen bestimmten Aspekt der Java-Technologie und können von einzelnen Personen, Organisationen, Gruppen oder Unternehmen eingereicht werden. Finden sie entsprechenden Widerhall in der Community können sie nach einem gewissen technischen Reifeprozess der Spezifikation und einer Referenzimplementierung auch Eingang in das Release neuer Java-Plattformen finden. [Sun04b]

Java Virtual Machine (JVM) Java Virtual Machine - virtuelle Maschine von Java, die den Java-Bytecode in Maschinencode der jeweiligen Architektur übersetzt und ausführt.

k-Nearest-Neighbor Klassifikation (kNN) Das kNN-Klassifikationsverfahren stellt eine der einfachsten Klassifikationsmethoden dar. Einem Datenpunkt unbekannter Klassenzugehörigkeit wird die Klasse zugeordnet, der die Mehrheit der k nächstgelegenen Datenpunkte bekannter Klassenzugehörigkeit angehört (siehe [HTF01]).

Liquid-Chromatography (LC) Die Flüssigchromatographie (Liquid-Chromatography) stellt eine Form der Chromatographie dar, also ein Verfahren zur Stofftrennung eines Stoffgemisches zwischen zwei verschiedenen Phasen, bei der die mobile Phase eine Flüssigkeit und die stationäre Phase ein Feststoff oder eine Flüssigkeit ist.

Lines of Code (LoC) LoC oder auch LOC stellt eine Metrik für den Umfang bzw. Komplexität eines Softwareprojektes dar. Es handelt sich dabei lediglich um eine ungefähre Abschätzung, da der Bezug zwischen der Anzahl der Programmzeilen und Komplexität sehr grob ist und von vielen Randbedingungen abhängt, wie beispielsweise der verwendeten Programmiersprache.

Level of Detail (LOD) Als Level of Detail (Detailstufen) bezeichnet man die Ebene der Detaildarstellung bis zu der eine Visualierungsmethode auflöst.

Locally Weighted Scatterplot Smoother (LOWESS) Mit LOWESS oder auch LOESS wird ein Verfahren der lokalen Regression bezeichnet. Hierbei wird ein einfaches lineares oder polynomielles (niedrigen Grades) Modell durch die lokale, abschnittsweise Anwendung der Methode der kleinsten Quadrate (Least-Square-Fit) an die Daten angepasst (siehe [Cle79, Cle81]).

Lipopolysaccharid (LPS) LPS sind aus einem hydrophilen Polysaccharid- und einem lipophilen Lipidanteil aufgebaut und bilden einen wesentlichen Bestandteil der Zellmembran von gramnegativen Bakterien. Sie werden beim Zerfall der Bakterien frei, wirken als Endotoxin und stimulieren damit eine Immunreaktion des Organismus.

LR-Zerlegung (LR) Als LR-Zerlegung (auch mit LU-Zerlegung bezeichnet) einer Matrix $A \in \mathbb{R}^{n \times n}$ bezeichnet man das Produkt $A = LR$, bestehend aus der unteren $L \in \mathbb{R}^{n \times n}$ und der oberen Dreiecksmatrix $R \in \mathbb{R}^{n \times n}$. Mitunter muss aus Gründen der numerischen Stabilität noch eine Permutationsmatrix P eingeführt werden $PA = LR$. Eine LR-Zerlegung kann mit Hilfe eines modifizierten Gauß'schen Eliminationsverfahrens gewonnen werden. (siehe [Wal03] Band 3 S. 340)

Look-Up-Table (LUT) Als Look-Up-Table (LUT) bezeichnet man eine Datenstruktur, ein mehrdimensionales oder assoziatives Feld, das die aufwendige Berechnung eines Wertes in einen einfachen indexbasierten Zugriff auf seine Elemente ersetzt. LUT's sind eine Möglichkeit zur Optimierung der Geschwindigkeit von Algorithmen, wenn der Speicherzugriff schneller als die Berechnung ist. Unter Umständen kann auch Speicher optimiert werden, wenn der Indexzugriff die Berechnung mit komplexen Datentypen ersetzt.

Microarray Genexpression - Markup Language (MAGE-ML) MAGE-ML ist ein von der *Object Managment Group* (OMG) verwalteter Standard eines Datenaustauschformats für die Daten eines Microarrayexperimentes [MGE08]. MAGE-ML bildet das in UML formulierte MAGE-OM in XML ab.

Microarray Genexpression - Object Model (MAGE-OM) MAGE-OM ist ein von der OMG verwalteter Standard eines UML-basierten Datenaustauschmodells für alle Ergebnisse und Informationen rund um ein Microarray-Experiment ([MGE08]). Inhaltliche Grundlage des Entwurfes bildet MIAME.

Markov Chain Monte-Carlo (MCMC) Eine Gruppe von Verfahren bei der mit Hilfe einer jeweils geeigneten Markow-Kette der Verlauf der gesuchten Verteilungsfunktion abgetastet wird. MCMC-Methoden stellen aktuell die grundlegende Implementation zur Lösung von Fragestellungen in der Bayesschen Statistik dar [GL06, GHM04].

Minimal Information About a Microarray Experiment (MIAME) Der MIAME Standard definiert den Inhalt aller notwendigen Informationen, um die Ergebnisse eines Microarray-Experiments eindeutig zu interpretieren bzw. es reproduzieren zu können (siehe [BHQ+01] und [MGE07]).

Meta Information Object (MIO) Als Meta Information Objects bzw. Meta-Informationsobjekte werden innerhalb von Mayday Container-Objekte bezeichnet, die zusätzlich geladenen oder dynamisch im Programmablauf erzeugte, sekundäre Informationen zu einem Array-Feature enthalten. Beispiele dafür sind textliche Annotationen, Klassifikatoren oder Evidenz-Schätzer, die innerhalb von MIOs gespeichert werden.

Mass-Spectrometry (MS) Die Massenspektrometrie (Mass-Spectrometry) ist ein wichtiges Verfahren der analytischen Chemie zur Detektion von Elementen und Verbindungen sowie zur Strukturaufklärung. In der Massenspektrometrie wird das Ladungsmassenverhältnis $\frac{m}{q}$ von Elementen, Molekülen oder Molekülbruchstücken bestimmt. Das Auftreten und die Verteilung charakteristischer Massefragmente einer zu untersuchenden Probe lässt den Schluss auf die enthaltenen Substanzen zu.

Model View Controller (MVC) Das MVC-Paradigma ist stellt ein sogenanntes Meta-Pattern dar, d.h. seine Bedeutung reicht über die eines normalen Pattern bzw. Musters hinaus [Ree79]. Es beschreibt die Trennung von Model, d.h. den vorliegenden Daten, Controller, d.h. den Anwendungsregeln zur Manipultation der bzw. Interaktion mit den Daten und dem View, der Darstellung der Daten. Diese Trennung der verschiedenen Aspekte erlaubt den Entwurf einer anpassungsfähigeren und leichter wartbaren Anwendung. Das MVC-Paradigma wird, in einer mehr oder weniger strikten Form, in den gängigen GUI-Bibliotheken angewendet, wie beispielsweise in den JFCs.

Wilcoxon-Mann-Whitney-Test (WMW-Test) Der Wilcoxon-Mann-Whitney-Test ist ein nichtparametrischer, statistischer Test auf Gleichheit der zentralen Tendenz zweier unabhängiger Stichproben (siehe Abschnitt 5.4 dieser Arbeit, sowie [SH06, Wil45, MW47]).

National Center for Biotechnology (NCBI) Das NCBI stellt das zentrale, amerikanische Institut für die Verarbeitung und Datenhaltung in der Molekularbiologie dar. Neben einer ganzen Reihe wichtiger öffentlicher Datenbanken für die Molekularbiologie wird auch Software entwickelt, wie zum Beispiel der sogenannte NCBI-BLAST. Das NCBI wurde 1988 gegründet und ist in Bethesda angesiedelt.

Network File System (NFS) Das NFS stellt ein UNIX-Netzwerkprotokoll zum Zugriff auf Dateien und Verzeichnisse über das Netzwerk hinweg dar.

National Institute of Standards and Technology (NIST) Bundesbehörde der Vereinigten Staaten zur technologischen Verwaltung innerhalb des Handelsministeriums und zur Koordination von Standardisierungsprozessen [Nat07].

On Line Analytical Processing (OLAP) Online Analytical Processing zählt neben dem Data Mining (siehe dort) zu den Methoden der Extraktion von Wissen aus umfangreichen Datenbeständen. Dabei werden an das OLAP-System Hypothesen formuliert, die durch dieses bestätigt oder falsifiziert werden. Wesentlich für OLAP-Systeme ist die ihnen zugrunde liegende multidimensionale Sicht auf die aus der operativen Datenbank extrahierten Daten (Datacube).

Object Managment Group (OMG) Die Object Management Group ist eine 1989 gegründete Vereinigung von Firmen und Organisationen, deren Anliegen die Festlegung herstellerübergreifender und systemunabhängiger Standards für die Objektorientierte Programmierung ist. Sehr bekannte und breit eingesetzte Beispiele für Standards der OMG sind die UML und die *Common Object Request Broker Architecture* (CORBA).

Object Oriented Programming (OOP) Als Object Oriented Programming (Objektorientierte Programmierung) bezeichnet man die Softwareentwicklung nach

dem objektorientierten Programmierparadigma. Derzeit stellt es das am weitesten verbreitete Programmierparadigma dar und betont vor allem die Flexibilität und Wiederverwendbarkeit von Entwürfen und erstellter Software. Zentral ist dabei die enge Kopplung zwischen Daten und ihren zugehörigen Funktionen bzw. Methoden in einem Objekt. [Bal04]

Open Services Gateway initiative (OSGi) Die OSGi Alliance stellt einen Zusammenschluß mehrerer Firmen und Organisationen dar, die eine dynamische, Hardware- und Hersteller-unabhängige, offene Service-Plattform nach dem SOA-Paradigma spezifizieren und verwalten [OSG08, OSG07] (siehe Abbildung 4.8 auf Seite 66). Die OSGi Alliance besteht seit 1999 und zielt mit ihrem Komponentenmodell auf einen sehr breiten Einsatzbereich, der von eingebetteten, leistungsarmen Systemen in Geräten und Fahrzeugen, über Mobiltelefone bis hin zu leistungsfähigen Desktoprechnern reicht. Mittlerweile ist OSGi Release 4.1 als *Java Specification Requests* (JSR) 291 (Dynamic Component Support for Java SE) Teil des Java Community Process (JCP) und als offizielles dynamisches Komponentenmodell für Java angenommen.

Principal Components Analysis (PCA) Die Hauptkomponentenanalyse (Principal Components Analysis, Hotelling Transformation, Karhunen-Loève Transformation) ist ein Verfahren zur Dimensionsreduzierung multidimensionaler Daten $X \in \mathbb{R}^{n \times m}$. Mathematisch stellt sie eine orthogonale, lineare Transformation der Daten in ein neues Koordinatensystem dar, dessen erste Koordinatenachse die Raumrichtung besitzt, in der die Daten die höchste Varianz aufweisen. Alle folgenden Koordinatenachsen stehen stets senkrecht auf den vorherigen und weisen in die Raumrichtung mit dem nächsthöheren Varianzgehalt. Sei X_z die zeilenzentrierte Datenmatrix, d.h. $E(X_{i.}) = 0$, dann erhält man als neue Koordinaten der Daten $Y = X_z V$. Dies ergibt sich aus der Eigenwertzerlegung der Kovarianzmatrix $S = \frac{1}{n-1} X_z^T X_z$: $S = \frac{1}{n-1} V \Sigma^2 V^T$ mit $X_z = U \Sigma V^T$ (siehe SVD) .

Polymerase Chain Reaction (PCR) Bei dere PCR handelt es sich um eine Technik zur Amplifizierung von DNA mit Hilfe von sequenzspezifischen Primern. Das Verfahren besteht aus einer Abfolge meherer DNA-Synthese-Zyklen, gefolgt von einer kurzen Wärmebehandlung zur Auftrennung der komplementären Basenstränge (siehe S. 508f in [AJL+02]). Die Amplifizierung verläuft dabei exponentiell 2^n mit der Anzahl der Zyklen n.

Parallelkoordinatenplot (PKP) Der Parallelkoordinatenplot stellt eine mögliche Form dar, hochdimensionale Daten zu visualisieren. Dabei werden die einzelnen Dimensionen als parallele Achsen nebeneinander aufgereiht und die Datenpunkte durch einen Linienzug über alle Achsen hinweg eingetragen. [Ins85]

Posttraumatic Stress Disorder (PTSD) Unter einer posttraumatischen Belastungsstörung (PTBS – engl. PTSD) versteht man die Entwicklung einer Vielzahl von psychischen und psychosomatischen Symptomen als Langzeitfolge eines Traumas oder einer Reihe von Traumata. Das oder die auslösende Traumata stellen ein Ereignis von außergewöhnlich großer psychischer Belastung dar, wie die Erfahrung von Lebensgefahr, starker Körperverletzung oder tiefster Verzweiflung (siehe Kapitel 9 auf Seite 177).

Quartet Mining (QM) QM stellt eine Methode für die robuste Merkmalsselektion von Genexpressionsprofilen bei binären Klassifikationen dar und wurde im Rahmen von [Sch07] als Adaption des in der Phylogenie bekannten Quartet Mapping [NSvH01] entwickelt (siehe Abschnitt 4.3.9).

QR-Zerlegung (QR) Als QR-Zerlegung einer Matrix $A \in \mathbb{R}^{m \times n}$ bezeichnet man das Produkt $A = QR$, bestehend aus der orthogonalen Matrix $Q \in \mathbb{R}^{m \times m}$ und einer oberen Dreiecksmatrix $R \in \mathbb{R}^{m \times n}$. Besitzt A vollen Spaltenrang $Rang(A) = n$, so existiert eine eindeutige QR-Zerlegung mit $r_{ii} \geq 0$. In der Numerik werden aufgrund der Stabilität häufig Householdertransformationen benutzt, um eine QR-Zerlegung zu finden. (siehe [Wal03] Band 4 S. 295)

Ribonukleinsäure (RNA) RNA ist ein Polynukleotid, das an der Umsetzung der genetischen Information in Proteine wesentlich in Form der mRNA (Boten-RNA) und tRNA (Transfer-RNA) beteiligt ist (vgl. Abschnitt 1.1 auf S. 3). Neben diesen gibt es jedoch weitere Formen von RNA und Zellprozesse an denen RNA wesentlich beteiligt ist.

Ribonuklease (RNase) Ribonukleasen sind Enzyme aus der Gruppe der Nukleasen und katalysieren die Umsetzung von Ribonukleinsäuren in kleinere Moleküle. Sie unterteilen sich in Endo- und Exoribonukleasen.

