TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

DISSERTATION

Analysis of Gene Expression Time–Course Data Using Cluster Techniques

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Univ.-Prof. Dipl.-Ing. Dr.techn. Friedrich Leisch
Institut für Statistik
Ludwig-Maximilians-Universität München

eingereicht an der Technischen Universität Wien
bei der Fakultät für Mathematik und Geoinformation

von

Dipl.-Ing. Theresa Scharl-Hirsch
Matrikelnummer: 9825831
Liebhartsgasse 40/20
1160 Wien

Wien, im September 2009

Kurzfassung

Diese Dissertation beschäftigt sich mit verschiedenen Aspekten der Cluster Analyse zur Auswertung von Zeitreihen Microarray Daten. Seit einigen Jahren ist die Interpretation von riesigen Datenmengen aus Microarray Experimenten eine große Herausforderung für die Statistik und Bioinformatik. Zeitreihen Microarray Experimente machen es möglich, die Genexpression von tausenden von Genen simultan zu studieren. Da Gene mit ähnlichem Expressionsmuster häufig auch koreguliert sind, kann das Clustern von Genexpressionsverläufen dabei helfen, koregulierte Gene zu finden. Letztendlich kann die Cluster Analyse dabei unterstützen, funktionale Stoffwechselwege und Interaktionen zwischen Genen zu finden.

In dieser Dissertation werden sowohl partitionierende Cluster Methoden wie K–Means und der qualitätsbasierte Cluster Algorithmus QT–Clust als auch modellbasiertes Clustern untersucht. Es werden entweder die Originaldaten geclustert oder die funktionalen Daten. In der funktionalen Datenanalyse wird eine Kurve an jede Beobachtung angepasst, um die Zeitabhängigkeit zu berücksichtigen. In Simulationsstudien auf künstlichen Datensätzen werden die Eigenschaften unterschiedlicher Clustermethoden untersucht und auf ihre Nützlichkeit für Echtdaten getestet. Neue Clustermethoden für diese Art von Daten werden vorgestellt sowie einige Methoden zur Evaluierung von Clusterlösungen. Alle Cluster Algorithmen and Evaluierungsmethoden wurden in R implementiert, und alle Simulationen wurden in R durchführt.

Ein wesentlicher Teil der Arbeit konzentriert sich auf die explorative Analyse von Clusterlösungen. Da genetische Interaktionen sehr komplex sind, ist die Definition von Genclustern schwierig. Beziehungen zwischen Clustern sind von großer Bedeutung, da koexprimierte Gene sehr leicht in unterschiedliche Cluster gruppiert werden können. Die Visualisierung von Clusterlösungen hilft dabei, ein besseres Verständnis für die Clusterstruktur der Daten zu bekommen und erleichtert die Interpretation der Clusterlösungen. Nachbarschaftsgraphen ermöglichen eine graphische Darstellung der Beziehungen zwischen angrenzenden Clustern. Unterschiedliche Visualisierungsmethoden zur interaktiven Untersuchung von Clusterlösungen wurden entwickelt und im R Paket **gcExplorer** implementiert. Die Funktionalität des Pakets beinhaltet die Visualisierung der Clusterstruktur, die Darstellung einzelner Cluster in Form von Graphiken oder HTML Tabellen, das Hervorheben bestimmter Eigenschaften von Clustern sowie einige Testprozeduren zur Beurteilung der Qualität von Clusterlösungen. Schließlich wird die Anwendung der verschiedenen Clustermethoden und die Verwendung des Pakets an mehreren Beispielen mit *E. coli* Daten vom Department für Biotechnologie an der Universität für Bodenkultur in Wien veranschaulicht.

Abstract

This thesis is concerned with different aspects of the analysis of gene expression time–course data using cluster techniques. The interpretation of enormous amounts of data from microarrays has been a challenging task in statistics and bioinformatics for the past few years. Time–course microarray experiments make it possible to look at the gene expression of thousands of genes at several time points simultaneously. Genes with similar expression pattern are likely to be co–regulated. Hence clustering gene expression patterns may help to find groups of co–regulated genes or to identify common temporal or spatial expression patterns. Finally cluster results can suggest functional pathways and interaction between genes.

The cluster methods investigated in this thesis include partitioning cluster methods like the well–known K–Means or the quality–based cluster algorithm Stochastic QT–Clust as well as model–based clustering. Clustering is either carried out on the raw data or on functional data. In functional data analysis a curve is fit to each observation in order to account for time dependency. In simulation studies on artificial and real data sets from publicly available databases the properties of different cluster methods are compared and evaluated using the adjusted Rand index, the sum of within cluster distances as well as the likelihood criterion. Additionally, test procedures are developed allowing to judge the biological relevance of cluster solutions. All cluster algorithms and evaluation procedures are implemented in the statistical computing environment R and all simulations are performed in R.

An essential part of this thesis deals with the visualization of cluster solutions. The definition of gene clusters is not very clear as genetic interactions are extremely complex. For this reason the relationships between clusters are very important as co–expressed genes can end up in different clusters. The visualization of cluster solutions helps to get an understanding of the cluster structure of the data and makes it easier to interpret the cluster results. Neighborhood graphs allow for visual assessment of relationships between adjacent clusters. A new visualization toolbox for the interactive exploration of cluster solutions is implemented in R package **gcExplorer**. The functionality of the package includes the visualization of the cluster structure in form of neighborhood graphs, the display of gene clusters in graphics or HTML tables, highlighting additional properties of the clusters as well as test procedures to judge the quality of cluster solutions. Finally, the methods are applied to *E. coli* data sets from the Department of Biotechnology at the University of Natural Resources and Applied Life Sciences in Vienna.

Acknowledgement

During my work on this thesis is was supported by the Austrian K_{ind}/K_{net} Center of Biopharmaceutical Technology (ACBT).

I would like to thank my supervisor Friedrich Leisch for his many suggestions and fruitful discussions. Many thanks to Karl Bayer for his vision about bioinformatics in the working group of Microbial Fermentation and constant support during this work. Many thanks to Bettina Grün and all my colleagues at the Boku, especially to Karin Dürrschmid, Karoline Marisch, Florentina Pötschacher and Gerald Striedner for the warm welcome and helpful advices.

I would like to thank my friends and my family for their encouragement and for taking my mind off my worries. Special thanks to Clemens Hirsch for his help and understanding.

Contents

Chapter 1

Introduction

The implementation of comprehensive analysis tools from systems biology into bioprocess development concepts enables the change from empirical to rational knowledge based approaches in host engineering and process design. As protein synthesis is a complex process genome wide understanding of cellular stress reactions due to recombinant gene expression is highly desirable. DNA microarrays are powerful, state of the art tools for the monitoring of cellular systems on transcriptome level providing insight into cellular response to defined changes in cultivation conditions, e.g induction of recombinant protein production. As recombinant proteins differ from endogenous proteins and have a distinct composition it is of high interest to reveal the most relevant pathways which are mainly concerned with recombinant protein production. To enable interpretation of results the most significant information must be extracted from the acquired microarray data by using optimally suited methods of statistics and bioinformatics. The enormous amounts of data from gene expression microarray experiments are typically controlled by dividing the whole data set into homogeneous subgroups with distinct properties, i.e., by cluster analysis.

This thesis is concerned with cluster analysis of gene expression time–course microarray data. Depending on the objective of a grouping e.g., clusters of highly correlated genes or tight clusters, different cluster approaches have to be used. The behavior of selected cluster methods including partitioning and model–based clustering is investigated on various artificial as well as real time–course data sets. Classification criteria are presented as well as inferential methods to judge the quality of a given clustering. Further, external information about the genes like gene function or association to gene sets are integrated into the exploration.

To complement the simulation results visualization tools are presented to interactively investigate the goodness of cluster results. Neighborhood graphs are used for visual assessment of the cluster structure. Several node functions and panel functions are implemented allowing to explore cluster solutions is more detail.

The new visualization methods as well as the test procedures are implemented in the package **gcExplorer** (Scharl and Leisch 2009a) in R, an environment for statistical computing and graphics (R Development Core Team 2009). The package design and implementational details are described and illustrated in applications. All computations in this thesis are made in R using packages **flexclust** (Leisch 2006) and **flexmix** (Grün and Leisch 2008).

1.1 Microarray data

DNA-microarrays are tools to study the expression level of thousands of individual DNA sequences simultaneously. They are now used in many different contexts to compare Messenger–RNA (mRNA) levels between two or more samples of cells. The two cell samples to be compared for gene expression are often cells subject to some treatment and normal cells. Microarray experiments typically give expression measurements on a large number of genes but with only few replicates for each gene. Reviews of microarray data analysis include Speed (2003), Parmigiani et al. (2003) and Gentleman et al. (2005).

Using microarrays different properties of gene expression can be studied, such as expression at the transcription or translation level, and subcellular localization of gene products. In this thesis attention focusses on expression at the transcription level, i.e., on mRNA levels. Although the regulation of protein synthesis in a cell is by no means controlled solely by mRNA levels, mRNA levels sensitively reflect the type and state of the cell. Microarrays derive their power and universality from a key property of DNA molecules: complementary base–pairing. The term *hybridization* refers to the annealing of nucleic acid strands from different sources according to the base–pairing rules. To utilize the hybridization property of DNA, complementary DNA or cDNA is obtained from mRNA by reverse transcription. There are different types of microarray systems, including cDNA microarrays and high–density oligonucleotide arrays.

Oligonucleotide microarrays consist of thousands of short sequences designed to match parts of the sequence of known or predicted open reading frames printed in a high–density array on a glass microscope slide using a robotic printer or arrayer. In order to evaluate the relative abundance of these spotted oligo sequences in two DNA or RNA samples the differential hybridization of the two samples has to be controlled. For mRNA samples, the two targets are reverse–transcribed into cDNA, labelled using different fluorescent dyes (usually a red–fluorescent dye, Cyanine 5 or Cy5, and a green–fluorescent dye, Cyanine 3 or Cy3), then mixed in equal proportions and hybridized with the arrayed oligo sequences or probes (following the definition of probe and target adopted in "The Chipping Forecast", a January 1999 supplement to Nature Genetics). After this competitive hybridization, the slides are imaged using a scanner and fluorescence measurements are made separately for each dye at each spot on the array. The ratio of the red and green fluorescence intensities for each spot is indicative of the relative abundance of the corresponding DNA probe in the two nucleic acid target samples.