Rank Product (RP) Rank Product bzw. Rang Produkt, Methode zur Detektion differentiell exprimierter Gene [BAAH04]

Deutsches Ressourcenzentrum für Genomforschung GmbH (RZPD) Das **R**essourcen**z**entrum **P**rimär**d**atenbank (RZPD) war ein in Berlin und Heidelberg angesiedeltes Dienstleistungzentrum für die Genomforschung. Es entwickelte sich zeitweise zur weiltweit größten Datenbank für genetsche Klone und war Kooperationspartern wichtiger nationaler und internationaler Forschungsprojekte. Mitte 2007 ging es in den beiden Firmen ImaGenes und ATLAS Biolabs GmbH auf.

Statistical Analysis of Microarrays (SAM) Bei SAM handelt es sich um eine robuste Adaption der t-Statistik, um differentielle Genexpression zwischen zwei verschiedenen Klassen oder Zuständen I und U zu detektieren [TTC01]. SAM korrigiert den Wert der t-Statistik für das Gen i mit Hilfe eines s_0, das aus der

Gesamtheit des Microarrayexperimentes geschätzt wird: $\frac{x_I(i)-x_U(i)}{s_{IU}(i)+s_0}$. Dadurch wird das Gesamtrauschen des Experimentes in jede einzelne Genbetrachtung integriert und es verringert sich die Wahrscheinlichkeit, ein Gen lediglich aufgrund einer zufällig sehr kleinen (gepoolten) Standardabweichung $s_{IU}(i)$ als differentiell exprimiert bzw. als guten Klassenprädiktor zu selektieren. Die Beurteilung der Signifikanz eines Wertes erfolgt anhand eines Permutationstests.

Software Configuration Management (SCM) Beim SCM (Softwarekonfigurationsmanagement) handelt es sich um alle Tätigkeiten, die mit der Erstellung von Software zu tun haben, wie beispielsweise Versionsmanagement, Konfliktauflösung, Zugriffskontrolle, Dokumentation des ganze Entwicklungprozesses, Prozessverfolgung und Änderungsmanagement. Es gibt sowohl eine Reihe kommerzieller wie auch Open-Source-Werkzeuge zur Unterstützung des SCM, wie beispielsweise CVS [CVS08], Subversion [The08] oder höher integrierte wie Trac [Tra08] oder DrProject [DrP08].

Symmetrisches Multiprozessing (SMP) Symmetrisches Multiprozessing (Symmetric Multiprocessing) stellt eine Multiprozessorarchitektur dar, bei der identische Prozessoren auf einen gemeinsam genutzten Hauptspeicher zugreifen. Ein SMP-System erlaubt die Zuweisung und Verschiebung jeder einzelnen Task auf jede beliebige CPU, um die Auslastung des Gesamtsystems zu optimieren.

Single-Nucleotide Polymorphism (SNP) Als SNP bezeichnet man die Variation einer einzigen Base an einem bestimmten Locus der DNA, die in einem hinreichend hohen Anteil der Population vorkommt. SNPs stellen den Großteil der genetischen Variation einer Population dar und werden als erfolgreiche Punktmutationen interpretiert (vgl. Seite 464 in [AJL+02]).

Service Oriented Architecture (SOA) Unter einer Service-orientierten Architektur versteht man in der Informatik das Konzept der Zerlegung der Funktionalität einer Anwendung in einzelne Services, die von einzelnen Einheiten eines sehr lose gekoppelten Systems erbracht werden. Jeder Service liegt in der Verantwortlichkeit des jeweiligen Teilsystems, ist in diesem Sinne vollständig und kann über Systemgrenzen und ein Netzwerk hinweg flexibel in eine neue Anwendung integriert werden [Org07]. SOA wird als der nächste evolutionäre Schritt zum Paradigma der verteilten und der modularen Programmierung gesehen.

Self Organizing Maps (SOM) Selbstorganisierende Karten gehören zu den künstlichen neuronalen Netzen und ermöglichen ein unüberwachtes Lernen. Anhand einer Lernstichprobe wird eine sogenannte Karte (Map), eine Repräsentation des Eingaberaums bestimmter Topologie in einem Raum niedriger Dimensionalität (oft zweidimensional) trainiert, die eine Klassifikation der hochdimensionalen Eingabedaten erlaubt. SOMs sind deshalb gut geeignet, hoch-

dimensionale Daten in niedrigeren Dimensionen zu visualisieren ([HTF01] S. 480 ff).

Structured Query Language (SQL) SQL stellt einen Sprachstandard für die Abfrage, die Manipulation und die Definition von Daten relationaler DBMS dar. Ihre breite Unterstützung in DBMS machen Anwendung unabhängig von bestimmten Datenbankprodukten und gestatten so einen Austausch zwischen bzw. eine flexible Kombintation mit unterschiedlichen Datenbank-Backends. SQL-92 bzw. SQL2 wird von gängigen Systemen größtenteils unterstützt. Neuere Standards wie SQL:1999 bzw. SQL3 oder SQL:2003 werden nur teilweise unterstütz.

Singular Value Decomposition (SVD) Die Singulärwertzerlegung (Singular Value Decomposition) einer Matrix $A \in \mathbb{R}^{n \times m}$ ist die Zerlegung der Matrix A in das Produkt A $= U\Sigma V^T$, wobei $U \in \mathbb{R}^{n \times m}$ und $V \in \mathbb{R}^{m \times m}$ orthogonale Matrizen sind und $\Sigma \in \mathbb{R}^{mxm}$ eine Diagonalmatrix darstellt, deren von Null verschiedenen Elemente aus den singulären Werten von A gebildet werden $\text{diag}(\Sigma) = (\sigma_1, \ldots, \sigma_r, 0, \ldots)^T$ mit $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$. Der Rang der Matrix A ist r. Die singulären Werte σ_i von A hängen mit den Eigenwerten λ_i der Matrix $A^T A \in \mathbb{R}^{m \times m}$ über $\sigma_i = \sqrt{\lambda_i}$ zusammen. (siehe [Wal03] Band 5 S. 34/36)

Support Vector Machine (SVM) Supportvektormaschinen sind ein Verfahren des maschinellen Lernens. SVMs bestimmen mit Hilfe der Supportvektoren, Datenpunkten an den Grenzbereichen der einzelnen Klassen, möglichst Breite zwischen den Klassen separierende Hyperflächen. Für den Fall nicht linear trennbarer Klassen wird ein Kernel eingeführt, der die Daten in einen höherdimensionalen Raum transformiert, in dem diese dann linear separierbar sind (vgl. [HTF01] S. 371)).

Standard Widget Toolkit (SWT) Das SWT ist eine Bibliothek zur Erstellung von GUIs in Java. Entwickelt im Rahmen des Eclipse Projektes stellt es möglichst dünne Wrapper-Klassen für die nativen Widgets des jeweiligen Betriebssystems zur Verfügung [Ecl08c].

Unified Modeling Language (UML) Die Unified Modeling Language ist ein umfangreicher Sprachstandard zur Beschreibung und Modellierung von Softwaresystemen. Ein wesentliches und wichtiges Element von UML sind die graphischen Notationen. Entwickelt von Booch, Rumbaugh und Jacobson bei der Firma *Rational Software Corporation* wurde sie 1997 von der OMG als Standard verabschiedet ([Bal04, BRJ99]).

Uniform Resource Identifier (URI) URIs werden zur eindeutigen Identifikation von abstrakten oder physischen Ressourcen benutzt, wie zum Beispiel Dateien, Webseiten oder Webservices. Sie folgen einem definierten Namensschema und finden im Zusammenhang mit dem Internet breite Anwendung.

Uniform Resource Locator (URL) Die URL stellt eine Unterart des *Uniform Resource Identifier*s (URI) dar, bei der über das Netzwerkprotokoll (z.B. `http`) und den Ort im Netzwerk (`www.beispiel.de`) eine Ressource beschrieben wird.

Extensible Markup Language (XML) XML ist eine flexible, erweiterbare Auszeichnungssprache zur Beschreibung hierarchisch strukturierter Daten in Form einfach interpretierbarer Textdateien. Sie hat sich als Standard zum Austausch von Informationen zwischen verschiedenen Informationssystemen etabliert, insbesondere in heterogenen Netzen über System- und Plattformgrenzen hinweg.

Extreme Programming (XP) XP ist eine agile Softwareentwicklungsmethode, die sich durch flexible, schnelle und kunden-orientierte Entwicklung auszeichnet und sich in den letzten Jahren immer mehr in Entwicklungsteams durchsetzt. Es gibt einige wichtige, praxisbewährte Prinzipien, die verwirklicht werden müssen, ansonsten ist XP jedoch sehr wenig formalisiert [BA04].

Extensible Stylesheet Language (XSL) Unter XSL versteht man eine Sammlung von Sprachen zur Beschreibung des Aussehens bzw. Layouts (Stylesheets) von XML-Dokumenten. Dies ist für die Darstellung des XML-Dokuments in unterschiedlichem Kontext wichtig, zum Beispiel das jeweilige Layout für die Druck- und die Bildschirmausgabe. Ein wichtiger Teil der XSL stellt die XSLT dar.

XSL Transformation (XSLT) Die XSLT ist ein Teil der *Extensible Stylesheet Language* (XSL). Mit ihrer Hilfe lassen sich Transformationen von XML-Dokumenten programmieren. Dazu lassen sich in sogenannten XML-konformen XSLT-Stylesheets Umwandlungsregeln definieren, die XSLT-Prozessoren für die Durchführung der Umwandlung benutzen.

Zufallsvariable (ZV) Eine Zufallsvariable Z ist die interessierende Größe bei einem Zufallsexperiment. Der Ausgang eines Zufallsexperiments wird als Realisierung z der Zufallsvariable bezeichnet, die n-malige Realisierung als Stichprobe und als Grundgesamtheit die Menge aller möglichen Realisierungen der Zufallsvariable (vgl. S. 144f in [SH06]).

Glossar

A-Posteriori-Wahrscheinlichkeit Unter der A-Posteriori-Wahrscheinlichkeit oder auch Statistischen Wahrscheinlichkeit versteht man die Wahrscheinlichkeit, die einem Ereignis aus der Beobachtung (beobachtete Statistik) heraus empirisch zugeordnet werden kann (vgl. auch A-Priori-Wahrscheinlichkeit).

Behrens-Fisher-Problem Das Behrens-Fisher-Problem bezeichnet in der Statistik die Fragestellung der Differenz der Erwartungswerte $E(X) = \mu_X$ und $E(Y) = \mu_Y$ der Verteilungsfunktionen einer Zufallsvariablen innerhalb zweier Stichproben X und Y bei Hypothesentests und Intervallschätzungen, wenn die Varianzen als ungleich angenommen werden $\sigma_X^2 = V(X) \neq V(Y) = \sigma_Y^2$ ([Beh29, Fis39], S.382 in [SH06]).

Bindung Als Bindung (engl. Tie) bezeichnet man das Auftreten von gleich großen Werten bei der Rangsortierung von Zahlen. Allen Zahlen der Bindung ordnet man im Allgemeinen ihren mittleren Rang zu (Mittelrangmethode).

Bootstrap Bootstrap stellt ein Resampling-Verfahren dar, um die Verteilung einer Statistik auf der Grundlage einer einzigen Stichprobe abzuschätzen. Hierbei werden durch Ziehen mit Zurücklegen neue Stichproben aus der einen vorhanden, „realen“Stichprobe generiert und für diese die Statistik neu berechnet. Als Ergebnis erhält man eine Abschätzung für die Verteilung der betrachteten Statistik.

ChIP-on-Chip Unter ChIP-on-Chip oder auch ChIP-Chip versteht man eine Kombination aus ChIP und Microarray-Technologie, um hochskalierbar und kostengünstig Interaktionen zwischen Proteinen und DNA nachzuweisen und die zugehörigen Bindestellen zu finden. Von hohem Interesse sind dabei vor allem die Bindestellen von Transkriptionsfaktoren und Proteinen, die in die Replikation der DNA involviert sind ([HL04] und Abschnitt 2.1.1).

Computational Statistics Sehr umfangreiches Gebiet der Statistik, das durch einen hohen Einsatz von Rechentechnik charakterisiert ist. (siehe hierzu Abschnitt 5.2 oder [GHM04]).

Data Mining Gewinnung von Wissen aus großen Datenmengen. „The science of extracting useful information from large data sets or databases is known as data mining.“[HMS01]

Dispersion Die Dispersion oder auch Streuung (engl. Dispersion) charakterisiert im Kontext der Statistik bzw. Wahrscheinlichtstheorie die Verteilung der Stichprobenelemente, bzw. der Realisierungen einer Zufallsvariablen um das Zentrum ihrer Häufigkeits- bzw. Wahrscheinlichkeitsverteilung herum. Gängige Maße der despritiven Statistik sind beispielsweise die empirische Standardabweichung, die empirische Varianz, die Spannweite, die mittlere quadratische Abweichung, der Interquartilabstand (IQR) und verschiedene andere Quantilmaße wie Dezile und Perzentile. [SH06, BHPT95]

Dye-Swap Als einen Dye-Swap bezeichnet man, im Zusammenhang mit der Microarrayanalyse, ein Replikat eines zweikanaligen Microarray-Experiments bei dem die farbliche Markierung getauscht wurde. Es dient einer Korrektur potentieller Verzerrungen der Resultate durch spezifische Eigenschaften der verwendeten Farbstoffe.

Erzeugendenfunktion Unter einer erzeugenden Funktion der Folge $\{a_n\}_{n\in\mathbb{N}_0}$ versteht man in verschiedenen Gebieten der Mathematik die Potenzreihe $F_{a_n}(z) = \sum_{n=0}^{\infty} a_n z^n$ mit $z \in \mathbb{C}$. Für ein Beispiel in der Zahlentheorie siehe unter Partitionsfunktion. ([Wal03] Bd. 2 S. 80).

Factory Als Factory oder Fabrik bezeichnet man im Softwaredesign ein Erzeugungsmuster (Creational Pattern), das eine Schnittstelle für die Erzeugung von Objekten definiert (siehe auch Entwurfsmuster). Die konkrete Erzeugung, Initialisierung und Konfiguration wird auf die (Unter-)Klassen delegiert, die diese Schnittstelle realisieren (siehe Seiten 18ff, 25ff in [ES04] bzw. 107ff, 131ff in [GHJV96]).

Gelelektrophorese Die Gelektrophorese stellt eine vielfach angewendete, analytische Trennmethode in der Chemie und Molekularbiologie dar. Hierbei wird die unterschiedliche Bewegungsgeschwindigkeit von Molekülen eines Gemisches innerhalb eines Gels unter dem Einfluß eines elektrischen Feldes ausgenutzt. Beeinflußt wird die Bewegungsgeschwindigkeit und damit die Art der Auftrennung des Stoffgemisches von der jeweiligen Molekülgröße und Ladungsverteilung.