1.2 Cluster analysis

Cluster analysis is frequently used in gene expression data analysis to find groups of co–expressed genes which can finally suggest functional pathways and interactions between genes. Clusters of co–expressed genes can help to discover potentially co–regulated genes or genes associated to conditions under investigation, e.g., different induction strategies. Usually cluster analysis provides a good initial investigation of microarray data before actually focusing on smaller gene groups of interest.

Clustering is commonly used to reduce the complexity of the data from multidimensional space to a single nominal variable, the cluster membership. In the analysis of microarray data clustering is used as vector quantization because no clear density clusters exist in the data. Genetic interactions are so complex that the definition of gene clusters is not clear. Additionally microarray data are very noisy and co–expressed genes can end up in different clusters. Therefore the set of genes is divided into artificial subsets where relationships between clusters play an important role. Depending on the purpose of the cluster analysis different numbers of clusters can be appropriate. Few large clusters are typically used for a broad overview of a data set and many small clusters are more suitable to detect co–regulated genes (e.g., over 25 clusters in Heyer et al. (1999)).

In the literature numerous methods for clustering gene expression data have been proposed. Detailed reviews of currently used methods and challenges with gene expression data are given in Sheng et al. (2005); Androulakis et al. (2007); Kerr et al. (2008). Besides traditional methods like hierarchical clustering , K–means (MacQueen 1967; Hartigan and Wong 1979), partitioning around medoids (PAM, K–medoids, Kaufman and Rousseeuw (1990)) or self–organizing maps (Kohonen 1989) there are several algorithms dealing with time–course gene expression data (e.g., Heyer et al. (1999), De Smet et al. (2002), Ben-

Dor et al. (1999), and Bickel (2003)). Additionally model–based clustering (eg., Fraley and Raftery (1998) or McLachlan et al. (2002)) is frequently used.

The display of cluster solutions particularly for a large number of clusters is very important in exploratory data analysis. See Leisch (2008) for an overview of cluster visualization. Visualization methods are necessary in order to make cluster analysis useful for practitioners. They give an understanding of the relationships between segments of a partition and make it easier to interpret the cluster results. In hierarchical clustering dendrograms and heatmaps are routinely used (e.g., Eisen et al. (1998)). The most popular group of partitioning cluster algorithms are centroid–based cluster algorithms (e.g., K–means or PAM). Once a set of centroids has been found centroid–based cluster solutions are usually visualized by projection of the data into two dimensions (e.g., by principal component analysis). Silhouette plots (Rousseeuw 1987) can be used to check whether clusters of points are well separated whereas topology representing networks (Martinetz and Schulten 1994) reveal similarity between clusters. Neighborhood graphs (Leisch 2006) combine these two approaches to visualize cluster structure. They can be used for visual assessment of the cluster structure of centroid–based cluster solutions. In a neighborhood graph each cluster is represented by a node. The similarity of two clusters is measured by the weighted percentage of data points that have one of the corresponding cluster centroids as closest and the other as second-closest, respectively.

1.3 Recombinant protein production

The optimization of manufacturing processes for biopharmaceutical products, i.e., the integrated development from "Gene to Product" is the major challenge in the biopharmaceutical industry. Host/vector systems are adapted to process needs by tuning of the recombinant gene expression rate in relation to metabolic capabilities of the host organism. Main objective is to achieve a significant improvement of the efficiency of biopharmaceutical process development in order to provide widely optimized manufacturing processes for clinical trials and mass production of biopharmceuticals in as short as possible time. This shall reduce development times drastically providing a faster access to innovative and highly effective biopharmaceutical drugs. The produced proteins are therapeutically active and have individual effects on the host cell metabolism.

The successful application of microarrays as monitoring tool in bioprocess development strongly depends on concerted design of cultivation experiments as well as array experiments and systematic data analysis. The development of a process monitoring and control platform allows reproducible cultivation conditions with reliable and defined samples.

Process Optimization for *Escherichia coli* host/vector systems is performed in a systemsbiological approach using transcriptome and proteome methods (Dürrschmid et al. 2008). A systems based understanding of the interaction between the host metabolism regulatory pathways, recombinant gene expression and the environment is necessary to identify key analytes, marker genes and proteins. Genome and proteome platforms are applied to monitor the impact of recombinant gene expression on the host metabolism, e.g., stress response and metabolic bottlenecks.

Several *E. coli* data sets from the Department of Biotechnology at the University of Natural Resources and Applied Life Scienes in Vienna are used in this thesis. In Chapter 5 as well as Section 6.1 and Appendix A the data of two cultivation processes is used. Two recombinant *E. coli* processes with different induction strategies were conducted in order to evaluate the influence of the expression level of the inclusion body forming protein N^{pro}GFPmut3.1 on the host metabolism. The standard strategy with a single pulse of inducer yielding in a fully induced system (in the following called experiment A or PS17) was compared to a process with continuous supply of limiting amounts of IPTG (Isopropyl–β–D–thiogalactopyranosid) inducer resulting in a partially induced system (in the following called experiment B or PS19) (Striedner et al. 2003). Comparative analysis of data sets from independent experiments provide additional information and contributes to the optimal exploitation of microarray data. Details about the two data sets can be found in Section 6.1.1.

In Section 6.2 another *E. coli* data set is used. The goal of this experiment is the detailed investigation of the cellular response of *E. coli* BL21(DE3) to high level expression of recombinant human super–oxide–dismutase (SOD) on the transcriptional level. Three biological replicates were generated by using a carbon limited exponential feedbatch cultivation similar to industrial setups for large scale production. For induction of the system a single pulse of IPTG yielding in a fully induced system is applied one doubling past feed start. In order to achieve a proper time–resolution of cellular responses sampling starts with a high frequency for the first hours past induction and decreases in the course of the experiment.

1.3.1 *E. coli* databases

Cluster analysis is used to find groups of co–expressed genes in the microarray data without prior knowledge about the gene functions. However, by clustering expression profiles of co–expressed genes groups of genes with similar function are often found.

The annotation of genes to categories or classes is a very important aspect in the analysis of gene expression data. The genes can for example be mapped to functional groups like Gene Ontology (GO, The Gene Ontology Consortium (2000)) classifications or to protein complexes. Gene functions are very complex, therefore genes are usually mapped to multiple classes. In any case the mapping is known a priori and does not depend on the data of the currently investigated experiment.

External information about the annotation of genes to functional groups can easily be included in exploratory and inferential analysis of gene expression data, e.g., the accumulation of gene ontology (GO) classifications in certain gene clusters. In microarray data analysis gene ontology classifications about Biological Process, Molecular Function and Cellular Component are typically investigated. Further sources of external knowledge for data from *E. coli* are the GenProtEC (Serres et al. (2004), `http://genprotec.mbl.edu/`) classification system for cellular and physiological roles of *E. coli* gene products and the RegulonDB (Salgado et al. (2006), `http://regulondb.ccg.unam.mx/`) for detailed information about operons and regulons.

1.4 Overview of the thesis

This thesis focuses on different aspects of cluster analysis. Starting with different cluster algorithms and distance measures used it discusses the evaluation and validation of cluster solutions as well as the graphical exploration using the **gcExplorer** (Scharl and Leisch 2009a).

Chapter 2 presents the cluster methods used including the stochastic QT–Clust algorithm (Scharl and Leisch 2006b), quality–based clustering of functional data (Scharl and Leisch 2009b) as well as new Jackknife distance measures (Scharl and Leisch 2006a). Additionally initialization strategies for model–based clustering of time–course gene expression data are described (Scharl et al. 2009a).

Chapter 3 presents different evaluation methods for cluster solutions including modified versions of the adjusted Rand index (Hubert and Arabie 1985) and test procedures for external validation of a given clustering (Scharl et al. 2009c).

In Chapter 4 several simulation studies on various artificial as well as real time–course data sets are presented.

Chapter 5 discusses the implementation in R (R Development Core Team 2009).

Chapter 6 gives results of applications of the investigated methods on *E. coli* data (Scharl et al. 2009b; Scharl and Leisch 2008a).

Chapter 7 summarizes the main findings of the thesis. Appendix A contains the vignette of **gcExplorer** and Appendix B contains the documentation of the package.

1.4.1 Publications used

This thesis is based on several publications as well as unpublished work.

Scharl, T., Grün, B., and Leisch, F. (2009a). Model–based clustering of time–course gene expression data: Evaluation of initialization and random effects. *Submitted.*

Scharl, T. and Leisch, F. (2006a). Jackknife distances for clustering time–course gene expression data. In *2006 JSM Proceedings*, pages 346–353. American Statistical Association, Alexandria, USA.

Scharl, T. and Leisch, F. (2006b). The stochastic qt-clust algorithm: evaluation of stability and variance on time-course microarray data. In Rizzi, A. and Vichi, M., editors, *Compstat 2006—Proceedings in Computational Statistics*, pages 1015–1022. Physica Verlag, Heidelberg, Germany.

Scharl, T. and Leisch, F. (2008a). Using neighborhood graphs for the investigation of *E. coli* gene clusters. In Ahdesmäki, M., Strimmer, K., Radde, N., Rahnenführer, J., Klemm, K., Lähdesmäki, H., and Yli-Harja, O., editors, *Proceedings of the 5th International Workshop on Computational Systems Biology, WCSB 2008 (June 11-13, 2008, Leipzig, Germany)*, pages 157–160. Tampere University of Technology, Tampere, Finland.

Scharl, T. and Leisch, F. (2008b). Visualizing gene clusters using neighborhood graphs in r. In Brito, P., editor, *Proceedings of COMPSTAT'2008, International Conference on Computational Statistics, Porto - Portugal, August 24th-29th 2008*, pages 51 –58. Physica–Verlag.

Scharl, T. and Leisch, F. (2009a). gcExplorer: Interactive exploration of gene clusters. *Bioinformatics*, 25(8):1089–1090.

Scharl, T. and Leisch, F. (2009b). Quality–based clustering of functional data: Applications to time course microarray data. In Fink, A., Lausen, B., Seidel, W., and Ultsch, A., editors, *Advances in Data Analysis, Data Handling and Business Intelligence, Proceedings of the 32nd Annual Conference of the Gesellschaft für Klassifikation e.V., Joint Conference with the British Classification Society (BCS) and the Dutch/Flemish Classification Society (VOC), Helmut–Schmidt–University, Hamburg, July 16–18, 2008*, Studies in Classification, Data Analysis, and Knowledge Organization. Springer Verlag. Accepted for publication.