Gene-Finding Vorhersage der Genstruktur innerhalb einer genomischen Sequenz.

Genotyping Unter dem Genotyping versteht man die Bestimmung des genauen, vorliegenden Genotyps des Individuums einer bestimmten Art, deren genetische Basis durch Polymorphismus variiert (vgl. den Eintrag zu SNP und Abschnitt 2.1.2).

Grid Computing Ein Grid koordiniert und integriert verteilte Ressourcen, wie beispielsweise Rechner, Massenspeicher oder Anwendungen über offene Protokolle und Standards, um ebenfalls verteilten Nutzern Dienste konfigurierbarer höherer Komplexität und Wertigkeit anbieten zu können (siehe auch

[FKT01, FK99]). Der Begriff Grid soll die Analogie zum Stromnetz (Power Grid) zum Ausdruck bringen - und damit zu abrufbarer Rechenkapazität aus dem Netz.

heteroskedastisch Als heteroskedastisch (varianzheterogen) bezeichnet man mindestens zwei Zufallsvariablen, deren zugrunde liegende Verteilungen nicht die gleiche Varianz besitzen.

homoskedastisch Als homoskedastisch (varianzhomogen) bezeichnet man mindestens zwei Zufallsvariablen, deren zugrunde liegende Verteilungen die gleiche Varianz besitzen.

Inhouse-System Software-System zur ausschließlich internen Nutzung. Ein System, das nur für die Nutzung durch die Arbeitsgruppe, dem Labor oder der Institution gedacht ist und deshalb auch von diesen selbst aufgesetzt und administriert werden muss.

Jackknife Jackknife stellt, ähnlich der Bootstrap-Methode, ein Resampling-Verfahren dar, um die Verzerrung bzw. den Bias und die Varianz einer Statistik abzuschätzen. Hierbei werden durch das jeweilige Weglassen eines Stichprobenwertes neue Stichproben erstellt und für diese die Statistik berechnet. Jackknife ist nicht so allgemein anwendbar wie Boostrapping.

k-Means Clustering Unter k-Means Clustering versteht man ein Partitionierungsverfahren, bei dem ein vorhandener Datensatz in k Cluster aufgeteilt wird. Dabei wird aus einer Anfangsinitialisierung von k Clusterzentren (Zentroiden) in einem iterativen Prozess von abwechselnder Zuordnung der zu diesen Zentren nächstgelegenen Datenpunkte und Neubestimmung der Zentren, ein vorher festgelegtes Konvergenzkriterium bis zu einer im Vorfeld definierten Schranke optimiert (siehe auch [HTF01]).

Lagemaß Die zentrale Tendenz, das Lagemaß oder auch der Lageparameter (engl. Central Tendency), gibt im Kontext der Statistik, bzw. Wahrscheinlichkeittheorie, die Lage des Zentrums einer Häufigkeits- bzw. Wahrscheinlichkeitsverteilung an. Gängige Maße der deskriptiven Statistik sind beispielsweise das arithmetische Mittel, der Median, der Modus, das geometrische und das harmonische Mittel. In der Wahrscheinlichkeitstheorie spricht man vom Erwartungswert einer Zufallsvariablen. [SH06, BHPT95]

Least-Square-Fit Durchführung der Regression nach der Methode der kleinsten Quadrate.

Machine Learning Verschiedenste Methoden zur Gewinnung von Wissen aus großen Datenmengen. [HTF01, WF05]

Metagenomik Die Metagenomik (engl. Metagenomics oder auch Ecogenomics) benutzt moderne molekularbiologische Verfahren zur Analyse und Charakterisierung ganzer Ökosysteme, z.B. in [SCC03, TvMK+05]. Ein großer Vorteil dieser Methode besteht darin, auch nicht im Labor kultivierbare Organismen zu erfassen. Der Begriff selbst setzt sich auch Metaanalyse und Genomik zusammen.

zentrales Moment Als zentrales Moment k-ter Ordnung werden in der Wahrscheinlichkeitstheorie die Parameter $\mu_k = E[(X - m_1)^k]$ mit $m_1 = E(X)$ und $k = \{1, 2, \ldots\}$ einer Verteilung der Zufallsvarbiablen X bezeichnet (S. 75 in [BHPT95]). Die Varianz entspricht dem zentralen Moment der zweiten Ordnung $V(X) = \mu_2 = m_2 - m_1^2 = E(X^2) - E^2(X)$ (vgl. auch Moment).

Monte-Carlo-Methode Stochastische Methode zur Schätzung von Lösungen für Integralgleichungen, Integrodifferentialgleichungen, Zustandsgleichungen oder von Verteilungsfunktionen. [MU49]

Normalisierung Im Kontext der Genexpressionsanalyse mit Hilfe von Microarrays versteht man unter der Normalisisierung die rechnerische Korrektur der gemessenen Expressionswerte um bekannte bzw. zu erwartende systematische Fehler zu vermindern oder zu eliminieren.

Overdraw-Problem Problem der gegenseitigen Verdeckung in Datendarstellungen

p-Value Der mit Hilfe einer Teststatistik berechneten Prüfgrösse T^* eines statistischen Tests kann eine Wahrscheinlichkeit zugeordnet werden, mit der, unter Gültigkeite der Nullhypothese H_0, gleich große oder größere Werte für T auftreten. Dieser Wahrscheinlichkeitswert $P_{H_0}(T^* \leq T)$ wird als p-Value (deut. P-Wert, aufgrund der weiten Verbreitung wird in dieser Arbeit der englische Begriff genutzt) bezeichnet und als Maß für die statistische Evidenz einer potentiellen Ablehnung der Nullhypothese H_0 interpretiert (vgl. S. 307 in [SH06] und für eine kritische Diskussion S.102 ff in [Tho07] sowie [SL85, Coh94, GGHG96, Sch96, SAW06]).

Partitionsfunktion Partitionsfunktionen geben die Anzahl an möglichen Zerlegungen einer natürlichen Zahl in Summanden an. In der einfachsten Form ist die Reihenfolge der Summanden gleichgültig. Kompliziertere Formen sind möglich, bei denen an die Summanden zusätzliche Forderungen gestellt werden. Sei $p(n)$ die Anzahl der Partitionen einer natürlichen Zahl n und $p(0) = 1, \{\}$; dann gilt $p(1) = 1 = \#\{\{1\}\}$; $p(2) = 2 = \#\{\{1,1\},\{2\}\}$;$p(3) = 3 = \#\{\{1,1,1\},\{1,2\},\{3\}\}$; $p(4) = 5 = \#\{\{1,1,1,1\},\{1,1,2\}\},\{1,3\},\{4\}\}$; ... und damit für seine erzeugende Funktion $F_{p_n}(z) = \prod_{n=1}^{\infty} \frac{1}{1-z^n} = 1 + 1x + 2x^2 + 3x^3 + 5x^4 + \ldots$ mit $z \in \mathbb{C} : |z| < 1$ ([Wal03] Bd. 2 S. 80).

Post-Hoc Test Als Post-Hoc Tests (teilweise auch als ungeplante Tests) werden Tests bezeichnet, die erst nach dem signifikanten Ergebnis eines allgemeineren, sogenannten Ominbustests, wie beispielsweise einer ANOVA oder dem Kurskal-Wallis-Test, über mehrere Gruppen angewendet werden. Ziel dieser Tests ist die Untersuchungen der paarweisen Unterschiede der einzelnen Gruppen.

QT Clustering Das QT-Clustering ist ein Clusterverfahren, bei dem im Vorfeld keine Clusteranzahl festgelegt werden muß. Vorgegeben wird ein *Quality Threshold* d, der den maximalen Durchmesser der zu findenden Cluster beschreibt. In einem Iterationsdurchgang werden von allen noch unzugeordneten Datenpunkten ausgehend, potentielle Cluster bis zum Erreichen von d bestimmt. Der mengenmäßig größte Cluster eines Durchgangs geht als endgültiger Cluster in das Ergebnis ein. Sind schließlich keine Datenpunkte mehr vorhanden, ist die vollständige Partitionierung erreicht (siehe [HKY99]).

Refactoring Als Refactoring oder Refaktorisieren bezeichnet man in der Informatik den Prozess, die interne Codestruktur eines Softwaresystems zu verbessern, ohne deren nach außen gezeigtes Verhalten zu ändern. Die gemachten Änderungen werden nach Schemen so durchgeführt, dass das Auftreten neuer Fehler minimal gehalten wird (siehe hierzu [Fow05]).

Resampling Unter Resampling versteht man die Erzeugung neuer „künstlicher“ Stichproben auf der Grundlage einer realen Stichprobe durch das Weglassen von Daten (Jackknife), das Ziehen von Stichprobenwerten mit Zurücklegen (Bootstrap) oder die zufällige Zuweisung von Klassen zu Stichprobenwerten (Permutationstests).

Slice-Analyse Die Slice-Analyse untersucht eine intensitätsbezogene z-Transformation (Transformation zur Standardnormalverteilung) der logarithmierten Intensitätsverhältnisse (Logratios M) der Spots bzw. Features eines Arrays, um Aussagen bezüglich der Signifikanzschwelle für als differentiell exprimiert anzusehende Gene machen zu können (siehe Abschnitt 7.3.2 auf Seite 162).

Southern-Blot Der Southern-Blot stellt eine molekularbiologische Untersuchungsmethode zur schnellen Indentifikation einer bestimmten Basensequenz innerhalb eines Nukleinsäuregemisches dar (DNA beim Southern- und RNA beim Nothern-Blot). Verfahrenstechnisch basiert er auf einer Kombination von Gelelektrophorese und sequenzspezifischer Hybridisierung mit markierten, komplementären Basensträngen. Er wurde 1975 von E.W. Southern vorgeschlagen [Sou75] und gilt, insbesonder in Form des Nothern-Blot, als Vorläufer der modernen Microarraytechnologie.

Tiling Array Tiling Arrays sind eine Form von Microarrays bei der die immobilisierten Sonden auf der Chip-Oberfläche entweder das ganze Genom oder

Teile (Contigs) davon überdecken. Beispiele für Tiling Arrays sind CGH-, SNP-Chips und Arrays für ChIP-on-Chip oder die Resequenzierung (vgl. Abschnitt 2.1 ab Seite 17).

Visual Analytics Feld der Visualisierung, das explizit eine interaktive, nutzergetriebene Visualisierung als Grundlage zum Erkennen von Mustern in Daten benutzt. „Visual Analytics is the science of analytical reasoning facilitated by interactive visual interfaces.“[Tho06]

Volcano-Plot Ein Volcano-Plot bildet den FC und seinen zugehörigen, logarithmierten Signifikanzwert (p-Value) in einem Scatterplot ab. Damit lässt sich die Validität eines gemessenen Verhälntisses sehr gut visuell beurteilen.

Literaturverzeichnis

Die Reihenfolge der Referenzen ergibt sich aus der alphabetisch Sortierung der Nachnamen der Autoren, beginnend mit dem Erstautor.

[Aff04] Affymetrix Inc. Data Sheet - GeneChip Human Genome Arrays. http://www.affymetrix.com/support/technical/datasheets/human_datasheet.pdf,(27.06.2007), 2004.

[Aff07] Affymetrix Inc. Data Sheet - Affymetrix Genome-Wide Human SNP Array 6.0. http://www.affymetrix.com/support/technical/datasheets/genomewide_snp6_datasheet.pdf,(27.06.2007), 2007.

[Aff08a] Affymetrix Inc. Data Sheet - NimbleExpress Array Programm. http://www.affymetrix.com/support/technical/datasheets/nimble_programs_datasheet.pdf, (18.05.2008), 2008.

[Aff08b] Affymetrix Inc. Products & Applications. http://www.affymetrix.com/products/index.affx, (10.01.2008), 2008.

[Agi07] Agilent Technologies (SiliconGenetics). GeneSpringGX 7.3.1, 2007. http://www.chem.agilent.com/scripts/pds.asp?lpage=27881 (20.04.2007).

[Agi08] Agilent Technologies Inc., Life Sciences and Chemical Analysis Group. DNA Microarrays. http://www.chem.agilent.com/Scripts/PCol.asp?lPage=494, (10.01.2008), 2008.

[AJL+02] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Number 978-0815340720. Taylor & Francis, 4 edition, Mar. 2002.

[Al 08] Al Danial. CLOC - Count Lines of Code. http://cloc.sourceforge.net, (23.04.2008), 2008.

[AMD06] AMD/ATI. *ATI CTM Guide - Technical Reference Manual Version 1.01*. AMD/ATI, http://ati.amd.com/companyinfo/researcher/documents/ATI_CTM_Guide.pdf, (22.10.2007), 2006.

[And72] D. F. Andrews. Plots of high dimensional data. *Biometrics*, 28:125–136, 1972.

[Apa08] Apache Software Foundation. The Apache Ant Project. `http://ant.apache.org/`, (23.04.2008), 2008.

[Asp48] A.A. Aspin. An Examination and Further Development of a Formula Arising in the Problem of Comparing Two Mean Values. *Biometrika*, 35(1/2):88 – 96, May 1948.

[AW49] A.A. Aspin and B.L. Welch. Tables for Use in Comparisons Whose Accuracy Involves Two Variances, Separately Estimated. *Biometrika*, 36(3/4):290 – 296, Dec. 1949.

[BA04] K. Beck and C. Andres. *Extreme Programming Explained. Embrace Change.* Number 978-0321278654. Addison-Wesley Longman, 2 edition, 2004.

[BAAH04] Rainer Breitling, Patrick Armengaud, Anna Amtmann, and Pawel Herzyk. Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. *FEBS Lett*, 573(1-3):83–92, Aug 2004.

[Bal04] Heide Balzert. *Lehrbuch der Objektmodellierung - Analyse und Entwurf mit der UML 2.* Number 978-3827411624. Spektrum Akademischer Verlag, 2. a. edition, Nov. 2004.

[BAS07] BASE-Team, University Lund. BASE - BioArray Software Environment, `http:// base1.thep.lu.se/index.phtml`, 2007. (20.04.2007).

[Bat07a] A. Bateman. Alan bateman's weblog - heap dumps are back with a vengeance! `http://blogs.sun.com/alanb/entry/heap_dumps_are_back_with`, (02.07.2007), June 2007.

[Bat07b] Florian Battke. A Processing Framework for Mayday. Studienarbeit, Universtät Tübingen, 2007.

[BDP+98] R.F. Boisvert, J.J. Dongarra, R. Pozo, K.A. Remington, and G.W. Stewart. Developing numerical libraries in Java. *Concurrency: Practice and Experience*, 10(11 – 13):1117 – 1129, Dec 1998.

[Beh29] W.V. Behrens. Ein Beitrag zur Fehlerberechnung bei wenigen Beobachtungen. *Landwirtschaftliche Jahrbücher*, 68:807 – 837, 1929.

[Ber03] Daniel Bernstein. Exercise assessment of transgenic models of human cardiovascular disease. *Physiol Genomics*, 13(3):217–226, May 2003.

[BH95] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Statist. Soc. B.*, 57:289–300, 1995.

[BH02] P. Baldi and G. W. Hatfield. *DNA Microarrays and Gene Expression. From Experiments to Data Analysis and Modeling.* Cambridge University Press, 2002.

[BH05] Rainer Breitling and Pawel Herzyk. Rank-based methods as a non-parametric alternative of the t-statistic for the analysis of biological microarray data. *J Bioinform Comput Biol*, 3(5):1171–1189, Oct 2005.

[BHPT95] O. Beyer, H. Hackel, V. Pieper, and J. Tiedge. *Mathematik fr Ingenieure, Naturwissenschaftler, ï¿½onomen und sonstige anwendungsorientierte Berufe: Wahrscheinlichkeitsrechnung und mathematische Statistik: Bd 17*, volume 17. B. G. Teubner Verlagsgesellschaft Stuttgart, Leipzig, 7. edition, 1995.

[BHQ+01] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C.A. Ball, and H.C. Causton et al. Minimum information about a microarray experiment (miame)-toward standards for microarray data. *Nat. Genet.*, 29:365–371, 2001.

[BIÅS03] B. M. Bolstadt, R. A. Irizzary, M. Åstrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19:185–193, 2003.

[Bio08a] Bioconductor Core Team. BioConductor. `http://www.bioconductor.org/`, 2008. (29.04.2008).

[Bio08b] BioDiscovery Inc. ImaGene - Leading-Edge Microarray Analysis Software. `http://www.biodiscovery.com/index/imagene`, (10.04.2008), 2008.

[Bit08] BitRock Inc. BitRock InstallBuilder. `http://bitrock.com/products_installbuilder_overview.html`, (21.04.2008), 2008.

[BL01] P. Baldi and A. D. Long. A bayesian framework for the analysis of microarray expression data: regularized t -test and statistical inferences of gene changes. *Bioinformatics*, 17(6):509–519, Jun 2001.

[BLA07] BLAS. BLAS (Basic Linear Algebra Subprograms). `http://www.netlib.org/blas/`,(25.06.2007), June 2007.

[BLB00] J. Bortz, G.A. Lienert, and K. Boehnke. *Verteilungsfreie Methoden in der Biostatistik*. Number 3-540-67590-6. Springer-Verlag Berlin Heidelberg New York, 2. edition, 2000.

[BLS00] R. Bergmann, J. Ludbrook, and W.P.J.M. Spooren. Different Outcomes of the Wilcoxon-Mann-Whitney Test from Different Statistics Packages. *The American Statistician*, 54(1):72 – 77, Feb. 2000.

[BM00] E. Brunner and U. Munzel. The Nonparametric Behrens-Fisher Problem: Asymptotic Theory and a Small-Sample Approximation. *Biometrical Journal*, 42(1):17 – 25, 2000.

[BMPP01] R.F. Boisvert, J. Moreira, M. Philippsen, and R. Pozo. Java and Numerical Computing. *Computing in Science & Engineering*, 3(2):18 – 24, Mar/Apr 2001.

[Bol04] W. M. Bolstad. *Introduction to Bayesian Statistics*. John Wiley & Sons, 1 edition, 2004.

[Bon36] C. E. Bonferroni. Teoria statistica delle classi e calcolo della probabilita. *Pubblicazioni del Instituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.

[BP07] R. Boisvert and R. Pozo. Java numerics group. `http://math.nist.gov/javanumerics/`, 2007. (04.06.2007).

[BR87] D.J. Best and J.C.W. Rayner. Welch's Approximate Solution for the Behrens-Fisher Problem. *Technometrics*, 29(2):205 – 210, May 1987.

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *Das UML-Bentutzerhandbuch*. Number 0201571684. Addison-Wesley-Longmann Verlag GmbH, 1. edition, 1999.

[BS88] W. Bains and G. C. Smith. A novel method for nucleic acid sequence determination. *J Theor Biol*, 135(3):303–307, Dec 1988.

[BS98] W. Baumgartner and P. Weiï¿½and H. Schindler. A Nonparametric Test for the General Two-Sample Problem. *Biometrics*, 54(3):1129 – 1135, Sep. 1998.

[BTW+07] Tanya Barrett, Dennis B Troup, Stephen E Wilhite, Pierre Ledoux, Dmitry Rudnev, Carlos Evangelista, Irene F Kim, Alexandra Soboleva, Maxim Tomashevsky, and Ron Edgar. Ncbi geo: mining tens of millions of expression profiles–database and tools update. *Nucleic Acids Res*, 35(Database issue):D760–D765, Jan 2007.

[Buc99] A. Di Bucchianico. Combinatorics, computer algebra and the Wilcoxon-Mann-Whitney test. *Journal of Statistical Planning and Inference*, 79(2):349 – 364, July 1999.

[bwc06] bwcon. bwcon: Sonderpreis it & life sciences 2006 verliehen. *bwcon:Meldungen*, 2006. `http://www.bwcon.de/bwcon_newsdetail.html?&L=0&encryptionKey=&tx_ttnews[backPid]=330&tx_ttnews[tt_news]=1205&cHash=4805f306e6`, (26.08.2006).

[BY01] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple hypothesis testing under dependency. *Annals of Statistics*, 29(4):1165–1188, 2001.

[CBS02] T. Connolly, C. Begg, and A. Strachan. *Datenbanksysteme – Eine praktische Anleitung zu Design, Implementierung und Management*. Number 3827320135. Addison-Wesley Verlag, 2002.

[Cer07] Cern. Colt Project. `http://dsd.lbl.gov/ hoschek/colt/index.html`, 2007. (09.07.2007).

[CFB+07] International HapMap Consortium, Kelly A Frazer, Dennis G Ballinger, David R Cox, David A Hinds, Laura L Stuve, Richard A Gibbs, John W Belmont, Andrew Boudreau, Paul Hardenbol, Suzanne M Leal, Shiran Pasternak, David A Wheeler, Thomas D Willis, Fuli Yu, Huanming Yang, Changqing Zeng, Yang Gao, Haoran Hu, Weitao Hu, Chaohua Li, Wei Lin, Siqi Liu, Hao Pan, Xiaoli Tang, Jian Wang, Wei Wang, Jun Yu, Bo Zhang, Qingrun Zhang, Hongbin Zhao, Hui Zhao, Jun Zhou, Stacey B Gabriel, Rachel Barry, Brendan Blumenstiel, Amy Camargo, Matthew Defelice, Maura Faggart, Mary Goyette, Supriya Gupta, Jamie Moore, Huy Nguyen, Robert C Onofrio, Melissa Parkin, Jessica Roy, Erich Stahl, Ellen Winchester, Liuda Ziaugra, David Altshuler, Yan Shen, Zhijian Yao, Wei Huang, Xun Chu, Yungang He, Li Jin, Yangfan Liu, Yayun Shen, Weiwei Sun, Haifeng Wang, Yi Wang, Ying Wang, Xiaoyan Xiong, Liang Xu, Mary M Y Waye, Stephen K W Tsui, Hong Xue, J. Tze-Fei Wong, Luana M Galver, Jian-Bing Fan, Kevin Gunderson, Sarah S Murray, Arnold R Oliphant, Mark S Chee, Alexandre Montpetit, Fanny Chagnon, Vincent Ferretti, Martin Leboeuf, Jean-François Olivier, Michael S Phillips, Stéphanie Roumy, Clémentine Sallée, Andrei Verner, Thomas J Hudson, Pui-Yan Kwok, Dongmei Cai, Daniel C Koboldt, Raymond D Miller, Ludmila Pawlikowska, Patricia Taillon-Miller, Ming Xiao, Lap-Chee Tsui, William Mak, You Qiang Song, Paul

K H Tam, Yusuke Nakamura, Takahisa Kawaguchi, Takuya Kitamoto, Takashi Morizono, Atsushi Nagashima, Yozo Ohnishi, Akihiro Sekine, Toshihiro Tanaka, Tatsuhiko Tsunoda, Panos Deloukas, Christine P Bird, Marcos Delgado, Emmanouil T Dermitzakis, Rhian Gwilliam, Sarah Hunt, Jonathan Morrison, Don Powell, Barbara E Stranger, Pamela Whittaker, David R Bentley, Mark J Daly, Paul I W de Bakker, Jeff Barrett, Yves R Chretien, Julian Maller, Steve McCarroll, Nick Patterson, Itsik Pe'er, Alkes Price, Shaun Purcell, Daniel J Richter, Pardis Sabeti, Richa Saxena, Stephen F Schaffner, Pak C Sham, Patrick Varilly, David Altshuler, Lincoln D Stein, Lalitha Krishnan, Albert Vernon Smith, Marcela K Tello-Ruiz, Gudmundur A Thorisson, Aravinda Chakravarti, Peter E Chen, David J Cutler, Carl S Kashuk, Shin Lin, Gonçalo R Abecasis, Weihua Guan, Yun Li, Heather M Munro, Zhaohui Steve Qin, Daryl J Thomas, Gilean McVean, Adam Auton, Leonardo Bottolo, Niall Cardin, Susana Eyheramendy, Colin Freeman, Jonathan Marchini, Simon Myers, Chris Spencer, Matthew Stephens, Peter Donnelly, Lon R Cardon, Geraldine Clarke, David M Evans, Andrew P Morris, Bruce S Weir, Tatsuhiko Tsunoda, James C Mullikin, Stephen T Sherry, Michael Feolo, Andrew Skol, Houcan Zhang, Changqing Zeng, Hui Zhao, Ichiro Matsuda, Yoshimitsu Fukushima, Darryl R Macer, Eiko Suda, Charles N Rotimi, Clement A Adebamowo, Ike Ajayi, Toyin Aniagwu, Patricia A Marshall, Chibuzor Nkwodimmah, Charmaine D M Royal, Mark F Leppert, Missy Dixon, Andy Peiffer, Renzong Qiu, Alastair Kent, Kazuto Kato, Norio Niikawa, Isaac F Adewole, Bartha M Knoppers, Morris W Foster, Ellen Wright Clayton, Jessica Watkin, Richard A Gibbs, John W Belmont, Donna Muzny, Lynne Nazareth, Erica Sodergren, George M Weinstock, David A Wheeler, Imtaz Yakub, Stacey B Gabriel, Robert C Onofrio, Daniel J Richter, Liuda Ziaugra, Bruce W Birren, Mark J Daly, David Altshuler, Richard K Wilson, Lucinda L Fulton, Jane Rogers, John Burton, Nigel P Carter, Christopher M Clee, Mark Griffiths, Matthew C Jones, Kirsten McLay, Robert W Plumb, Mark T Ross, Sarah K Sims, David L Willey, Zhu Chen, Hua Han, Le Kang, Martin Godbout, John C Wallenburg, Paul L'Archevêque, Guy Bellemare, Koji Saeki, Hongguang Wang, Daochang An, Hongbo Fu, Qing Li, Zhen Wang, Renwu Wang, Arthur L Holden, Lisa D Brooks, Jean E McEwen, Mark S Guyer, Vivian Ota Wang, Jane L Peterson, Michael Shi, Jack Spiegel, Lawrence M Sung, Lynn F Zacharia, Francis S Collins, Karen Kennedy, Ruth Jamieson, and John Stewart. A second generation human haplotype map of over 3.1 million snps. *Nature*, 449(7164):851–861, Oct 2007.

[CHL+07] D. Cook, H. Hofmann, E.K. Lee, H. Yang, B. Nikolau, and E. Wurtele. Exploring Gene Expression Data, Using Plots. *Journal of Data Science*, 5:151 – 182, 2007.

[Chu06a] M. Chung. Mandy Chung's Blog - From Monitoring to Diagnosing Memory Problem in Mustang. `http://weblogs.java.net/blog/mandychung/archive/2006/02/from_monitoring_1.html`, (01.07.2007), Feb. 2006.

[Chu06b] M. Chung. Monitoring and Managing Java SE 6 Platform Applications. `http://java.sun.com/developer/technicalArticles/J2SE/monitoring/`,(01.07.2007), Aug. 2006.

[CK97] Y.K. Cheung and J.H. Klotz. The Mann Whitney Wilcoxon Distribution using Linked Lists. *Statistica Sinica*, 7:805 – 813, 1997.

[Cle79] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–839, 1979.

[Cle81] William S. Cleveland. LOWESS: A Program for Smoothing Scatterplots by Robust Locally Weighted Regression. *The American Statistician*, 35(1):54, Feb. 1981.

[Cob06] K. Cobb. Microarrays: The Search for Meaning In a Vast Sea of Data. *Biomed Comput Rev*, 2(4):16 – 23, 2006.

[Coc52] W. G. Cochran. The χ^2 Test of Goodness of Fit. *The Annals of Mathematical Statistics*, 23(3):315 – 345, Sep. 1952.

[Coh94] J. Cohen. The Earth Is Round (p ¡ .05). *American Psychologist*, 49(12):997 – 1003, Dec. 1994.

[Cru08] CruiseControl Team. CC – CruiseControl. `http://cruisecontrol.sourceforge.net/`, (21.04.2008), 2008.

[CVS08] CVS Team. CVS - Concurrent Versions System. `http://www.nongnu.org/cvs/`, (21.04.2008), 2008.

[Dar98] J.D. Darcy. Borneo 1.0.2 - Adding IEEE 754 floating point support to Java. `http://sonic.net/~jddarcy/Borneo/borneo.pdf`, May 1998. (08.06.07).

[Dau07a] J.-M. Dautelle. javolution - The Java Solution for Real-Time and Embedded Systems, Version 5.1.0. `http://javolution.org/`, July 2007. (09.07.2007).

[Dau07b] J.-M. Dautelle. JScience - Java Tools and Libraries for the Advancement of Sciences, Version 4.1.2. `http://www.jscience.org/`, July 2007. (09.07.2007).

[Deu06] Katrin Deubel. Funktionelle Analyse von Genexpressionsdaten und Visualisierung beteiligter metabolischer Netze. Diplomarbeit, Universität Tübingen, 2006.

[DGB+03] Michael Dondrup, Alexander Goesmann, Daniela Bartels, Jörn Kalinowski, Lutz Krause, Burkhard Linke, Oliver Rupp, Alexander Sczyrba, Alfred Pühler, and Folker Meyer. Emma: a platform for consistent storage and efficient analysis of microarray data. *J Biotechnol*, 106(2-3):135–146, Dec 2003.

[DGH03] H. Doleisch, M. Gasser, and H. Hauser. Interactive Feature Specification for Focus+Context Visualization of Complex Simulation Data. In *Proc. of EG/IEEE VGTC Symposium on Visualization*, pages 239–248, 2003.