Scharl, T., Striedner, G., Pötschacher, F., Leisch, F., and Bayer, K. (2009b). Interactive visualization of clusters in microarray data: an efficient tool for improved metabolic analysis of *E. coli*. *Microbial Cell Factories*, 8:37.

Scharl, T., Voglhuber, I., and Leisch, F. (2009c). Exploratory and inferential analysis of gene cluster neighborhood graphs. *BMC Bioinformatics*. Accepted for publication.

Chapter 2

Methods

2.1 Partitioning cluster methods

2.1.1 K–Means

For a given data set $X_N = \{x_1, \ldots, x_N\}$ the distance between points x and y is given by $d(x, y)$, e.g., the Euclidean or absolute distance. $C_K = \{c_1, \ldots, c_K\}$ is a set of centroids and the centroid closest to x is denoted by

$$c(x) = \text{argmin}_{c \in C_K} d(x, c).$$

The set of all points where c_k is the closest centroid is given by

$$A_k = \{x_n | c(x_n) = c_k\}.$$

Minimizing the average distance between each data point and its closest centroid

$$D(X_n, C_K) = \frac{1}{N} \sum_{n=1}^{N} d(x_n, c(x_n)) \to \min_{C_K}$$

is the task of most cluster algorithms.

The well–known K–Means algorithm (MacQueen 1967; Hartigan and Wong 1979) works as follows. First the number k of clusters has to be specified:

1. Start with k randomly chosen centers.

2. Assign each point to its nearest center.

3. Compute new centroids minimizing the average distance.
4. If there have been changes from the previous iteration,
   goto 2.

2.1.2 Quality threshold clustering

The quality–based cluster algorithm stochastic QT–Clust (Scharl and Leisch 2006b) is an adaptation of the original QT–Clust algorithm proposed by Heyer et al. (1999). In contrast to cluster algorithms like K–means where the number of clusters is defined a priori the quality of clusters is the central parameter now. The quality of a cluster is given by the maximum diameter of the cluster. The possibility to tune the quality of clusters is very helpful for practitioners. Depending on the goal of the experiment different properties of the clusters are desirable which can either be a few rather large clusters or many small clusters with very specific expression patterns. Additionally the minimum number of points that form a single cluster is chosen. Microarray data are noisy data and outliers can easily distort cluster solutions. Stochastic QT–Clust is robust to outliers as outlier observations will not be added to any cluster. Hence the number of clusters is controlled indirectly through these two parameters. A further tuning parameter is the number `ntry` of candidate clusters generated in each run. The algorithm works as follows:

1. Start with a randomly chosen centroid.
2. Iteratively add the gene that minimizes the increase in cluster diameter.
3. Continue until no gene can be added without surpassing the diameter threshold.
4. Repeat from 1. for `ntry` $-$ 1 further centroids.
5. Select the largest candidate cluster and remove the genes it contains from further consideration.
6. Goto 1. on the smaller data set.
7. Stop when the largest remaining cluster has fewer than some prespecified number of elements.

If `ntry` is equal to the number of genes G the original QT–Clust algorithm is obtained. Stochastic QT–Clust speeds up the procedure and yields different local maxima of the objective function. The original algorithm will always converge in the same local optimum. The impact of the hyperparameter `ntry` is evaluated in Section 4.2.1. Throughout the remaining simulations `ntry` is equal to 5.

In order to gain maximum information the choice of the cluster diameter and the minimum number of points has to be carefully chosen as both have a large impact on the resulting clustering and its interpretation. A small diameter will yield a cluster solution with many small clusters containing genes with very similar expression patterns whereas a larger diameter will result in a smaller number of less tight clusters. Additionally, if the diameter is chosen too small many genes cannot be added to a cluster and will be treated as outliers. The minimum number of points also has a big influence on the number of clusters and the number of outliers. If small clusters are allowed (e.g., the minimum number of points is 2) there will be less outliers than in the case of a larger minimum number of points. There is a tradeoff between the number of clusters, the size of the clusters and the number of outliers. Therefore it is necessary to finetune these parameters for each data set to obtain a cluster solution that fits the needs of the current experiment. The relationship between the radius, the number of clusters and the number of outliers is investigated in Section 4.1.3.

2.1.3 Distance measures

The distance measure used has major impact on the resulting clusters (Gentleman et al. 2005). The properties of different distance measures have to be investigated to be able to answer biological questions more precisely. A comparison of different distance measures which are commonly used in the context of clustering time–course microarray data was performed in Scharl and Leisch (2006a).

Four distance measures were chosen which are commonly used in the context of clustering time–course microarray data (see for example Chipman et al. (2003); Jiang et al. (2004); Sheng et al. (2005); Gentleman et al. (2005)). Here three geometric distances and "1 - Correlation" distance are used.

One of the most commonly used methods to measure the distance between two data objects is *Euclidean distance* which is given by

$$d_{xy} = \sqrt{\sum_{i=1}^{T} (x_i - y_i)^2},$$

where x and y are T–dimensional vectors and T is the number of time points in the experiment. *Manhattan distance*

$$d_{xy} = \sum_{i=1}^{T} |x_i - y_i|$$

is more robust to outliers than Euclidean distance. Both Euclidean and Manhattan distance yield clusters with a certain band width which can vary from one time point to the next. *Maximum distance*

$$d_{xy} = \max |x_i - y_i|$$

looks at the maximum differences between time points and yields clusters of a fixed band width.

If one is interested in the relative changes of gene expression a *correlation*–based distance measure is more appropriate as correlation is invariant to location and scale. The dissimilarity between two gene profiles can be defined as

$$d_{xy} = 1 - \rho_{xy} = 1 - \frac{\sum_{i=1}^{T}(x_i - \bar{x})(y_i - \bar{y})}{[\sum_{i=1}^{T}(x_i - \bar{x})^2]^{1/2}[\sum_{i=1}^{T}(y_i - \bar{y})^2]^{1/2}}$$

where ρ_{xy} is the Pearson sample correlation coefficient. This distance measure removes changes in the average or range of the expression level from one gene to the next. Both strongly positively correlated as well as negatively correlated genes are considered co–expressed.

Jackknife distance measures

A possible problem using these distance measures for clustering time–course gene expression data is that single outlier variables can completely change the expression pattern of certain genes. There are several algorithms which are able to deal with outlier observations. Partitioning around medoids described in Kaufman and Rousseeuw (1990) is a more robust version of k–means for arbitrary distance measures. Trimmed K-means (Cuesta-Albertos et al. 1997) is a robust version of the original algorithm. All these algorithms can handle outliers in the data points. Our goal is to identify outliers in the variables. Outliers at special time points are very common in microarray experiments as technical problems like dust or a scratch on the slide can easily distort the data. In such a case these outlier variables can lead to unwanted correlations between genes and to incorrect assignment to clusters. There is a need for distance measures which are robust against outlier variables. The idea of Jackknife (Efron 1982) distance measures is not to exclude the whole observation for such a gene but rather one or several variables. We want to introduce so–called "Jackknife" distance measures which can handle one outlier time point. The *Jackknife correlation* was first used by Heyer et al. (1999) to cluster gene expression data. It is defined as

$$d_{xy} = 1 - \min(\rho_{xy}^{(1)}, \rho_{xy}^{(2)}, \ldots, \rho_{xy}^{(T)})$$

where $\rho_{xy}^{(t)}$ is the correlation of pair x,y computed with the tth time point deleted.

Now we want to extend this concept and introduce robust versions of the three geometric distance measures Euclidean, Manhattan and Maximum distance. Jackknife Euclidean distance is defined as

$$d_{xy} = \min(d_{xy}^{(1)}, d_{xy}^{(2)}, \ldots, d_{xy}^{(T)})$$

where $d_{xy}^{(t)}$ is the Euclidean distance of pair x,y computed with the tth time point deleted. Jackknife Manhattan distance and Jackknife Maximum distance can be defined in the same way. The impact of Jackknife distance measures is evaluated in Section 4.2.2.

2.1.4 Clustering of functional data

The standard application of K-Means is to assign data points to clusters based on minimal Euclidean distance to the cluster centers. However, observations over time are not just ordinary points in Euclidean space but curves with distinct shapes. Clustering functional data using the K–Means algorithm (Tarpey 2003, 2007) is very useful to determine representative curve shapes in a functional data set. This approach is frequently used in clustering microarray data (e.g., Abraham et al. (2003); de Hoon et al. (2002); Hakamada et al. (2006); Serban and Wasserman (2005)) where different methods are used to fit curves to the data.

In the simulation study a cubic spline using a B–spline basis is fit to each gene expression profile in order to account for time dependency. The estimated regression coefficients are then plugged into both the K–Means and the QT-Clust algorithm.

Additionally a K–Means–like algorithms is included in the study where splines are used for the computation of cluster centroids in order to account for the time–dependence of time course gene expression data.

Table 2.1: Overview of the partitioning methods used

algorithm	dist	cent	type
kmeans	eucl	mean	orig
kmeans	man	median	orig
kmeans	cor	optim	orig
kmeans	max	optim	orig
qtclust	eucl	mean	orig
qtclust	man	mean	orig
qtclust	cor	mean	orig
qtclust	max	mean	orig
kmeans	euc	spline	orig
kmeans	euc	mean	fd
qtclust	euc	mean	fd

2.1.5 Summary

The partitioning methods used in the simulation study on artificial data are summarized in Table 2.1. The two algorithms used are K–Means and QT–Clust, the four distance measures used are Euclidean, Manhattan, “1 - Correlation” and Maximum distance. For K–Means different types of centroid computation are used: cluster–wise means and cluster–wise medians are used for Euclidean distance and for Manhattan distance. A general purpose optimizer is used for “1 - Correlation” and Maximum distance. Additionally a new cluster algorithms is included with Euclidean distance and splines as centroid computation method. Finally clustering is performed either on the original data or on the functional data.

2.2 Model–based clustering

Finite mixtures of regression models are the state-of-the-art technique for modeling time course microarray data. The Expectation-Maximization (EM) algorithm (Dempster et al. 1977) is the most common method for maximum likelihood (ML) estimation despite its drawbacks such as convergence only to local optima in dependence of the initialization. Good starting values are therefore crucial for the EM algorithm to perform well. A common strategy is to use random initialization and to run the algorithm several times in order to overcome the convergence to local optima already determined by the initalization.

2.2.1 Model specification

The mixture density h of a finite mixture model with K components is given by

$$h(y|x, w, \psi) = \sum_{k=1}^{K} \pi_k(w) f(y|x, \theta_k).$$

y is the response, x the predictor, w the concomitant variables and ψ denotes the vector of all parameters for the mixture density h. For the component weights $\pi_k(w)$ it holds that $\pi_k(w) > 0$ for all k and $\sum_{k=1}^{K} \pi_k(w) = 1$. θ_k is the component-specific parameter vector for the density function f.

Recently, many model-based clustering approaches have been applied to microarray data. Mixtures of multivariate normal models are for example used by Fraley and Raftery (1998, 2002); Yeung et al. (2001); Ghosh and Chinnaiyan (2002) and mixtures of t distributions are used by McLachlan et al. (2002).

Mixtures of mixed-effects models are used to account for different kinds of heterogeneity between individuals. The components of the mixture correspond to different groups with distinct parameterizations while the random effects allow for individual differences which cluster around a common mean value.

The data of each individual i is given by (Y_i, X_i, Z_i, w_i) which consists of n_i observations on the dependent variables $Y_i = (y_{ij})_{j=1,\ldots,n_i}$, the covariates for the fixed effects $X_i = (x'_{ij})_{j=1,\ldots,n_i}$ and the covariates for the random effects $Z_i = (z'_{ij})_{j=1,\ldots,n_i}$. w_i denote the individual specific concomitant variables. The finite mixture density of mixed effects models with K components is given for the observations of individual i by

$$h(Y_i|X_i, Z_i, w_i, \Theta) =$$

$$= \sum_{k=1}^{K} \pi_k(w_i) \int \prod_{j=1}^{n_i} \phi_1(y_{ij}; x'_{ij}\beta_k + z'_{ij}b_i^k, \sigma_k^2) \phi_q(b_i^k; 0, \Psi_k) db_i^k$$

$$= \sum_{k=1}^{K} \pi_k(w_i) \phi_{n_i}(Y_i; X_i\beta_k, Z_i\Psi_k Z_i^T + \sigma_k^2 I_{n_i}).$$

$\phi_d(.; \mu, \Sigma)$ denotes the d-dimensional multivariate normal distribution with mean μ and variance-covariance matrix Σ.

- Fixed effects: $x'_{ij}\beta_k$ with β_k deterministic
- Random effects: $z'_{ij}b_i^k$ with $b_i^k \sim N(0, \Psi_k)$

Splines are frequently included in mixed-effects models to treat the gene expression level as a continuous function of time. They are used with B-splines (Luan and Li 2003; Bar-Joseph et al. 2003) and smoothing splines (Ma et al. 2006). For smoothing splines the degree of smoothness is chosen automatically by cross-validation.

In the simulation study (see Section 4.1.6) smoothing splines and B-splines are used to fit finite mixtures of linear regression models to time course gene expression data. The normal mixture model in combination with smoothing splines is compared to the mixed-effects model with a random intercept with B-splines using various initialization strategies.

2.2.2 Parameter estimation

The EM algorithm (Dempster et al. 1977) is the standard tool for ML estimation of finite mixture models. In the E-step the expectation of the complete likelihood is taken, i.e., the a-posteriori probabilities are computed. In the M-step the expected complete likelihood is maximized where the missing memberships are replaced by the a-posteriori probabilities. The likelihood is increased in each step and convergence of the algorithm is guaranteed for bounded likelihoods. Detection of the global optimum however cannot be ensured.

2.2.3 Initialization strategies

Up to our knowledge different initialization strategies have only been investigated in multivariate normal settings. In Scharl et al. (2009a) the performance of initialization strategies is investigated for mixtures of regression models with respect to time course microarray data. The goal of this study is to find good starting values for clusterwise regression using mixtures of linear models and mixtures of linear mixed models.

Biernacki et al. (2003) give an overview of simple initialization strategies including random initialization, classification EM (CEM) algorithms, stochastic EM (SEM) algorithms and preliminary short runs of EM itself. Their aim is to identify a simple method that gives the highest likelihood in a fixed number of iterations.

Wehrens et al. (2004) present an approach for large data sets where a random subsample is clustered prior to applying the model to the whole data set. A modification of this method is given in Fraley et al. (2005). They propose incremental model-based clustering for large data sets with small clusters and apply it to image data. The situation of large data sets with small clusters is also characteristic of microarray data.

- **True cluster membership:** For simulated data the true cluster memberships can be used for initialization in order to investigate the behaviour of the EM algorithm when starting in the optimal solution.

- **Random initialization:** A commonly used approach is to run EM x times with random starting values and to select the solution maximizing the likelihood among those x runs.

- **Classification EM algorithm:** CEM (Celeux and Govaert 1992) is a three step procedure where the E-step is equivalent to the standard algorithm. In the C-step a partition is designed by assigning each point to the component with the maximum a-posteriori probability. In the M-step the ML estimates are computed using the clusters of the C-step as sub-sample of the mixture components. CEM converges in a finite number of iterations and tends to produce a mixture with well separated components (Biernacki et al. 2003). It is not maximizing the observed log-likelihood but the complete likelihood.

  As an initialization strategy CEM is run from x random starting positions and the one providing the highest complete data log-likelihood is chosen as an initial solution for EM. CEM is started with K much larger than the actual number of clusters in the data as hard classification tends to omit too small clusters whereas the large ones dominate.

- **Stochastic EM algorithm:** SEM (Diebolt and Ip 1996) includes a restoration of the unknown component labels by drawing them at random from their current a-posteriori probabilities. The E-step is equivalent to the standard algorithm. In the S-step a partition is designed by assigning each point at random to one of the mixture components according to the multinomial distribution with parameter equal to the a-posteriori probabilities. In the M-step the ML estimates are computed using the clusters of the S-step as sub-sample of the mixture components. Random drawing at each iteration prevents the SEM from being trapped in local optima.

  For initialization SEM is run x times keeping the position leading to the highest maximum likelihood value. The stopping criterion for SEM is the maximum number of iterations which is set to 100.

- **Short runs of EM:** This procedure is suggested by Biernacki et al. (2003). EM is run x times from random starting positions before passing to EM without waiting for convergence using the threshold value $(L_q - L_{q-1})/(L_q - L_0) \leq tol$, where the tolerance ($tol$) is set to 10^{-2}. L_q is the log-likelihood at the qth iteration.

- **Sampling:** In Wehrens et al. (2004) the simple strategy to cluster larger data sets by clustering a small random sample of the data and to apply the resulting estimated model to the full data set is modified. The sampling methods starts with drawing x samples of size 100 from the full data set. Next, the EM algorithm is run 3 times on the x samples and the ML solution is used to initialize the EM algorithm for the full data set. Finally, the ML solution of the x models is selected.

- **Incremental Method:** Fraley et al. (2005) developed incremental model-based clustering which is an extension of the sampling method. The method starts at drawing a random sample of the data, selecting and fitting a clustering model to the sample that underestimates the number of components, and extending the model to the full data set by additional EM iterations. New clusters are then added incrementally, initialized with the observations that are poorly fit by the current model. The algorithm stops if adding further components does not increase the log-likelihood or if an a-priori fixed maximum number of components is reached.

  In the simulation study (see Section 4.1.6) the incremental method is started on x samples of size 100 with K equal to 6. As in the sampling method EM is started 3 times and the ML solution is applied to the full data set. New clusters are added incrementally and initialized with those observations with the lowest 5% log-likelihoods.

- **Spectral clustering:** A completely different approach is given by spectral clustering (e.g., Ng et al. (2001)) which does not make any assumptions on the form of the clusters. In spectral clustering data points are clustered using eigenvectors of matrices derived from the data. Spectral clustering is implemented in R package **kernlab** (Karatzoglou et al. 2004).

  In the simulation study the algorithm of Ng et al. (2001) is applied where the K eigenvectors are used simultaneously. The cluster solution from spectral clustering is used as starting value for the EM algorithm.

An overview of the investigated initialization strategies is given in Table 2.2. Throughout all computations the minimum component weight of clusters (Leisch 2004) is 0.005 and the maximum number of iterations is 5000 (except for SEM where it is 100). In Table

Table 2.2: Overview of the initialization strategies and parameters used where e.g. cem.em indicates the procedure of initializing the EM algorithm in the cluster solution of CEM providing the highest log-likelihood.

	Method	k	nrep
rep.em	Random initialization	16	10
true	True cluster membership	16	1
cem	Classification EM	30	10
cem.em	CEM.EM	result from cem	1
sem	Stochastic EM	16	10
sem.em	SEM.EM	result from sem	1
tol	Short runs of EM	16	10
tol.em	Short.EM	result from tol	1
sam	Sampling	16	$10 \cdot 3$
inc	Incremental Method	6	$10 \cdot 3$
sc	Spectral clustering	16	1
sc.em	SC.EM	result from sc	1

2.2 K is the number of clusters the algorithm starts with and *nrep* is the number of times the algorithm is started keeping only the solution with maximum likelihood. For the incremental method the number of starts is $10 \cdot 3$, i.e., 10 samples are drawn from the full data sets and the algorithm is started 3 times with random initialization on each of the samples.

2.3 Neighborhood graphs

The neighborhood graph (Leisch 2006) is a method for the visual assessment of centroid–based cluster solutions. It uses the idea of topology–representing networks (TRNs, Martinetz and Schulten (1994)) to count the number of data points a pair of centroids is closest and second–closest. In TRNs the counts are used as weights for the edges of the graph. Silhouette plots (Rousseeuw 1987) are diagnostic plots revealing the goodness of a partition. The distance from each point to the points in its own cluster is compared to the distance to points in the second closest cluster. The larger the silhouette values the better a cluster is separated from the other clusters. But silhouette plots do not show the proximity of clusters. They only give an indicator how well-separated single points are from other clusters. Neighborhood graphs combine these two approaches and use the mean relative distances as edge weights in order to measure how separated pairs of clusters are. Hence they display the distance between clusters. In the graph each node corresponds to a cluster centroid and two nodes are connected by an edge if there exists at least one point that has these two as closest and second–closest centroid.