[DGN03] J. Dietzsch, N. Gehlenborg, and K. Nieselt. Poster: Mayday. In *German Conference on Bioinformatics (GCB)*, München, 2003.

[DGN06] Janko Dietzsch, Nils Gehlenborg, and Kay Nieselt. Mayday-a microarray data analysis workbench. *Bioinformatics*, 22(8):1010–1012, 2006.

[DGS+08] J. Dietzsch, N. Gehlenborg, S. Symons, M. Zschunke, F. Battke, M. Riester, and K. Nieselt. Mayday - Microarray Data Analysis. `http://www.zbit.uni-tuebingen.de/pas/mayday/index.html`, 2008. (05.02.2007).

[DH97] A.C. Davison and D.V. Hinkley. *Bootstrap Methods and their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.

[DHHR02] B.P. Durbin, J.S. Hardin, D.M. Hawkins, and D.M. Rocke. A variance-stabilizing function for gene-expression microarray data. *Bioinformatics*, 18:105 – 110, 2002.

[DHNB06] J. Dietzsch, J. Heinrich, K. Nieselt, and D. Bartz. Poster: Extended parallel coordinates for bioinformatical applications. In *German Conference on Bioinformatics (GCB)*, Tübingen, 2006.

[DHNB08] J. Dietzsch, J. Heinrich, K. Nieselt, and D. Bartz. Visual Analysis of Microarray Data from Bioinformatics Applications. WSI-Report, WSI-2008-1 ISSN 0946-3852, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Jan. 2008.

[Dig07] Digital Mars. D Programming Language. `http://www.digitalmars.com/d/index.html`,(25.06.2007), June 2007.

[DKF07] DKFZ Heidelberg. Illumina Technology. `http://www.dkfz.de/gpcf/beadchip_technology.html`, (19.05.2008), 2007.

[Don07] M. Dondrup. *EMMA2 – A MAGE-Compliant System for the Analysis of Microarray Data in Integrated Functional Genomics*. Dissertation, Universität Bielefeld, Center for Biotechnology (CeBiTec), Jan. 2007.

[Dra03] Sorin Draghici. *Data Analysis Tools for DNA Microarrays*. Chapman & Hall (Taylor & Francis Ltd.), 2003.

[DrP08] DrProject Team. dp – DrProject. `https://www.drproject.org/`, (21.04.2008), 2008.

[Dub07] M. Dublin. Not Just for Kids Anymore. *Genome Technology (online)*, 2007. `http://www.genome-technology.com/issues/2_7/webreprints/141908-1.html`, (23.10.2007).

[DW75] J.M. Davenport and J.T. Webster. The Behrens-Fisher problem, an old solution revisited. *Metrika*, 22(1):47 – 54, Dec. 1975.

[DYCS02] S. Dudoit, Y.H. Yang, M. J. Callow, and T.P. Speed. Statistical methods for identifying genes with differential expression in replicated cDNA microarray experiments. *Statistica sinica*, 12:111 – 139, 2002.

[DZN04] J. Dietzsch, M. Zschunke, and K. Nieselt. R inside - let r drive your bioinformatic application. In *useR! 2004*, Wien, 2004.

[Ecl08a] Eclipse Foundation. Eclipse - an open development platform. `http://www.eclipse.org/`, (21.04.2008), 2008.

[Ecl08b] Eclipse Foundation. Equinox. `http://www.eclipse.org/equinox/`, (21.04.2008), 2008.

[Ecl08c] Eclipse Foundation. SWT: The Standard Widget Toolkit. `http://www.eclipse.org/swt/`, (05.02.2008), 2008.

[Efr79a] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1 – 26, Jan 1979.

[Efr79b] B. Efron. Computers and the Theory of Statistics: Thinking the Unthinkable. *SIAM Review*, 21(4):460 – 480, Oct. 1979.

[Efr87] B. Efron. Better Bootstrap Confidence Intervals. *Journal of the American Statistical Association*, 82(397):171 – 185, Mar. 1987.

[Efr04] B. Efron. Large-Scale Simultaneous Hypothesis Testing: The Choice of a Null Hypothesis. *Journal of the American Statistical Association*, 99(465):96 – 104, March 2004.

[EG83] B. Efron and G. Gong. A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation. *The American Statistician*, 37(1):36 – 48, Feb. 1983.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *2nd int. Conf. on Knowledge Discovery and Data Mining (KDD 1996), Protland, Oregon*. AAAI Press, 1996.

[EPC07] EPCC - Concurrency and Applications (Benchmark) Group. The Java Grande Forum Benchmark Suite. `http://www2.epcc.ed.ac.uk/computing/research_activities/java_grande/index_1.html`, 2007. (04.06.2007).

[ES04] K. Eilbrecht and G. Starke. *Patterns kompakt - Entwurfsmuster fr effektive Software-Entwicklung*. Number 3827414431. Spektrum Akademischer Verlag Heidelberg, Berlin, 2004.

[ESBB98] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A*, 95(25):14863–14868, Dec 1998.

[ET86] B. Efron and R. Tibshirani. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science*, 1(1):54 – 75, Feb. 1986.

[FCI05] E. Fanea, S. Carpendale, and T. Isenberg. An Interactive 3D Integration of Parallel Coordinates and Star Glyphs. In *Proc. of IEEE Symposium on Information Visualization*, pages 149–156, 2005.

[Fel03] N. Feltovich. Nonparametric Tests of Differences in Medians: Comparison of the Wilcoxon-Mann-Whitney and Robust Rank-Order Tests. *Experimental Economics*, 6(3):273– 297, Nov. 2003.

[Fel05] N. Feltovich. Critical values for the robust rank-order test. *Communications in Statistics - Simulation and Computation*, 34(3):525 – 547, Jul. 2005.

[FG03] R.F. Fowler and C. Greenough. Software tools for hec applications. Technical report, Daresbury Laboratory, Central Laboratory of the Research Councils (CLRC), June 2003. `http://www.ukhec.ac.uk/publications/reports/softtools.pdf`.

[FH56] R.A. Fisher and M.J.R. Healy. New Tables of Behrens' Test of Significance. *Journal of the Royal Statistical Society, Series B (Methodological)*, 18(2):212 – 216, 1956.

[Fis22] R. A. Fisher. On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P. *Journal of the Royal Statistical Society*, 85(1):87 – 94, Jan. 1922.

[Fis39] R.A. Fisher. The Comparison of Samples With Possibly Unequal Variances. *Annals of Eugenics*, 9:174 – 180, 1939.

[FK99] I. Foster and C. Kesselman, editors. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[FKT01] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[Fow05] M. Fowler. *Refactoring oder: wie Sie das Design vorhandener Software verbessern*. Number 9783827322784. Addison-Wesley Verlag, 2005.

[FP81] M.A. Fligner and G.E. Policello. Robust Rank Procedures for the Behrens-Fisher Problem. *Journal of the American Statistical Association*, 76(373):162 – 168, Mar. 1981.

[Fre07a] Free Software Foundation. GCC, the GNU Compiler Collection. `http://gcc.gnu.org/`, (06.07.2007), 2007.

[Fre07b] Free Software Foundation. GSL - GNU Scientific Library. `http://www.gnu.org/software/gsl/`,(25.06.2007), Feb. 2007.

[Fri99] Michael Friendly. Re-visions of minard. *Statistical Computing and Graphics Newsletter*, 11(1):1, 13–19, 1999.

[Fri07a] E. D. Friedman. GNU Trove: High performance collections for Java (GNU Trove 2.0). `http://trove4j.sourceforge.net/index.shtml`, June 2007. (02.07.2007).

[Fri07b] Michael Friendly. Gallery of data visualization. `http://www.math.yorku.ca/SCS/Gallery/`, 2007. (20.04.2007).

[FRP+91] S. P. Fodor, J. L. Read, M. C. Pirrung, L. Stryer, A. T. Lu, and D. Solas. Light-directed, spatially addressable parallel chemical synthesis. *Science*, 251(4995):767–773, Feb 1991.

[FW06] John Fox and Sanford Weisberg. UseR! for teaching. In *useR! 2006*, Wien, 2006.

[GCB+04] Robert C Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. H. Yang, and Jianhua Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.

[GCH+05] Robert Gentleman, Vincent J. Carey, Wolfgang Huber, Rafael A. Irizarry, and Sandrine Dudoit. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Number 0-387-25146-4. Springer Science+Business Media Inc., 2005.

[GDN05] Nils Gehlenborg, Janko Dietzsch, and Kay Nieselt. A framework for visualization of microarray data and integrated meta information. *Information Visualization*, 4(3):164–175(12), 2005.

[GDS03] Y. Ge, S. Dudoit, and T.P. Speed. Resampling-based multiple testing for microarray data analysis. *Sociedad Española de Estadistica e Invetigación Operativa Test*, 12:1 – 77, 2003.

[Geh03] Nils Gehlenborg. Mayday - Microarray Data Analysis. Studienarbeit, Universität Tübingen, 2003.

[Gen03] James E. Gentle. *Random Number Generation and Monte Carlo Methods*. Number 978-0387001784 in Statistics and Computing. Springer-Verlag New York, 2003.

[GEP08] GEPAS Team. GEPAS - Gene Expression Pattern Analysis Suite. `http://gepas.bioinfo.cipf.es/`, (29.04.2008), 2008.

[GGHG96] A. G. Greenwald, R. Gonzalez, R. J. Harris, and D. Guthrie. Effect sizes and p values: what should be reported and what should be replicated? *Psychophysiology*, 33(2):175–183, Mar 1996.

[GHJV96] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Number 3893199500. Addison-Wesley Longman Verlag GmbH, 1. edition, 1996.

[GHM04] James E Gentle, Wolfgang Härdle, and Yuichi Mori, editors. *Handbook of Computational Statistics - Concepts and Methods*. Springer Science+Business Media Inc., 2004.

[GKG+04] Kevin L Gunderson, Semyon Kruglyak, Michael S Graige, Francisco Garcia, Bahram G Kermani, Chanfeng Zhao, Diping Che, Todd Dickinson, Eliza Wickham, Jim Bierle, Dennis Doucet, Monika Milewski, Robert Yang, Chris Siegmund, Juergen Haas, Lixin Zhou, Arnold Oliphant, Jian-Bing Fan, Steven Barnard, and Mark S Chee. Decoding randomly ordered dna arrays. *Genome Res*, 14(5):870–877, May 2004.

[GL06] D. Gamerman and H. F. Lopes. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Texts in Statistical Science. Chapman & Hall/CRC, 2. edition, 2006.

[Goo05] P. Good. *Permutation, Parametric, and Bootstrap Tests of Hypotheses*. Springer Science+Business Media Inc., third edition, 2005.

[GPG07] GPGPU.org. General-Purpose Computation Using Graphics Hardware. `http://www.gpgpu.org`, (22.10.2007), 2007.

[GPH+03] G.L. Gadbury, G.P. Page, M. Heo, J.D. Mountz, and D.B. Allison. Randomization tests for small samples: an application for genetic expression data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 52(3):365 – 376, 2003.

[Gra06] M. Grammling. OSGi Schichtenmodell. `http://de.wikipedia.org/wiki/Bild:Osgi_layer.png`, (21.04.2008), Feb 2006.

[GRW+00] D. Gresh, B. Rogowitz, R. Winslow, D. Scollan, and C. Yung. WEAVE: A System for Visually Linking 3-D and Statistical Visualization, Applied to Cardiac Simulation and Measurement Data. In *Proc. of IEEE Visualization*, pages 489–492, 2000.

[GRY05] Michael G Griffin, Patricia A Resick, and Rachel Yehuda. Enhanced cortisol suppression following dexamethasone administration in domestic violence survivors. *Am J Psychiatry*, 162(6):1192–1199, Jun 2005.

[GSE08] GSEA Team. GSEA - Gene Set Enrichment Analysis. `http://www.broad.mit.edu/gsea/index.jsp`, (16.04.2008), 2008.

[GWD+99] N. P. Gerry, N. E. Witowski, J. Day, R. P. Hammer, G. Barany, and F. Barany. Universal dna microarray method for multiplex detection of low abundance point mutations. *J Mol Biol*, 292(2):251–262, Sep 1999.

[Hal06] M. Hale. JSci - A science API for Java, Version 0.943. http://jsci.sourceforge.net/, May 2006. (15.07.2007).

[HAM00] Y.F. Hu, R.J. Allan, and K.C.F. Maguire. Comparing the performance of java with fortran and c for numerical computing. Technical report, Daresbury Laboratory, Central Laboratory of the Research Councils (CLRC), 2000. http://www.ukhec.ac.uk/publications/reports/bench.pdf, (05.06.2007).

[HAQS07] Daniel H Huson, Alexander F Auch, Ji Qi, and Stephan C Schuster. Megan analysis of metagenomic data. *Genome Res*, 17(3):377–386, Mar 2007.

[Hay00] A.F. Hayes. Randomization tests and the equality of variance assumption when comparing group means. *Animal Behaviour*, 59:653 – 656, 2000.

[HBMS03] H. Hochheiser, E.H. Baehrecke, S.M. Mount, and B. Shneiderman. Dynamic querying for pattern identification in microarray and genomic data. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 3, pages III–453–6vol.3, 6-9 July 2003.

[Hei07] Julian Heinrich. Extending Parallel Coordinates for Bioinformatical Applications. Studienarbeit, Universtät Tübingen, 2007.

[Hen07] T. Hentrich. ChIP-on-Chip. http://en.wikipedia.org/wiki/Image:ChIP-on-chip_wet-lab.png, (16.04.2008), 2007.

[HH64] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Methuen's Monographs on Applied Statistics. John Wiley & Sons Inc, 1964.

[HJS+05] J. Hong, D. Jeong, C. Shaw, W. Ribarsky, M. Borodovsky, and C. Song. GVis: A Scalable Visualization Framework for Genomic Data. In *Proc. of EG/IEEE VGTC Symposium on Visualization*, pages 191–198, 2005.

[HKY99] L. J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome Res*, 9(11):1106–1115, Nov 1999.

[HL04] S. E. Hanlon and J. D. Lieb. Progress and challenges in profiling the dynamics of chromatin and transcription factor binding with dna microarrays. *Curr Opin Genet Dev*, 14(6):697–705, Dec 2004.

[HMJ+06] Nicole C Hauser, Rafael Martinez, Anette Jacob, Steffen Rupp, Jörg D Hoheisel, and Stefan Matysiak. Utilising the left-helical conformation of l-dna for analysing different marker types on a single universal microarray platform. *Nucleic Acids Res*, 34(18):5101–5111, 2006.

[HMM+01] G. K. Hu, S. J. Madore, B. Moldover, T. Jatkoe, D. Balaban, J. Thomas, and Y. Wang. Predicting splice variant from dna chip expression data. *Genome Res*, 11(7):1237–1245, Jul 2001.