For a given data set X_N and a set of centroids C_K the centroid closest to x is denoted by

$$c(x) = \arg\min_{c \in C_K} d(x, c).$$

The second closest centroid to x is denoted by

$$c_2(x) = \arg\min_{c \in C_K \setminus \{c(x)\}} d(x, c).$$

The set of all points where c_k is the closest centroid is given by

$$A_k = \{x_n | c(x_n) = c_k\}.$$

Now the set of all points where c_i is the closest centroid and c_j is second–closest is given by

$$A_{ij} = \{x_n | c(x_n) = c_i, c_2(x_n) = c_j\}.$$

For each observation x the shadow value $s(x)$ is defined as

$$s(x) = \frac{2d(x, c(x))}{d(x, c(x)) + d(x, c_2(x))}.$$

$s(x)$ is small if x is close to its cluster centroid and close to 1 if it is almost equidistant between the two cluster centroids. The average s–value of all points where cluster i is closest and cluster j is second closest can be used as a proximity measure between clusters and as edge weight in the graph.

$$s_{ij} = \begin{cases} |A_i|^{-1} \sum_{x \in A_{ij}} s(x), & A_{ij} \neq \emptyset \\ 0, & A_{ij} = \emptyset \end{cases}$$

$|A_i|$ is used in the denominator instead of $|A_{ij}|$ to make sure that a small set A_{ij} consisting only of badly clustered points with large shadow values does not induce large cluster similarity.

Observations that have similar distances to both centroids get a larger weight than observations which are close to one and far away from the other. Note that as a percentage the cluster similarity is always between 0 and 1. If we remove one of the two cluster centroids, all points contributing to the cluster similarity of the centroid pair would be re-assigned to the remaining centroid. Thus it can also be used as an indication which clusters are candidates for being merged.

Neighborhood graphs are generally applicable to various partitioning cluster algorithms like the well–known K–means or PAM. In order to use the neighborhood graph for the visualization of a cluster solution obtained from QT–Clust the corresponding cluster centroids are computed. Neighborhood graphs are a useful tool for the visualization of the structure of a cluster solution. Additionally they can be used as exploratory tool to determine the quality of a given clustering and to validate the number of clusters. Too fine partitions show very high connectivity in the graph, while too coarse clusterings are identified if different expression profiles show up in one cluster.

Chapter 3

Evaluation

For the evaluation of cluster solutions the Rand index is typically used as classification criterion. The likelihood criterion as well as the AIC or BIC can be used to compare cluster results from model–based clustering.

3.1 Rand index

For a given data set $X = \{x_1, \dots, x_n\}$ let $P_T = \{t_1, \dots, t_T\}$ and $P_C = \{p_1, \dots, p_C\}$ be two partitions with $\cup_{i=1}^{T} t_i = X = \cup_{j=1}^{C} p_j$ and $t_i \cap t_i' = \emptyset = p_j \cap p_j'$ for $1 \leq i \neq i' \leq T$ and $1 \leq j \neq j' \leq C$. Suppose P_T is the true underlying grouping and P_C some cluster result.

The agreement between the two partitions can be measured by the Rand index (Hubert and Arabie 1985) which is given by

$$\frac{A}{A+D}$$

where A is the number of all pairs of data points which are either in the same cluster in both partitions or in different clusters in both partitions. D is the number of all pairs of data points that are in the same cluster in one partition, but in different clusters in the other partition. Hence, D is the number of disagreements whereas A is the number of agreements between P_T and P_C.

Let n_{ij} be the number of objects that are in both class t_i and cluster p_j. Let $n_{i.}$ and $n_{.j}$ be the number of objects in class i and cluster j respectively. The Rand index corrected for agreement by chance (Hubert and Arabie 1985) is given by

$$R = \frac{\sum_{i,j} \binom{n_{ij}}{2} - [\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}]/\binom{n}{2}}{0.5 \cdot [\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}] - [\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}]/\binom{n}{2}}.$$

An adjusted Rand index of 1 corresponds to identical partitions and a Rand index of 0 corresponds to agreement by chance given cluster size.

One of the benefits of the QT–Clust algorithms is that outliers are not forced into clusters. However, the adjusted Rand index cannot deal with missing cluster memberships. Hence objects which are not assigned to a cluster in either partition P_T or P_C are omitted from the calculation. Thalamuthu et al. (2006) suggest a modified version of the adjusted Rand index where outliers form an additional cluster, i.e., the numbers of clusters are $T+1$ and $C+1$ and the number of outliers are $|t_{R+1}|$ and $|p_{C+1}|$ respectively. The adjusted Rand index ignoring missing values in a cluster solution is

$$R_1 = \frac{\sum_{i,j}\binom{n_{ij}}{2} - [\sum_i \binom{\tilde{n}_{i.}}{2}\sum_j \binom{\tilde{n}_{.j}}{2}]/\binom{\tilde{n}}{2}}{0.5\cdot[\sum_i \binom{\tilde{n}_{i.}}{2} + \sum_j \binom{\tilde{n}_{.j}}{2}] - [\sum_i \binom{\tilde{n}_{i.}}{2}\sum_j \binom{\tilde{n}_{.j}}{2}]/\binom{\tilde{n}}{2}}$$

where $\tilde{n}_{i.} = n_{i.} - n_{i(C+1)}$ and $\tilde{n}_{.j} = n_{.j} - n_{(T+1)j}$. Additionally, Thalamuthu et al. (2006) propose to include the clusters of outliers in the calculation of the Rand index, i.e., $R_2 = R$ for $i = 1,\dots,T+1$ and $j = 1,\dots,C+1$.

In order to give less weight to outliers they propose to form a weighted Rand index

$$R* = \lambda \cdot R_1 + (1-\lambda)\cdot R_2$$

where $\lambda = |t_{R+1} \cup p_{C+1}|/n$.

A further possibility is to allow clusters of size 1 and to put each outlier into a single clusters, i.e. the number of clusters in partitions P_T and P_C is $T + |t_{R+1}|$ and $C + |p_{C+1}|$ respectively. The corresponding Rand index is $R_3 = R$ for $i = 1,\dots,T,T+1,\dots,T+|t_{R+1}|$ and $j = 1,\dots,C,C+1,\dots,C+|p_{C+1}|$.

The performance of the different versions of the modified Rand index is evaluated on artificial data in Section 4.1.2.

3.2 Sum of within cluster distances

The sum of within cluster distances W is an indicator for the tighness of a given clustering. It is given by

$$W = \sum_{j=1}^{K}\sum_{x\in X_j} d(x, c_j)$$

where x is an element of cluster X_j, d is the distance measure and c_j is the centroid of cluster j.

3.3 Inferential analysis

3.3.1 Functional relevance test

The obtained similarity between clusters and the neighborhood graph can be used to evaluate a cluster result at hand. The cluster structure can be used to decide whether the clustering is too coarse and needs further subdivision to respect the data or if it is too fine and some clusters should be merged. On the one hand this can be accomplished by defining some threshold t for the shadow value s above which two clusters are merged. In the case of too large clusters more accurate clusters can for instance be obtained by running the algorithm again with larger K.

On the other hand external knowledge about the data can be used to validate a given clustering. In the case of microarray data a priori information about gene function or the association to functional groups can be used as functionally related genes are more likely to be co–expressed. Clusters with similar expression pattern are connected in the neighborhood graph. If functional group F is independent of the experimental setup genes classified to group F will be assigned to arbitrary clusters, i.e., they are assumed to be spread all over the neighborhood graph. Further, genes functionally independent of the experimental setup do not have a common expression pattern. If functional group F plays a role in the experiment the corresponding genes are more likely to show a typical pattern of either up– or down–regulation and there should be clusters with accumulation of such genes.

Assigning all genes in the clustered data set to some functional group F yields proportions $\pi_1, \ldots, \pi_K$ where K is the number of clusters or nodes and N_F is the total number of genes in the data set assigned to group F. If there is no association between the functional group and the cluster solution then all proportions are the same, i.e., the differences between proportions $d_{ij} = 0$ where

$$d_{ij} = |\pi_i - \pi_j|, \qquad i, j = 1, \ldots, K.$$

If there is an association then some π_k will be large and others small. The test for functional relevance of a given clustering is conducted in a stepwise way.

Step 1: Perform a global test of the equality of proportions, i.e., test the null hypothesis that all proportions π_k^F are the same

$$H_0 : d_{ij} = 0 \qquad \forall i, j = 1, \ldots, K.$$

The test procedure stops if there is no difference in proportions. But if there are significant differences in proportions each single difference has to be investigated in more detail. If the proportion of functionally related genes is the same in two clusters these two clusters are similar with respect to functional group F and can therefore be merged. This procedure yields separated subgraphs with common gene function within the neighborhood graph.

Without knowledge about the cluster structure and the similarities between clusters given in the neighborhood graph G each pair of clusters has to be tested for a significant difference in proportions, i.e., $K(K-1)/2$ tests have to be conducted. Using the neighborhood structure only a fraction of all possible pairs, i.e., clusters connected by an edge have to be tested. A further reduction of tests can be achieved by taking into account only nodes where the number of functionally assigned genes is above a threshold m.

Step 2: Assess the significance of the observed differences with respect to a reference distribution by permuting the function labels. The null hypothesis is again no difference in proportions.

- Select all clusters where the number of functionally assigned genes is above the predefined threshold m and conduct all further calculations on the resulting subgraph G'.

- Calculate the difference between proportions d_{ij}, $i, j = 1, \ldots, K$ for each edge in the subgraph.

- Permute the function labels, i.e., randomly assign N'_F genes to functional group F, where N'_F is the number of assigned genes in the subgraph G' with $N'_F \leq N_F$. Compute the resulting differences in proportions d^l_{ij}, $i, j = 1, \ldots, K$ and keep the respective maximum
$$M^l = max_{i,j} d^l_{ij}$$
as used in Zeileis et al. (2007) to form a reference distribution $\{M^l\}_{l=1}^L$ where L is the number of permutations considered.

- Compute marginal tests whether a particular d_{ij} is extreme relative to the joint distribution M^l, i.e., compute how often the maximum of the permuted differences in proportions is larger than the observed one.

In other words, if the observed difference in proportions is very unlikely with respect to the reference distribution of the maxima M^l the edge will be removed. In this procedure a modified neighborhood graph is formed for the cluster solution and functional group under investigation. In this modified graph two clusters are only connected if they have

1. a large similarity value s and
2. no significant difference in proportions of functionally related genes.

3.3.2 Compare cluster results

Validation of microarray cluster results is a challenging task (e.g., Androulakis et al. (2007)) as there is in general no true cluster membership. The quality of a cluster solution should be judged based on its ability to provide insight into the underlying mechanistic biology. As described in the previous section the validity of a cluster solution can be judged based on its ability to find groups of functionally related genes. Another approach is to find genes with common mechanism of regulation by searching for groups of genes that show a common response in different experiments.

For that purpose another test procedure was developed. We test how valid a given cluster solution is on a different data set taking into account the average within cluster distance $W = (w_1, \ldots, w_K)$ where

$$w_k = \frac{1}{|A_k|} \sum_{n \in A_k} d(x_n, c_k).$$

Let X_N be the data matrix of N genes for a given experiment and let M be the vector of length N of the corresponding cluster memberships. Further let Y_N be the data matrix of the same N genes in a different experiment. In order to test if the cluster memberships M found for data set X_N are also valid in data set Y_N the following procedure is used.

1. Compute the new cluster centroids $\tilde{C}_K$ for data set Y_N using the vector of cluster memberships M.
2. For each cluster k compute the average within cluster distance of data points y_n to their assigned centroid $\tilde{c}_k$, i.e.,

$$\tilde{w}_k = \frac{1}{|A_k|} \sum_{n \in A_k} d(y_n, \tilde{c}_k).$$

3. Permute the cluster memberships, i.e., randomly assign the genes to clusters but do not modify cluster sizes. Compute the resulting average within cluster distance $\tilde{w}_k^l$ for each cluster and keep the $\tilde{W}_k = (\tilde{w}_k^1, \ldots, \tilde{w}_k^L)$ where L is the number of permutations considered.

4. Compute marginal tests for each cluster of whether a particular $\tilde{w}_k$ is extreme relative to the joint distribution of $\tilde{W}_k$.

For each k where $k = 1, \ldots, K$ a single test is performed with the null hypothesis

$$H_0 : \tilde{w}_k = \tilde{w}_k^l \qquad \forall l = 1, \ldots, L$$

and the alternative hypothesis is

$$H_1 : \tilde{w}_k < \tilde{w}_k^l.$$

The null hypothesis is rejected if the propability of observing a smaller within cluster distance by randomly assigning genes to clusters is less than e.g. 5%. In this case there is a relationship between the investigated cluster solution on the original data set and on the new data set and genes with common expression pattern across experiments are found.

Chapter 4

Simulations

4.1 Artificial data

The performance of the different cluster methods is now evaluated on artificial data sets which are designed to resemble time course gene expression patterns. The number of clusters is 15 (as used in Thalamuthu et al. (2006)) plus an additional noise cluster of genes. The number of time points is 16 (equal to the number of time points in the *E. coli* data set used in Section 6.2). This is a common length for time series microarray data (see for example Cho et al. (1998)). The cluster sizes vary between 10 and 100 yielding a total of 630 genes with defined cluster pattern.

Typical time course microarray data have the following form

$$y_{ij} = b_i + \epsilon_{ij}$$

where $b_i \sim N(\mu_k, \sigma_b^2)$ and $\epsilon_{ij} \sim N(0, \sigma_\epsilon^2)$.

The expression pattern y of each gene i at time point j in a given cluster k is assumed to follow the shape of the cluster center μ_k but with a gene specific shift b_i (specified by the noise parameter "SD of RI" (σ_b) where SD denotes standard deviation and RI is the Random Intercept). Additionally a normally distributed measurement error ϵ_{ij} (specified by the noise parameter "SD of mean of genes" (σ_ϵ)) is added to each observation (time point) j.

As typical gene clusters do have arbitrary cluster sizes all simulated data sets consist of clusters of sizes between 10 and 100. Finally an additional noise set of genes of specified size N (given by the noise parameter "number of noise genes") is added to the data. For each noise gene $\mu_k \sim N(0, \sigma_m^2)$ and $\sigma_b \sim U(0.1, 0.3)$. σ_m is specified by the noise parameter

Table 4.1: Overview of the varying noise parameters.

	Noise level	low	medium	high
N	number of noise genes	100	500	1000
σ_ϵ	SD of mean of genes	0.1	0.3	0.5
σ_m	SD of mean of noise genes	0	1	2
σ_b	SD of RI	0.1	0.7	1.5

"SD of mean of noise genes".

An overview of the different noise parameters used is given in Table 4.1. One set of cluster centers is used to generate 81 data sets using all possible combinations of noise parameters.

The framework of this simulation study is the following:

1. generate 100 sets of centroids using integrated AR processes (see Section 4.1.1),
2. add the 81 combinations of noise presented in Table 4.1 to the centroids,
3. perform all presented cluster methods on these data sets and
4. evaluate the performance of the different methods ignoring the noise cluster.

4.1.1 Integrated AR processes for simulated data

In this simulation setup cluster centers are created using integrated autoregressive models. These have been used before to describe gene expression time series (e.g., Ramoni et al. (2002)) as autoregressive processes resemble the shape of gene expression over time observed in real time course data very well. An autoregressive process A_j of order 1 is defined by

$$A_j = \alpha A_{j-1} + \epsilon_j$$

where ϵ_j is a series of uncorrelated random variables with mean 0 and variance σ^2. It describes how each observation is a function of the previous observation.

An integrated AR(1) process is a process whose dth difference is an AR(1) process. If $d = 0$ the observations are modeled directly, if $d = 1$ the differences between consecutive observations are modeled, i.e.,

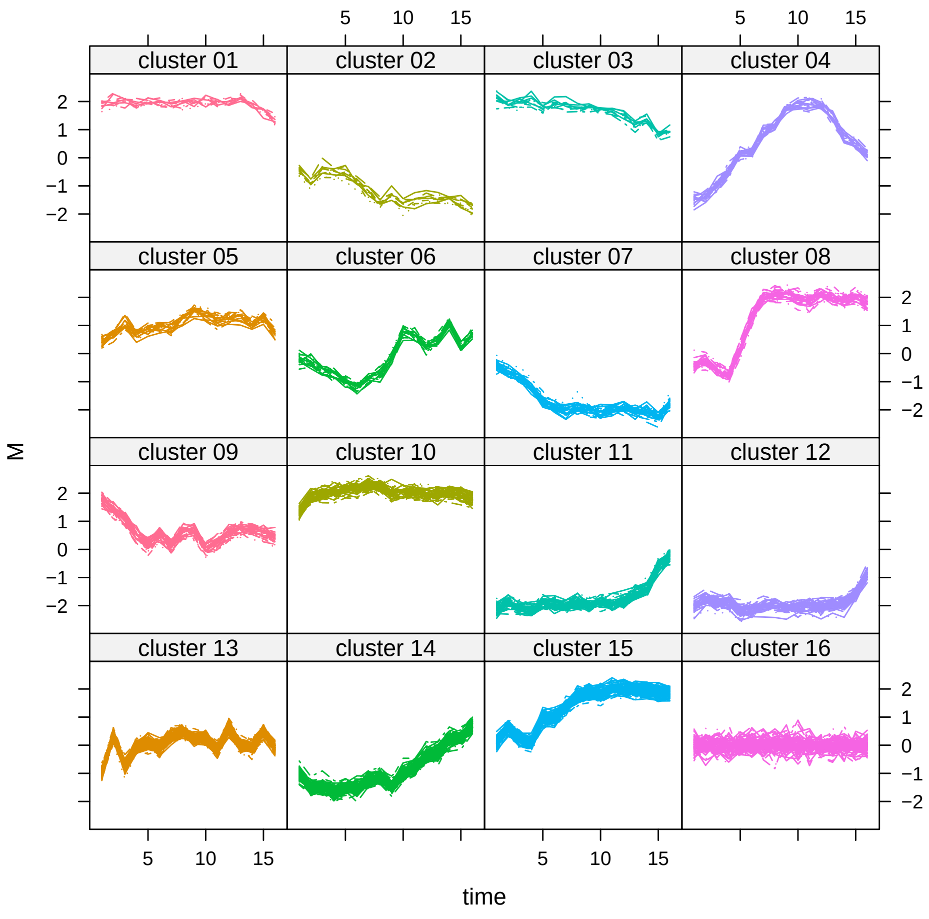


Figure 4.1: Artificial data set with low noise level where integrated AR processes are used to create cluster centers.

$$A_j = A_{j-1} + \alpha(A_{j-1} - A_{j-2}) + \epsilon_j.$$

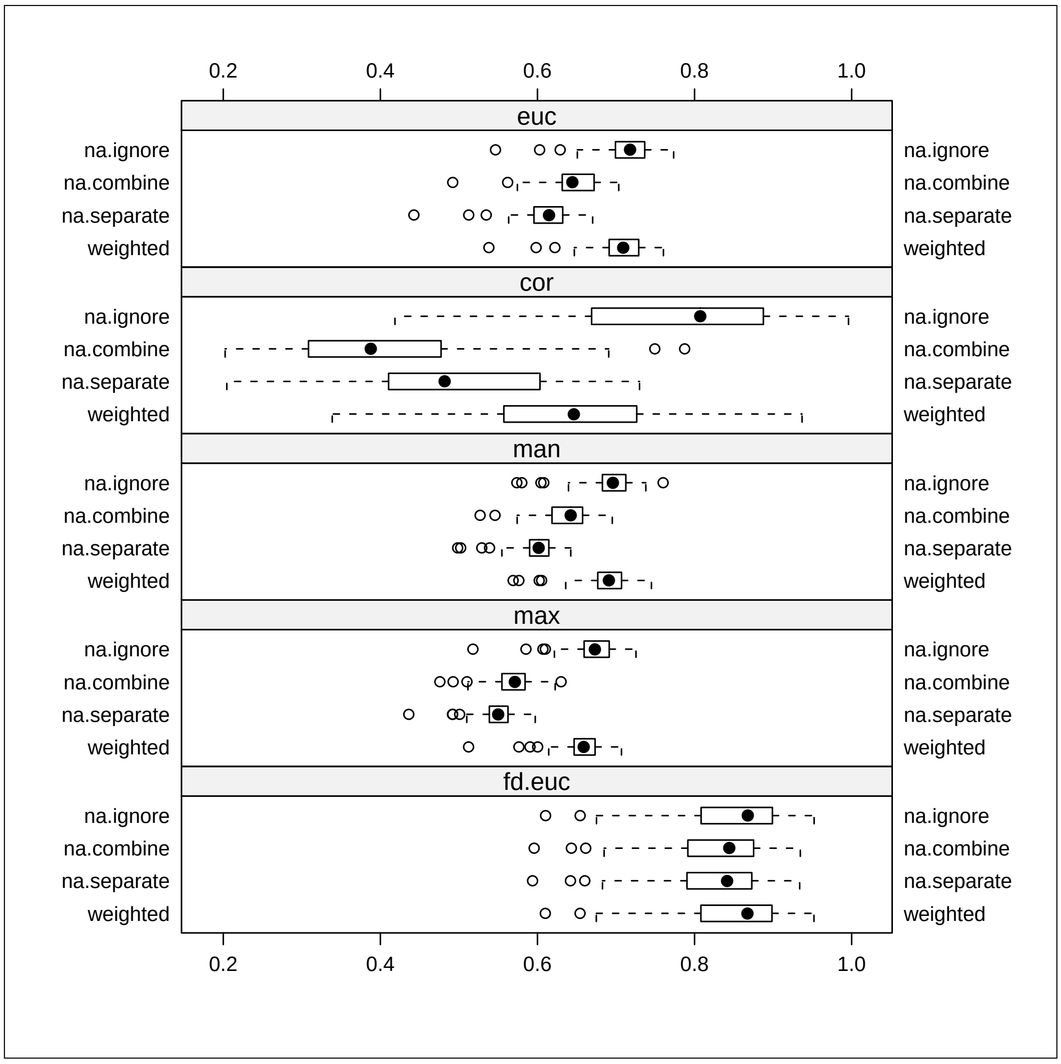
If $d = 2$ the differences of differences are modeled, etc.