[HMS01] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press Cambridge, MA, USA, 2001.

[Hoc88] Y. Hochberg. A sharper bonferroni prcedure for multiple tests of significance. *Biometrika*, 75:800–802, 1988.

[Hoh06] Jörg D Hoheisel. Microarray technology: beyond transcript profiling and genotype analysis. *Nat Rev Genet*, 7(3):200–210, Mar 2006.

[Hol79] S. Holm. A simple sequentially rejective multiple test procedure. *Scand. J. Statist.*, 6:65–70, 1979.

[HRR+07] Daniel H Huson, Daniel C Richter, Christian Rausch, Tobias Dezulian, Markus Franz, and Regula Rupp. Dendroscope: An interactive viewer for large phylogenetic trees. *BMC Bioinformatics*, 8:460, 2007.

[HSPWR04] S. Havre, M. Singhal, D. Payne, and B. Webb-Robertson. PQuad: Visualization of Predicted Peptides and Proteins. In *Proc. of IEEE Visualization*, pages 473–480, 2004.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. Springer Science+Business Media Inc., 2001.

[HTLS01] L. Hunter, R. C. Taylor, S. M. Leach, and R. Simon. Gest: a gene expression search tool based on a novel bayesian similarity metric. *Bioinformatics*, 17 Suppl 1:S115–S122, 2001.

[IEE85] IEEE. IEEE Standard 754-1985 for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985, IEC-60559:1989 - International Version). *Reprinted in SIGPLAN Notices*, 22(2):9 – 25, 1985. `http://754r.ucbtest.org/standards/754.pdf` (07.06.2007).

[Ill08a] Illumina Inc. HumanWG-6 & HumanRef-8 v3.0 Expression BeadChips Data Sheet. `http://www.illumina.com/downloads/GX_Humanv3.0_DataSheet.pdf`,(20.05.2008), 2008.

[Ill08b] Illumina Inc. Products & Service. http://www.illumina.com/pages.ilmn?ID=208, (10.01.2008), 2008.

[Inf07] InfoVis:Wiki Team. InfoVis:Wiki - the Information Visualization community platform. http://www.infovis-wiki.net/index.php?title=Main_Page,(11.12.2007), 2007.

[Ins85] A. Inselberg. The Plane with Parallel Coordinates. *The Visual Computer*, 1:69–92, 1985.

[Int07] Intel Corporation. Intel Research Advances 'Era Of Tera' – World's First Programmable Processor to Deliver Teraflops Performance with Remarkable Energy Efficiency. http://www.intel.com/pressroom/archive/releases/20070204comp.htm, (22.10.2007), 2007.

[Jam59] G.S. James. The Behrens-Fisher Distribution and Weighted Means. *Journal of the Royal Statistical Society, Series B (Methodological)*, 21(1):73 – 90, 1959.

[Jav98] Java Grande Forum. The Java Grande Forum Charter. http://www.javagrande.org/Charter.html, 1998. (04.06.2007).

[Jav07a] Java Grande Forum. Java Numerics. http://math.nist.gov/javanumerics/, 2007. (09.07.2007).

[Jav07b] Java Grande Forum. The Java Grande Forum. http://www.javagrande.org, 2007. (06.06.2007).

[J.C08] J.C. Venter Institute. TM4 - Microarray Software Suite. http://www.tm4.org/, 2008. (29.04.2008).

[JLJC05] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing Structure within Clustered Parallel Coordinates Displays. In *Proc. of IEEE Symposium on Information Visualization*, pages 125–132, 2005.

[Jon06] M. Jones. Information flow in biological systems. http://en.wikipedia.org/wiki/Image:CDMB2.png, (07.05.2008), Dec 2006.

[JUn08] JUnit Team. JUnit.org - Resources for Test Driven Development. http://www.junit.org/, (23.04.2008), 2008.

[Kah00] W. Kahan. Marketing versus mathematics and other ruminations on the design of floating-point arithmetic. In *receiving the IEEE's Emanuel R. Piore Award*. IEEE, Aug. 2000. http://www.cs.berkeley.edu/∼wkahan/MktgMath.pdf.

[Kas01] E. Kasuya. Mann-Whitney U test when variances are unequal. *Animal Behaviour*, 61:1247 – 1249, 2001.

[KBC+04] Kenneth Kuhn, Shawn C Baker, Eugene Chudin, Minh-Ha Lieu, Steffen Oeser, Holly Bennett, Philippe Rigault, David Barker, Timothy K McDaniel, and Mark S Chee. A novel, high-performance random array platform for quantitative gene expression profiling. *Genome Res*, 14(11):2347–2356, Nov 2004.

[KD98] W. Kahan and J.D. Darcy. How Java's Floating-Point Hurts Everyone Everywhere. In *ACM 1998 Workschop on Java for High-Performance Network Computing*, Stanford, 1998. http://www.cs.berkeley.edu/ wkahan/JAVAhurt.pdf (updated July 2004) (07.06.2007).

[KKC+04] M. Kapushesky, P. Kemmeren, A.C. Culhane, S. Durinck, J. Ihmels, C. Körner, M. Kull, A. Torrente, U. Sarkans, J. Vilo, and A. Brazma. Expression profiler: next generation–an online platform for analysis of microarray data. *Nucleic Acids Res*, 32(Web Server issue):W465–W470, Jul 2004.

[KL51] S. Kullback and R. A. Leibler. On Information and Sufficiency. *Ann. Math. Statist.*, 22(1):79 – 86, 1951.

[Klo66] J.H. Klotz. The Wilcoxon, Ties, and the Computer. *Journal of the American Statistical Association*, 61(315):772 – 787, Sep. 1966.

[KMS+04] Philipp Khaitovich, Bjoern Muetzel, Xinwei She, Michael Lachmann, Ines Hellmann, Janko Dietzsch, Stephan Steigele, Hong-Hai Do, Gunter Weiss, Wolfgang Enard, Florian Heissig, Thomas Arendt, Kay Nieselt-Struwe, Evan E Eichler, and Svante Pääbo. Regional patterns of gene expression in human and chimpanzee brains. *Genome Res*, 14(8):1462–1473, Aug 2004.

[Kna07] B. Knapp. Molekulare Signaturen primärer intestinaler Tumore. Diplomarbeit, Universität Tübingen, August 2007.

[Lan99a] E. S. Lander. Array of hope. *Nat Genet*, 21(1 Suppl):3–4, Jan 1999.

[Lan99b] Kenneth Lange. *Numerical Analysis for Statisticians*. Statistics and Computing. Springer-Verlag New York, 2. edition, 1999.

[Lap07] Lapack. LAPACK – Linear Algebra PACKage. `http://www.netlib.org/lapack/`,(25.06.2007), June 2007.

[Lau03] Hang T. Lau. *A Numerical Library in Java for Scientists and Engineers*. Number 978-1584884309. Chapman & Hall/CRC, Aug. 2003. Book and CD.

[LD98] J. Ludbrook and H. Dudley. Why Permutation Tests Are Superior to t and F Tests in Biomedical Research. *The American Statistician*, 52(2):127 – 132, May 1998.

[LDB+96] D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat Biotechnol*, 14(13):1675–1680, Dec 1996.

[L'E07] P. L'Ecuyer. SSJ - Stochastic Simulation in Java, Version 1.2.2. `http://www.iro.umontreal.ca/ simardr/ssj/index.html`, Apr. 2007. (16.07.2007).

[Lee04] P. M. Lee. *Bayesian Statistics. An Introduction.* Hodder Arnold, 3. edition, 2004.

[Liu07] X. Shirley Liu. Getting started in tiling microarray analysis. *PLoS Comput Biol*, 3(10):1842–1844, Oct 2007.

[LLBB05] Lars Linsen, Julia Löcherbach, Matthias Berth, and Jörg Bernhardt. Differential Protein Expression Analysis via Liquid-Chromatography/Mass-Spectrometry Data Visualization. In *Proc. of IEEE Visualization*, pages 447–454, 2005.

[Lon08] J. Long. The Science Creative Quarterly - Image (bank). `http://www.scq.ubc.ca/image-bank/`, (08.05.2008), 2008.

[Mak08] Makewave AB. Knopflerfish - Open Source OSGi. `http://www.knopflerfish.org/index.html`, (21.04.2008), 2008.

[Man08] Mangapoco. A Fragment of cDNA Microarray. `http://en.wikipedia.org/wiki/Image:Cdnaarray.jpg`, (18.05.2008), 2008.

[Mat08] MathWorks Inc. MATLAB R2007b. `http://www.mathworks.de`, (10.08.2008), 2008.

[Mer07] B. Merkle. C++ – C++0x: Ausblick auf den neuen C++-Standard. *iX – Magazin fr professionelle Informationstechnik*, (6/2007):60 – 66, Juni 2007.

[MGE07] MGED Board. Minimum Information About a Microarray Experiment - MIAME 2.0 (a proposal for discussion). `http://www.mged.org/Workgroups/MIAME/miame_2.0.html`, (16.01.2008), 2007.

[MGE08] MGED. Mage-ml, 2008. (18.01.2008).

[MGK+86] J. W. Mason, E. L. Giller, T. R. Kosten, R. B. Ostroff, and L. Podd. Urinary free-cortisol levels in posttraumatic stress disorder patients. *J Nerv Ment Dis*, 174(3):145–149, Mar 1986.

[Mil74] R.G. Miller. The Jackknife - A Review. *Biometrika*, 61(1):1 – 15, Apr. 1974.

[MLB+98] M. Maes, A. Lin, S. Bonaccorso, F. van Hunsel, A. Van Gastel, L. Delmeire, M. Biondi, E. Bosmans, G. Kenis, and S. Scharpé. Increased 24-hour urinary cortisol excretion in patients with posttraumatic stress disorder and patients with major depression, but not in patients with fibromyalgia. *Acta Psychiatr Scand*, 98(4):328–335, Oct 1998.

[Mor00] J. E. Moreira. JSR 83: Multiarray package. `http://jcp.org/en/jsr/detail?id=83`,(25.06.2007), Sep. 2000.

[MP80] C.R. Mehta and N.R. Patel. A network algorithm for the exact treatment of the 2×k contingency table. *Communications in Statistics - Simulation and Computation*, 9(6):649 – 664, 1980.

[MPT84] C.R. Mehta, N.R. Patel, and A.A. Tsiatis. Exact Significance Testing to Establish Treatment Equivalence with Ordered Categorical Data. *Biometrics*, 40(3):819 – 825, Sep. 1984.

[MPW88] C.R. Mehta, N.R. Patel, and L.J. Wei. Constructing exact significance tests with restricted randomization rules. *Biometrika*, 75(2):295 – 302, 1988.

[MRR+53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087 – 1092, June 1953.

[MS70] J.S. Mehta and R. Srinivasan. On the Behrens-Fisher Problem. *Biometrika*, 57(3):649 – 655, Dec. 1970.

[MTHC+06] David Montaner, Joaquín Táraga, Jaime Huerta-Cepas, Jordi Burguet, Juan M Vaquerizas, Lucía Conde, Pablo Minguez, Javier Vera, Sach Mukherjee, Joan Valls, Miguel A G Pujana, Eva Alloza, Javier Herrero, Fátima Al-Shahrour, and Joaquín Dopazo. Next station in microarray data analysis: Gepas. *Nucleic Acids Res*, 34(Web Server issue):W486–W491, Jul 2006.

[MU49] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341, Sep. 1949.

[Mur93] James D. Murray. *Mathematical Biology*. Springer-Verlag Berlin, second edition, 1993.

[MW47] H.B. Mann and D.R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60, Mar. 1947.

[MyS07] MySQL AB. MySQL. `http://www.mysql.com`, 2007. (20.04.2007).

[Nat07] National Institute of Standards and Technology. NIST. `http://www.nist.gov/`, 2007. (04.06.2007).

[NCB08] NCBI. GEO – Gene Expression Omnibus. `http://www.ncbi.nlm.nih.gov/projects/geo/`, (14.06.2008), 2008.

[Neu02] M. Neuhäuser. The baumgartner-weiss-schindler test in the presence of ties. *Biometrics*, 58(1):250–251, Mar. 2002.

[Nim08a] NimbleGen Systems Inc. Eukaryotic Gene Expression Arrays. `http://www.nimblegen.com/products/exp/#eukaryotic`, (18.05.2008), 2008.

[Nim08b] NimbleGen Systems Inc. NimbleGen - Welcome to High-Definition Genomics. `http://www.nimblegen.com`, (25.05.2008), 2008.

[NL04] M. Neuhäuser and F.C. Lam. Nonparametric Approaches to Detecting Differentially Expressed Genes in Replicated Microarray Experiments. In *2nd Asia-Pacific Bioinformatics Conference (APBC 2004)*, volume 29. Conferences in Research and Practice in Information Technology, 2004.

[NNC94] D. C. Nieman and S. L. Nehlsen-Cannarella. The immune response to exercise. *Semin Hematol*, 31(2):166–179, Apr 1994.

[Now05] J. Nowak. *Fortgeschrittene Programmierung mit Java 5*. Number 978-3898643061. dpunkt.Verlag Heidelberg, 2005.

[NS04] M. Neuhäuser and R. Senske. The baumgartner-weiss-schindler test for the detection of differentially expressed genes in replicated microarray experiments. *Bioinformatics*, 20(18):3553–3564, Dec. 2004.

[NSvH01] Kay Nieselt-Struwe and Arndt von Haeseler. Quartet-Mapping, a Generalization of the Likelihood-Mapping Procedure. *Mol Biol Evol*, 18(7):1204–1219, 2001.

[NVA07] NVAC - National Visualization and Analytics Center. FAQ, 2007. http://nvac.pnl.gov/qa.stm (13.04.2007).

[NVI07] NVIDIA Corporation. NVIDIA CUDA Homepage. `http://developer.nvidia.com/object/cuda.html`, (22.10.2007), 2007.

[O'H05] K. O'Hair. Kelly o'hair's blog - heap dump snapshots. `http://weblogs.java.net/blog/kellyohair/archive/2005/09/heap_dump_snaps.html`,(01.07.2007), Sep. 2005.

[Ora08] Oracle Corporation. Oracle Berkeley DB Product Family. `http://www.oracle.com/database/berkeley-db/index.html`, (21.04.2008), 2008.

[Org07] Organization for the Advancement of Structured Information Standards (OASIS). OASIS SOA Reference Model TC. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm`, (18.01.2008), 2007.

[OSG07] OSGi Alliance. OSGi Alliance Specifications – OSGi Service Platform Release 4. `http://www.osgi.org/Specifications/HomePage`, (21.04.2008), May 2007.

[OSG08] OSGi Alliance. OSGi Alliance - Home Page. `http://www.osgi.org/Main/HomePage`, (21.04.2008), 2008.