In this study parameter d is either 1 or 2 in order to get different degrees of smoothness. Half of the generated time series are then reversed and finally transformed to the range of typical gene expression profiles.

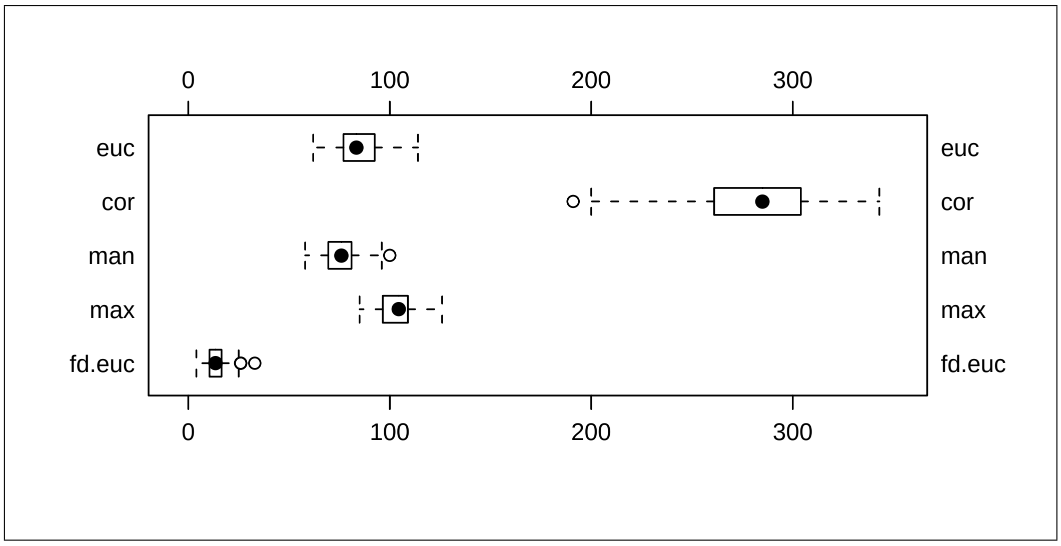
One set of cluster centers consists of 15 expression patterns yielding data sets of 15 clusters with dimension (number of time points) 16. An artificial data set where low noise is added to the cluster centers is given in Figure 4.1 where cluster 16 is a noise cluster of genes showing no differential expression.

4.1.2 Modifications of the adjusted Rand index

In this section the impact of outliers on the agreement between the true cluster membership and the cluster solutions is investigated using the modified versions of the adjusted Rand index presented in Section 3.1. Figure 4.2 shows the four different version of the adjusted Rand index (panel (a)) and the number of outliers (panel(b)) on 100 data sets with low noise level for the five investigated QT–Clust methods, i.e., QT–Clust on the original data using Euclidean, "1 - Correlation", Manhattan and Maximum distance and QT–Clust on the functional data using Euclidean distance. It can be seen from panel (a) that ignoring the missing values (outliers) in the cluster solutions yields the highest Rand index whereas combining the outliers to a single cluster and treating each outlier as a separate cluster yield smaller Rand indices. The weighted Rand index is very similar to ignoring the outliers except for "1-Correlation" distance. This is due to the fact that the number of outliers is much larger for "1-Correlation" distance (see panel (b)). When the number of outliers is very small (e.g., QT–Clust on functional data) the impact of the different versions of the Rand index is very low. In the following the adjusted Rand index ignoring outliers is used.



(a) Rand index



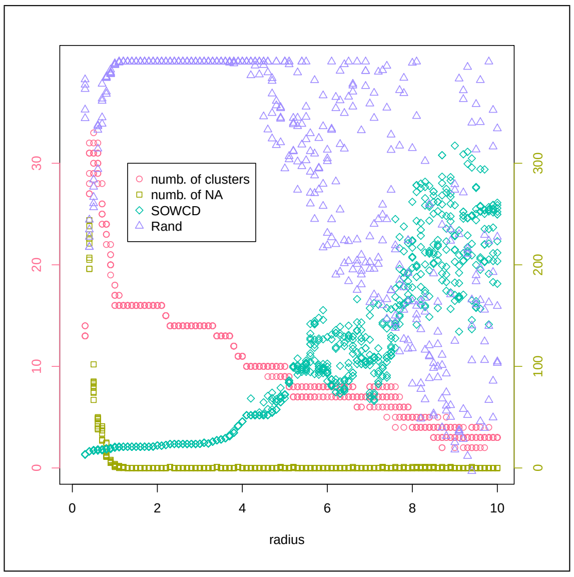
(b) number of outliers

Figure 4.2: Different versions of the adjusted Rand index when outliers are identified by the cluster algorithm.

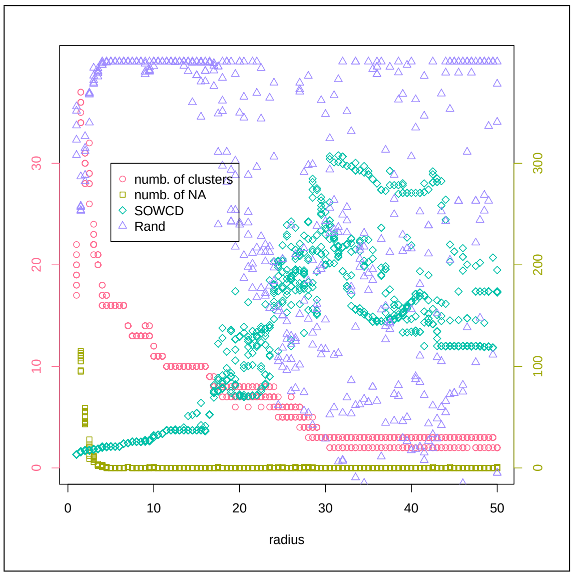
4.1.3 The choice of a radius for QT-Clust

In contrast to K–Means it is not possible for QT–Clust to specify the number of clusters when calling the algorithm. The number of clusters for QT–Clust is controlled indirectly by the choice of the quality threshold (radius). Data points that exceed the threshold are not assigned to any cluster and treated as outliers. A data set with low noise level is now used to investigate the impact of the radius for QT–Clust on the number of clusters, the number of outliers, the tightness of clusters (given by the sum of within cluster distances (SOWCD) and the adjusted Rand index between partitions generated with the same radius. In Figure 4.3 comparative plots are shown for Euclidean, Manhattan, "1-Correlation" and Maximum distance. The left and right axis of the single plots correspond to the number of clusters and number of outliers respectively. The Rand index is rescaled that 40 on the left axis corresponds to a Rand index of 1. The SOWCD is rescaled that the maximum is approximately 40.

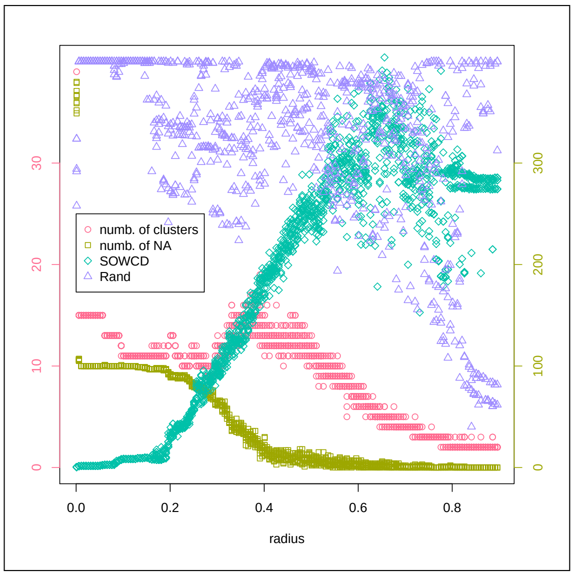
For all four distance measures the number of outliers explodes if the radius is set too small. In this case only a few objects are assigned to clusters but the majority are outliers. On the other hand the SOWCD increases very fast if the radius is too large. The adjusted Rand index between repeated calls of QT–Clust is close to 1 in the region where the SOWCD is small and decreases very fast as soon as the SOWCD increases. The optimal radius is in the region where the number of clusters and the SOWCD cross and the Rand index starts to decrease. For the geometric distance measures the number of outliers is close to zero in this region but for "1-Correlation" (panel (c) of Figure 4.3) the number of outliers is still very high. Additionally the region of high agreement between repeated calls of QT–Clust is very small for "1-Correlation". This indicates that "1-Correlation" distance is much more sensitive to the choice of the radius than the remaining distance measures. A general suggestion for a suitable radius is not possible. The radius has to be tuned separately for each data set and each distance measure used.



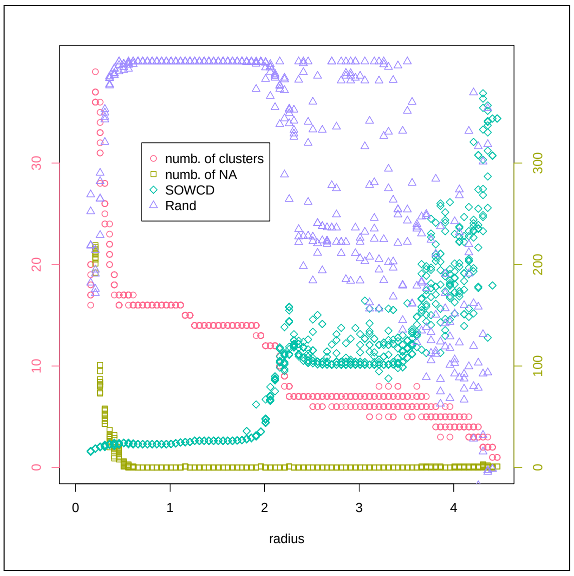
(a) Euclidean



(b) Manhattan



(c) 1-Correlation



(d) Maximum

Figure 4.3: The impact of the radius chosen for QT–Clust on the number of clusters, the number of outliers, the sum of within cluster distances (SOWCD) and the adjusted Rand index between partitions of repeated calls of QT–Clust.