[OW07] A. Oram and G. Wilson, editors. *Beautiful Code*. Number 978-0596510046. O'Reilly Media, 1. edition, 2007.

[oW08] University of Waikato. Weka 3: Data Mining Software in Java. `http://www.cs.waikato.ac.nz/ml/weka`,(15.04.2008), 2008.

[Pan02] Wei Pan. A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. *Bioinformatics*, 18(4):546–554, Apr 2002.

[Pat65] V.H. Patil. Approximation to the Behrens-Fisher Distributions. *Biometrika*, 52(1/2):267 – 271, Jun. 1965.

[PBM05] K. Pradhan, D. Bartz, and K. Mueller. SignatureSpace: A Multidimensional, Exploratory Approach for the Analysis of Volume Data. Technical Report WSI-2005-11, ISSN 0946-3852, Dept. of Computer Science (WSI), University of Tübingen, 2005.

[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[Pea11] K. Pearson. On the Probability that Two Independent Distributions of Frequency are Really Samples from the Same Population. *Biometrika*, 8(1/2):250 – 254, Jul. 1911.

[php07] php.net. PHP. `http://php.net/`, 2007. (20.04.2007).

[Pit37] E.J.G. Pitman. Significance Tests Which May be Applied to Samples From any Populations. *Supplement to the Journal of the Royal Statistical Society*, 4(1):119 – 130, 1937.

[PM00] R. Pozo and B. Miller. SciMark 2.0. `http://math.nist.gov/scimark2/`,(07.06.2007), 2000.

[PO90] R. K. Pitman and S. P. Orr. Twenty-four hour urinary cortisol and catecholamine excretion in combat-related posttraumatic stress disorder. *Biol Psychiatry*, 27(2):245–247, Jan 1990.

[Pos07] PostgreSQL. PostgreSQL - SQL-compliant, open source object-relational database management system. `http://www.postgresql.org`, 2007. (20.04.2007).

[Pre08] PreAnalytiX - a QIAGENE/BD Company. PAXgene Blood RNA System. `http://www.preanalytix.com/RNA.asp`, (31.05.2008), 2008.

[PS01] M. L. Pall and J. D. Satterlee. Elevated nitric oxide/peroxynitrite mechanism for the common etiology of multiple chemical sensitivity, chronic fatigue syndrome, and posttraumatic stress disorder. *Ann N Y Acad Sci*, 933:323–329, Mar 2001.

[PTV92] W. H. Press, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[PvdWFvW04] T. Peeters, H. van de Wetering, M. Fiers, and J. van Wijk. Case Study: Visualization of Annotated DNA Sequences. In *Proc. of EG/IEEE VGTC Symposium on Visualization*, pages 109–114, 2004.

[R D07] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3-900051-07-0.

[RBN07] Chris Rorden, Leonardo Bonilha, and Thomas E Nichols. Rank-order versus mean based statistics for neuroimaging. *Neuroimage*, 35(4):1531–1537, May 2007.

[RDC+02] T. Rhyne, T. Dunning, G. Calapristi, C. North, and D. Gresh. Panel 4: Evolving Visual Metaphors and Dynamic Tools for Bioinformatics Visualization. In *Panel 4, IEEE Visualization*, pages 579–582, 2002.

[Ree79] Trygve Reenskaug. Models-Views-Controllers. Note, Xerox Palo Alto Research Laboratory (PARC), Dec. 1979.

[Rie05] Markus Riester. Expressionsphylogenien. Studienarbeit, Universität Tübingen, 2005.

[RIF+06] Richard Redon, Shumpei Ishikawa, Karen R Fitch, Lars Feuk, George H Perry, T. Daniel Andrews, Heike Fiegler, Michael H Shapero, Andrew R Carson, Wenwei Chen, Eun Kyung Cho, Stephanie Dallaire, Jennifer L Freeman, Juan R González, Mònica Gratacòs, Jing Huang, Dimitrios Kalaitzopoulos, Daisuke Komura, Jeffrey R MacDonald, Christian R Marshall, Rui Mei, Lyndal Montgomery, Kunihiro Nishimura, Kohji Okamura, Fan Shen, Martin J Somerville, Joelle Tchinda, Armand Valsesia, Cara Woodwark, Fengtang Yang, Junjun Zhang, Tatiana Zerjal, Jane Zhang, Lluis Armengol, Donald F Conrad, Xavier Estivill, Chris Tyler-Smith, Nigel P Carter, Hiroyuki Aburatani, Charles Lee, Keith W Jones, Stephen W Scherer, and Matthew E Hurles. Global variation in copy number in the human genome. *Nature*, 444(7118):444–454, Nov 2006.

[Rip04] B. D. Ripley. Data mining: Large databases and methods or ... finding needles in haystacks: Finding unusual patterns in large data sets. In *useR! 2004*, Wien, 2004.

[Rit01a] K. Ritley. Java as a Scientific Programming Language (Part 1). `http://www.developer.com/java/other/article.php/631151`, Jan. 2001. (08.06.2007).

[Rit01b] K. Ritley. Scientific Computing in Java (Part 2): Writing Scientific Programs in Java. `http://www.developer.com/java/other/article.php/631281`, Jan. 2001. (08.06.2007).

[Rou02] V. Roubtsov. Java Tip 130: Do you know your data size? - Don't pay the price for hidden class fields. http://www.javaworld.com/javaworld/javatips/jw-javatip130.html, Aug. 2002.

[Rux06] G.D. Ruxton. The unequal variance t-test is an underused alternative to Student's t-test and the Mann-Whitney U test. *Behavioral Ecology*, 17(4):688 –690, 2006.

[RWK+06] O. Rübel, G. Weber, S. Keränen, C. Fowlkes, C. Luengo Hendriks, L. Simirenko, N. Shah, M. Eisen, M. Biggin, H. Hagen, D. Sudar, J. Malik, D. Knowles, and B. Hamann. PointCloudXplore: Visual Analysis of 3D Gene Expression Data Using Physical Views and Parallel Coordinates. In *Proc. of EG/IEEE VGTC Symposium on Visualization*, pages 203–210, 2006.

[SAS08] SAS Institute Inc. SAS 9. `http://www.sas.com/offices/europe/germany/solutions/ei_.html`, (10.08.2008), 2008.

[Sat46] F. E. Satterthwaite. An Approximate Distribution of Estimates of Variance Components. *Biometrics Bulletin*, 2(6):110 – 114, Dec. 1946.

[SAW06] Ronald C Serlin, Suzanne Ameringer, and Sandra E Ward. Re: A note on "the significance of significance". *Res Nurs Health*, 29(2):166–7; author reply 168–9, Apr 2006.

[SB01] L.A. Smith and J.M. Bull. Java for high performance computing. Technical report, Edinburgh Parallel Computing Centre (EPCC), University of Edinburgh, 2001. `http://www.ukhec.ac.uk/publications/tw/hpcjava.pdf`, (05.06.2007).

[SBD+06] S. Symons, F. Battke, J. Dietzsch, M. Zschunke, and K. Nieselt. Poster: Automated processing and machine learning tools for mayday. In *German Conference on Bioinformatics (GCB)*, Tübingen, 2006.

[SC02] Kerby Shedden and Stephen Cooper. Analysis of cell-cycle gene expression in saccharomyces cerevisiae using microarrays and multiple synchronization methods. *Nucleic Acids Res*, 30(13):2920–2929, Jul 2002.

[SCC03] Jonathan L Sebat, Frederick S Colwell, and Ronald L Crawford. Metagenomic profiling: microarray analysis of an environmental genomic library. *Appl Environ Microbiol*, 69(8):4927–4934, Aug 2003.

[Sch70] H. Scheffe. Practical Solutions of the Behrens-Fisher Problem. *Journal of the American Statistical Association*, 65(332):1501 – 1508, Dec. 1970.

[Sch96] M.J. Schervish. P Values: What They Are and What They Are Not. *The American Statistician*, 50(3):203 – 206, Aug. 1996.

[Sch01] U. Schöning. *Algorithmik*. Number 3827410924. Spektrum Akademischer Verlag Heidelberg, Berlin, 2001.

[Sch06] Schutz. Two Affymetrix Chips. `http://en.wikipedia.org/wiki/Image:Affymetrix-microarray.jpg`, (18.05.2008), 2006.

[Sch07] Christian Schillinger. Robuste Merkmalsselektion aus Bipartitionen von Microarray-Expressionsdaten mit Anwendung auf aggregierte Tumordaten. Diplomarbeit, Universität Tübingen, 2007.

[SCPT+04] F. Sanchez-Cabo, A. Prokesch, G. G. Thallinger, R. Pieler, Z. Trajanoskie, P. D. Butcher, J. Hinds, L. E. A. Holmes, S. G. Campbell, M. P. Ashe, S. Hubbard, K.-H. Cho, and O. Wolkenhauer. Assesing the efficiency of dye-swap normalization to remove systematic bias from two-color microarray data. 2004. `http://www.sbi.uni-rostock.de/publications_journals.html` (20.04.2007).

[SD07a] K. Seymour and J. Dongarra. F2j. `http://icl.cs.utk.edu/f2j/index.html`,(25.06.2007), June 2007.

[SD07b] K. Seymour and J. Dongarra. JLapack Version 0.8. `http://www.netlib.org/java/f2j/`, Mar. 2007. (18.07.2007).

[SDN07] S. Symons, J. Dietzsch, and K. Nieselt. Poster: Flexible microarray data analysis using mayday. In *European BioPerspectives 2007*, Köln, 2007.

[SF98] J. M. Stonehouse and G. J. Forrester. Robustness of the t and u tests under combined assumption violations. *Journal of Applied Statistics*, 25:63 – 74, Feb. 1998.

[SFD+07] Barbara E Stranger, Matthew S Forrest, Mark Dunning, Catherine E Ingle, Claude Beazley, Natalie Thorne, Richard Redon, Christine P Bird, Anna de Grassi, Charles Lee, Chris Tyler-Smith, Nigel Carter, Stephen W Scherer, Simon Tavaré, Panagiotis Deloukas, Matthew E Hurles, and Emmanouil T Dermitzakis. Relative impact of nucleotide and copy number variation on gene expression phenotypes. *Science*, 315(5813):848–853, Feb 2007.

[SG04] D. Sorensen and D. Gianola. *Likelihood, Bayesian, and MCMC Methods in Quantitative Genetics*. Springer Science+Business Media Inc., second edition, 2004.

[SH06] L. Sachs and J. Heddrich. *Angewandte Statistik*. Springer-Verlag Berlin, 12. edition, 2006.

[Sid67] Z. Sidak. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62:626–633, 1967.

[SKG+07] Aravind Subramanian, Heidi Kuehn, Joshua Gould, Pablo Tamayo, and Jill P Mesirov. Gsea-p: a desktop application for gene set enrichment analysis. *Bioinformatics*, 23(23):3251–3253, Dec 2007.

[SL85] R.C. Serlin and D.K. Lapsley. The Good-Enough Principle. *American Psychologist*, 40(1):73 – 83, Jan. 1985.

[SL07] R. Simon and A. P. Lem. *BRB Array Tools 3.5 User's Manual.*, 2007. `http://linus.nci.nih.gov/BRB-ArrayTools.html` (20.04.2007).

[SMZ01] J. E. Stewart, H. Mangalam, and J. Zhou. Open source software meets gene expression. *Brief Bioinform*, 2(4):319–328, Dec 2001.

[SND05] P. Saraiya, C. North, and K. Duca. Visualizing Biological Pathways: Requirements Analysis, Systems Evaluation and Research Agenda. In *Proc. of IEEE Symposium on Information Visualization*, pages 191–205, 2005.

[SNLD06] Purvi Saraiya, Chris North, Vy Lam, and Karen A. Duca. An Insight-Based Longitudinal Study of Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1511–1522, 2006.

[Sou75] E. M. Southern. Detection of specific sequences among dna fragments separated by gel electrophoresis. *J Mol Biol*, 98(3):503–517, Nov 1975.

[Spe00] T.P. Speed. Hints and prejudices - always log spot intensities and ratios, 2000. `http://stat-www.berkeley.edu/users/terry/zarray/Html/log.html`, (22.08.2006).

[Spo08a] Spotfire Inc. DecisionSite for Functional Genomics - Rapidly Analyze and Isolate Significant Genes, 2008. `http://spotfire.tibco.com/products/decisionsite_functional_genomics.cfm` (22.01.2008).

[Spo08b] Spotfire Inc. Spotfire DecisionSite for Microarray Analysis (DSMA) – Visual Analytics and Powerful Statistics, 2008. `http://spotfire.tibco.com/products/decisionsite_microarray_analysis.cfm` (10.01.2008).

[SPS07] SPSS Inc. Neuerungen in SPSS 16.0: SPSS Base. `http://www.spss.com/de/spss16/whats_new_base.htm`, (22.10.2007), 2007.

[SS02] J Seo and B Shneiderman. Interactively exploring hierarchical clustering results. *IEEE Computer*, 35:80 – 86, 2002.

[SS03] Gordon K Smyth and Terry Speed. Normalization of cdna microarray data. *Methods*, 31(4):265–273, Dec 2003.

[SSD+07] S. Symons, C. Schillinger, J. Dietzsch, F. Battke, and K. Nieselt. Poster: Genemining in mayday, a feature selection framework for binary classification. In *German Conference on Bioinformatics (GCB)*, Potsdam, 2007.

[SSDB95] M. Schena, D. Shalon, R.W. Davis, and P.O. Brown. Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray. *Science*, 270(5235):467–470, 1995.

[SST+04] Gali Steinberg, Katie Stromsborg, Lynette Thomas, David Barker, and Chanfeng Zhao. Strategies for covalent attachment of dna to beads. *Biopolymers*, 73(5):597–605, Apr 2004.

[SSW+03] AI Saeed, V Sharov, J White, J Li, W Liang, N Bhagabati, J Braisted, M Klapa, T Currier, M Thiagarajan, A Sturn, M Snuffin, A Rezantsev, D Popov, A Ryltsov, E Kostukovich, I Borisovsky, Z Liu, A Vinsavich, V Trush, and J Quackenbush. TM4: a free, open-source system for microarray data management and analysis. *BioTechniques*, 34(2):374 – 378, 2003.

[SSZ+98] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9(12):3273–3297, Dec 1998.

[Sta07] M. Stal. D – die neue Programmiersprache mit C++-Wurzel. *iX – Magazin fr professionelle Informationstechnik*, (6/2007):68 – 72, Juni 2007.

[STM+05] Aravind Subramanian, Pablo Tamayo, Vamsi K Mootha, Sayan Mukherjee, Benjamin L Ebert, Michael A Gillette, Amanda Paulovich, Scott L Pomeroy, Todd R Golub, Eric S Lander, and Jill P Mesirov. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A*, 102(43):15545–15550, Oct 2005.

[Str00] Bjarne Stroustrup. *Die C++ Programmiersprache*. Number 0-201-70073-5. Addison-Wesley Verlag, 4. edition, 2000.

[Stu] G. W. Stuart. JAMPACK - A Java Package for Matrix Computations, Version 1.0.2. ftp://math.nist.gov/pub/Jampack/Jampack/AboutJampack.html. (16.07.2007).

[STVC+02] Lao Saal, Carl Troein, Johan Vallon-Christersson, Sofia Gruvberger, Ake Borg, and Carsten Peterson. Bioarray software environment (BASE): a platform for comprehensive management and analysis of microarray data. *Genome Biology*, 3(8):software0003.1–software0003.6, 2002.

[Sun96] Sun Microsystems. JAVASOFT SHIPS JAVA 1.0. http://www.sun.com/smi/Press/sunflash/1996-01/sunflash.960123.10561.xml, (30.05.2007), Jan. 1996.

[Sun98] Sun Microsystems. SUN DELIVERS NEXT VERSION OF THE JAVA PLATFORM – Java 2 Brand Unveiled. http://www.sun.com/smi/Press/sunflash/1998-12/sunflash.981208.9.xml, (03.06.2007), Dec. 1998.

[Sun04a] Sun Microsystems. Java Native Interface - JNI 5.0 Specification. http://java.sun.com/j2se/1.5.0/docs/guide/jni/index.html, (05.02.2008), 2004.

[Sun04b] Sun Microsystems. JCP 2: Process Document. http://jcp.org/en/procedures/jcp2,(25.06.2007), Mar. 2004.

[Sun04c] Sun Microsystems. New Features and Enhancements J2SE 5.0. http://java.sun.com/j2se/1.5.0/docs/relnotes/features.html, (02.07.2007), 2004.

[Sun04d] Sun Microsystems. Tuning Garbage Collection with the 5.0 Java Virtual Machine. http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html, (28.06.2007), 2004.

[Sun06a] Sun Microsystems. Java SE Monitoring and Management Guide - Chapter 3. Using JConsole. http://java.sun.com/javase/6/docs/technotes/guides/management/jconsole.html, (30.06.2007), 2006.

[Sun06b] Sun Microsystems. Monitoring and Management for the Java Platform (JavaSE 6). http://java.sun.com/javase/6/docs/technotes/guides/management/, (01.07.2007), 2006.

[Sun07a] Sun Microsystems. Sun Developer Network - (SDN), Java SE Downloads. http://java.sun.com/javase/downloads/index.jsp, (25.06.2007), June 2007.

[Sun07b] Sun Microsystems. The Fortress Language Specification Version 1.0 β. `http://research.sun.com/projects/plrg/Publications/fortress1.0beta.pdf`,(31.05.2007), 2007.

[Sun07c] A. Sundararajan. A. sundararajan's weblog - querying java heap with oql. `http://blogs.sun.com/sundararajan/entry/querying_java_heap_with_oql`, (02.07.2007), Apr. 2007.

[Sun08] Sun Microsystems. Java Web Start Technology. `http://java.sun.com/products/javawebstart/`, (01.02.2008), 2008.

[SWF+04] Larry A Sonna, C. Bruce Wenger, Scott Flinn, Holly K Sheldon, Michael N Sawka, and Craig M Lilly. Exertional heat injury and gene expression changes: a dna microarray analysis study. *J Appl Physiol*, 96(5):1943–1953, May 2004.

[Sym05] S. Symons. MayDB - The Mayday Database. Studienarbeit, Universität Tübingen, 2005.

[Sym06] S. Symons. Machine Learning Algorithms for Microarray Data. Diplomarbeit, Universität Tübingen, 2006.

[SYS03] Gordon K Smyth, Yee Hwa Yang, and Terry Speed. Statistical issues in cdna microarray data analysis. *Methods Mol Biol*, 224:111–136, 2003.

[TC05] James J. Thomas and Kristin A. Cook, editors. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Press, 2005. http://nvac.pnl.gov/agenda.stm (13.04.2007).

[TGB+02] Olga G Troyanskaya, Mitchell E Garber, Patrick O Brown, David Botstein, and Russ B Altman. Nonparametric methods for identifying differentially expressed genes in microarray data. *Bioinformatics*, 18(11):1454–1461, Nov 2002.

[The99] M. Theus. JAVA - The Next Generation of Statistical Computing? In *Computer Science and Statistics, Proceedings of the 30th Symposium on the Interface 1998*, 1999.

[The05] The JAMA Team. JAMA - A Java Matrix Package, Version 1.0.2. `http://math.nist.gov/javanumerics/jama/`, July 2005. (16.07.2007).

[The06a] The Apache Software Foundation. The Jakarta Project - Commons Primitives.

http://jakarta.apache.org/commons/primitives/, June 2006. (02.07.2007).

[The06b] The MTJ Team. Matrix Toolkits for Java (MTJ), Version 0.9.9. http://rs.cipr.uib.no/mtj/, Nov. 2006. (16.07.2007).

[The07a] The Apache Software Foundation. Apache-Server. http://httpd.apache.org, 2007. (20.04.2007).

[The07b] The Apache Software Foundation. The Jakarta Project - Commons-Math: The Jakarta Mathematics Library. http://jakarta.apache.org/commons/math/, 2007. (09.07.2007).

[The08] The Subversion Developers. Subversion. http://subversion.tigris.org/, (21.04.2008), 2008.

[Tho06] J. Thomas. Visual analytics: a grand challenge in science - turning information overload into the opportunity of the decade. In *Symposium on the Future of Visualization*, Charlotte, 2006. http://www.viscenter.uncc.edu/symposium06/OnlineContent/Slides%5CThomasSlides.pdf.

[Tho07] B. Thompson. *The Nature of Statistical Evidence*. Springer Science+Business Media Inc., 2007.

[Tie94] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701 – 1728, Dec. 1994.

[TOR+01] G. Tseng, M. Oh, L. Rohlin, J. Liao, and W. Wong. Issues in cdna microarray analysis: quality filtering, channel normalization, models of variations and assesment of gene effects. *Nucleic Acids Research*, 29(12):2549 – 2557, 2001.

[TOTZ01] J.G. Thomas, J.M. Olson, S.J. Tapscott, and L.P. Zhao. An Efficient and Robust Statistical Modeling Approach to Discover Differentially Expressed Genes Using Genomic Expression Profiles. *Genome Res.*, 11(7):1227 – 1236, 2001.

[Tra08] Trac Team. trac – Integrated SCM & Project Management. http://trac.edgewall.org/, (21.04.2008), 2008.

[TTC01] V. G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proc Natl Acad Sci U S A*, 98(9):5116–5121, Apr 2001.

[Tuf83] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, USA, 1983.

[Tuf90] Edward R. Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, USA, 1990.

[Tuk77] J. W. Tukey. *Exploratory data analysis*. Addison-Wesley Series in Behavioral Science: Quantitative Methods, Reading, Mass.: Addison-Wesley, 1977.

[TVCS06] Carl Troein, Johan Vallon-Christersson, and Lao H Saal. An introduction to bioarray software environment. *Methods Enzymol*, 411:99–119, 2006.

[TvMK+05] Susannah Green Tringe, Christian von Mering, Arthur Kobayashi, Asaf A. Salamov, Kevin Chen, Hwai W. Chang, Mircea Podar, Jay M. Short, Eric J. Mathur, John C. Detter, Peer Bork, Philip Hugenholtz, and Edward M. Rubin. Comparative Metagenomics of Microbial Communities. *Science*, 308(5721):554–557, 2005.

[Val07] Valgrind Developers. Valgrind. `http://valgrind.org/`, (09.07.2007), 2007.

[vdWBvdL99] M. A. van de Wiel, A. Di Bucchianico, and P. van der Laan. Symbolic Computation and Exact Distributions of Nonparametric Test Statistics. *The Statistician*, 48(4):507 – 516, 1999.

[Vis06a] Visual Numerics. JMSL Numerical Library for Java Applications. `http://www.vni.com/products/imsl/jmsl/jmsl.php`, 2006. (16.07.2007).

[Vis06b] Visual Numerics. JMSL Numerical Library Version 4.0 - Function Catalog. `http://www.vni.com/books/dod/pdf/jmsl/JMSL40FunCat.pdf`, 2006. (16.07.2007).

[Vli99] J. Vlissides. *Entwurfsmuster anwenden*. Number 3827315441. Addison-Wesley Longman Verlag GmbH, 1999.

[VR02] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, fourth edition, 2002.

[VTPF92] W. T. Vetterling, S. A. Teukolsky, W. H. Press, and B. R. Flannery. *Numerical Recipes: Example Book (C)*. Cambridge University Press, 2. edition, 1992.

[WAG06] Leland Wilkinson, Anushka Anand, and Robert Grossman. High-Dimensional Visual Analytics: Interactive Exploration Guided by Pairwise Views of Point Distributions. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1363–1372, 2006.

[Wal03] G. Walz, editor. *Lexikon der Mathematik*. Number 978-3827404371. Spektrum Akademischer Verlag Heidelberg Berlin, 2003.

[Wan71] Y.Y. Wang. Probabilities of the Type I Errors of the Welch Tests for the Behrens-Fisher Problem. *Journal of the American Statistical Association*, 66(335):605 – 608, Sep. 1971.

[Wan08] J. Wang. The Science Creative Quarterly - Image (bank). `http://www.scq.ubc.ca/image-bank/`, (08.05.2008), 2008.

[Wel38] B. L. Welch. The Significance of the Difference Between Two Means when the Population Variances are Unequal. *Biometrika*, 29(3/4):350 – 362, Feb. 1938.

[Wel47] B. L. Welch. The Generalization of 'Student's' Problem when Several Different Population Variances are Involved. *Biometrika*, 34(1/2):28 – 35, Jan. 1947.

[WF05] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2. edition, 2005.

[WH07] C. S. Wells and J. M. Hintze. Dealing with assumptions underlying statistical tests. *Psychology in the Schools*, 44(5):495–502, 2007.

[Wil45] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80 – 83, Dec. 1945.

[Wil05] Leland Wilkinson. *The Grammar of Graphics*. Springer Science+Business Media Inc., second edition, 2005.

[Wil06] G.V. Wilson. Where's the Real Bottleneck in Scientific Computing? *American Scientist*, 94(1):5, 2006.

[Wil08] G.V. Wilson. Software Carpentry. `http://www.swc.scipy.org`,(21.04.2008), 2008.

[WM04] E. Wit and J. McClure. *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd, 2004.

[Wol07] Wolfram Research Inc. Wolfram Mathematica. `http://www.wolfram.com/products/mathematica/index.html`, (24.09.2007), Sep. 2007.

[XL03] R. Xu and X. Li. A comparison of parametric versus permutation methods with applications to general and temporal microarray gene expression data. *Bioinformatics*, 19(10):1284 – 1289, 2003.

[Yat34] F. Yates. Contigency Tables Involving Small Numbers and the χ^2 Test. *Journal of Royal Statistical Society (Supplement)*, 1(2):217 – 235, 1934.

[YBDS02] Y.H. Yang, M.J. Buckley, S. Dudoit, and T.P. Speed. Comparison of methods for image analysis on cDNA microarray data. *Journal of Computational and Graphic Statistics*, 11:108 – 136, 2002.

[YDLS01] Y. H. Yang, S. Dudoit, P. Luu, and T. Speed. Normalization for cDNA microarray data. 4266 of Proceedings of SPIE BiOS, 2001. `http://www.stat.berkeley.edu/users/terry/zarray/Html/normspie.html`, (22.08.2006).

[YSN+90] R. Yehuda, S. M. Southwick, G. Nussbaum, V. Wahby, E. L. Giller, and J. W. Mason. Low urinary cortisol excretion in patients with posttraumatic stress disorder. *J Nerv Ment Dis*, 178(6):366–369, Jun 1990.

[ZDS+06] M. Zschunke, K. Deubel, S. Symons, J. Dietzsch, and K. Nieselt. Poster: Fage and vega: two new tools for functional and epigenomics expression analysis in mayday. In *German Conference on Bioinformatics (GCB)*, Tübingen, 2006.

[ZFD+05] Derek Zieker, Elvira Fehrenbach, Janko Dietzsch, Judith Fliegner, Marc Waidmann, Kay Nieselt, Peter Gebicke-Haerter, Rainer Spanagel, Perikles Simon, Andreas Michael Niess, and Hinnak Northoff. cDNA microarray analysis reveals novel candidate genes expressed in human peripheral blood following exhaustive exercise. *Physiol Genomics*, 23(3):287–294, Nov 2005.

[Zie06] D. Zieker. *Etablierung von Home-Made cDNA Microarrays und Untersuchung der Genexpression nach Hochleistungssport*. Dissertation, Universität Tübingen, 2006.

[Zie07] J. Zieker. *Etablierung von home-made cDNA Microarrays und Untersuchung der Genexpression bei Posttraumatischen Belastungströrungen*. Dissertation, Universität Tübingen, 2007.

[ZKT+08] Derek Zieker, Ingmar Königsrainer, Frank Traub, Kay Nieselt, Bettina Knapp, Christian Schillinger, Christian Stirnkorb, Falko Fend, Hinnak Northoff, Susan Kupka, Björn L D M Brücher, and Alfred Königsrainer. PGK1 a potential marker for peritoneal dissemination in gastric cancer. *Cell Physiol Biochem*, 21(5-6):429–436, 2008.

[ZP03] Yanli Zhao and Wei Pan. Modified nonparametric approaches to detecting differentially expressed genes in replicated microarray experiments. *Bioinformatics*, 19(9):1046–1054, Jun 2003.

[Zsc04] Matthias Zschunke. Connecting R to Mayday. Studienarbeit, Universität Tübingen, 2004.

[Zsc06] Matthias Zschunke. Comparative Analysis and Visualization of Epigenomic and Gene Expression Microarray Data. Diplomarbeit, Universität Tübingen, 2006.

[ZZD+05] Derek Zieker, Judith Zieker, Janko Dietzsch, Michael Burnet, Hinnak Northoff, and Elvira Fehrenbach. cDNA-microarray analysis as a research tool for expression profiling in human peripheral blood following exercise. *Exerc Immunol Rev*, 11:86–96, 2005.

[ZZJ+07] J. Zieker, D. Zieker, A. Jatzko, J. Dietzsch, K. Nieselt, A. Schmitt, T. Bertsch, K. Fassbender, R. Spanagel, H. Northoff, and P. J. Gebicke-Haerter. Differential gene expression in peripheral blood of patients suffering from post-traumatic stress disorder. *Mol Psychiatry*, 12(2):116–118, Feb 2007.

Index

Fett gedruckte Seitenzahlen sind Verweise auf die Definition des zugehörigen Begriffes. Normal gedruckte Seitenzahlen geben lediglich das Vorkommen des Begriffes an.