4.1.4 Comparison of distance measures for K–Means and QT-Clust

In Figures 4.4 and 4.5 the agreement between the true cluster membership and different cluster solutions of QT–Clust and K–Means is given for the four investigated distance measures. In Figure 4.4 the cluster methods are investigated when low, medium and high noise is present in the data. In all three cases the performance of K–Means is better than the performance of QT–Clust and “1–Correlation” distance outperforms the other distance measures. For a medium and high noise level only “1-correlation” distance can be recommended whereas the other distance measures perform very poorly.

In Figure 4.5 the goodness of partitioning cluster methods is investigated when only one type of noise is present in the data. In the top left panel of Figure 4.5 the performance of the different cluster methods is displayed when the genes deviate very much from their cluster centroid (“large SD of mean of genes”). In this case Euclidean distance and Manhattan distance as well as centroid computation using splines for K–Means perform best whereas the worst results are found for “1–Correlation” distance. In the case of a large gene specific shift (“large SD of random intercept”) the picture is very different and “1–Correlation” distance clearly outperforms all other methods. It can be seen from the bottom left and right panel that noise genes do not affect the clustering of the relevant genes much.

4.1.5 Comparison of clustering functional data vs. original data

Next the impact of clustering functional data is investigated for Euclidean distance. In Figure 4.6 K–Means and QT–Clust cluster solutions on the functional and original data are investigated when low, medium and high noise is present in the data. For low noise level QT–Clust on functional data performs best. For medium and high noise level the best cluster method is K–Means clustering on functional data. In all three cases clustering functional data outperforms clustering the original data.

In Figure 4.7 the four different methods are investigated when only one type of noise is present in the data. For a large SD of the mean of genes clustering the original data is clearly the method of choice and clustering the functional data wrongly groups the gene expression profiles to clusters. For a large SD of the random intercept the results are very different as expected. In this case the expression profiles are very similar but with a gene specific shift and clustering the functional data is much better suited in this situation. However, the adjusted Rand index is smaller than 0.80 for all methods. Again the impact

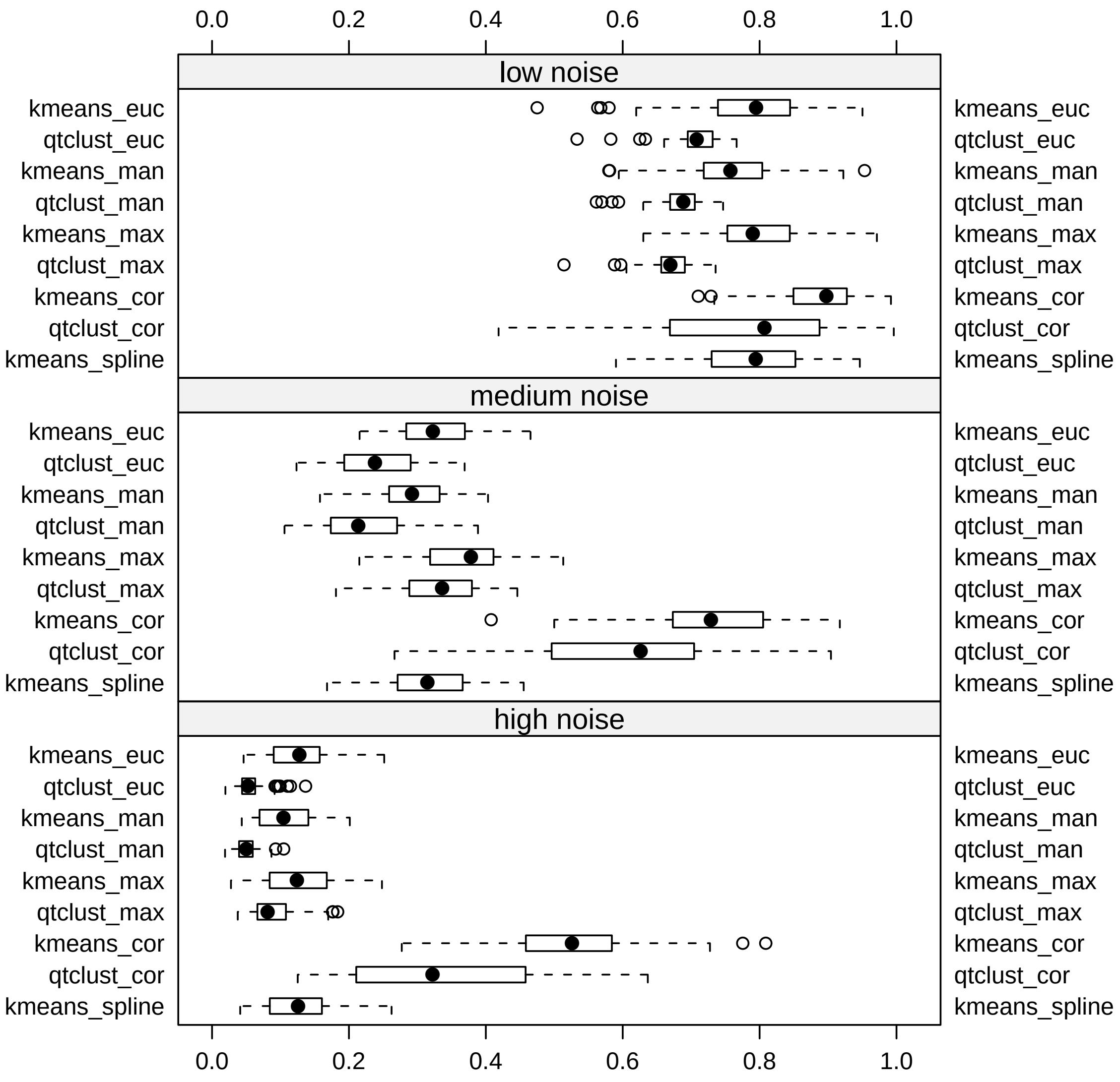


Figure 4.4: Comparison of distance measures for K–Means and QT-Clust using the adjusted Rand index on 100 data sets with low, medium and high noise level added to the cluster centers.

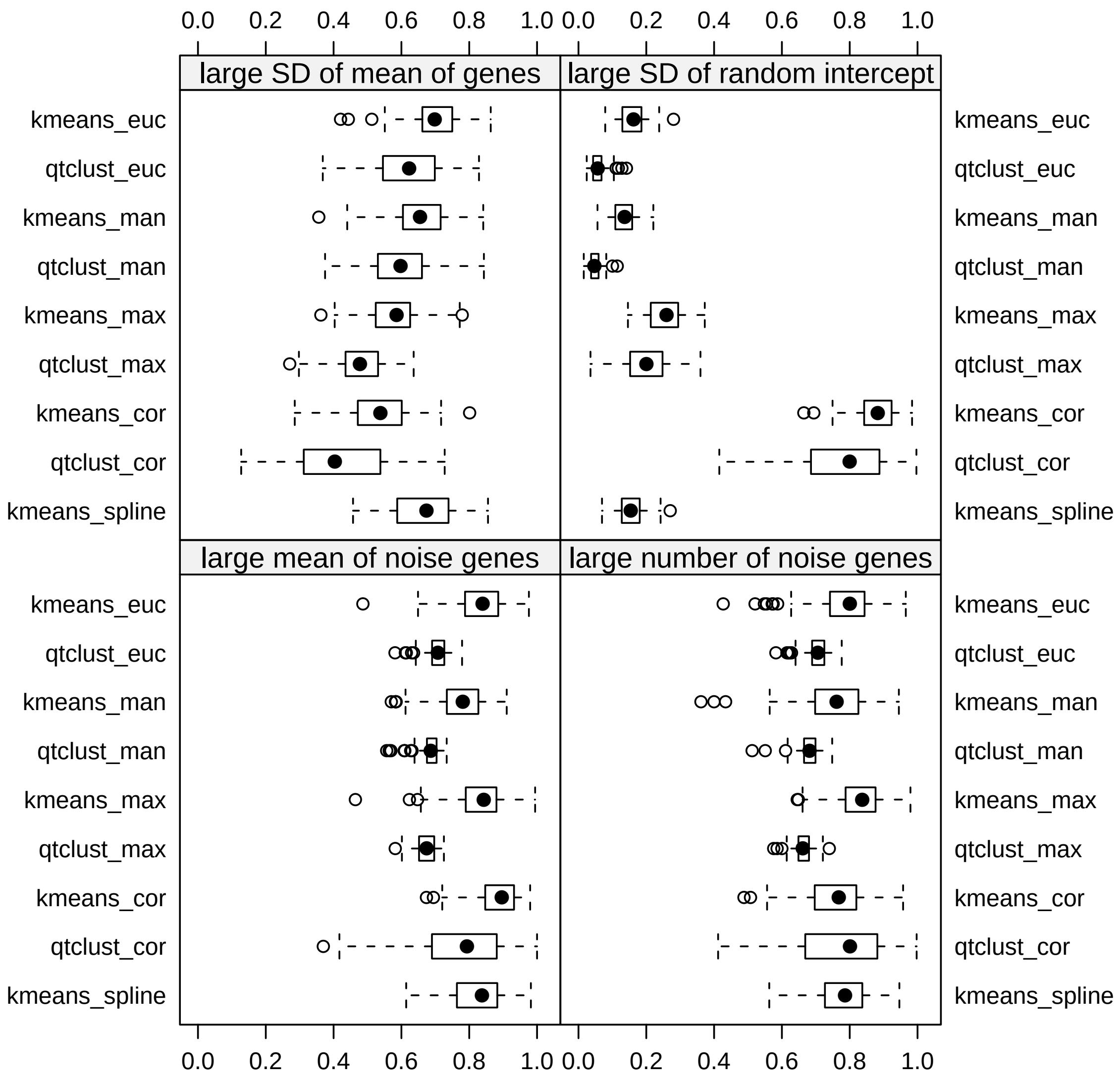


Figure 4.5: Comparison of distance measures for K–Means and QT-Clust using the adjusted Rand index on 100 data sets when only one type of noise is present in the data.

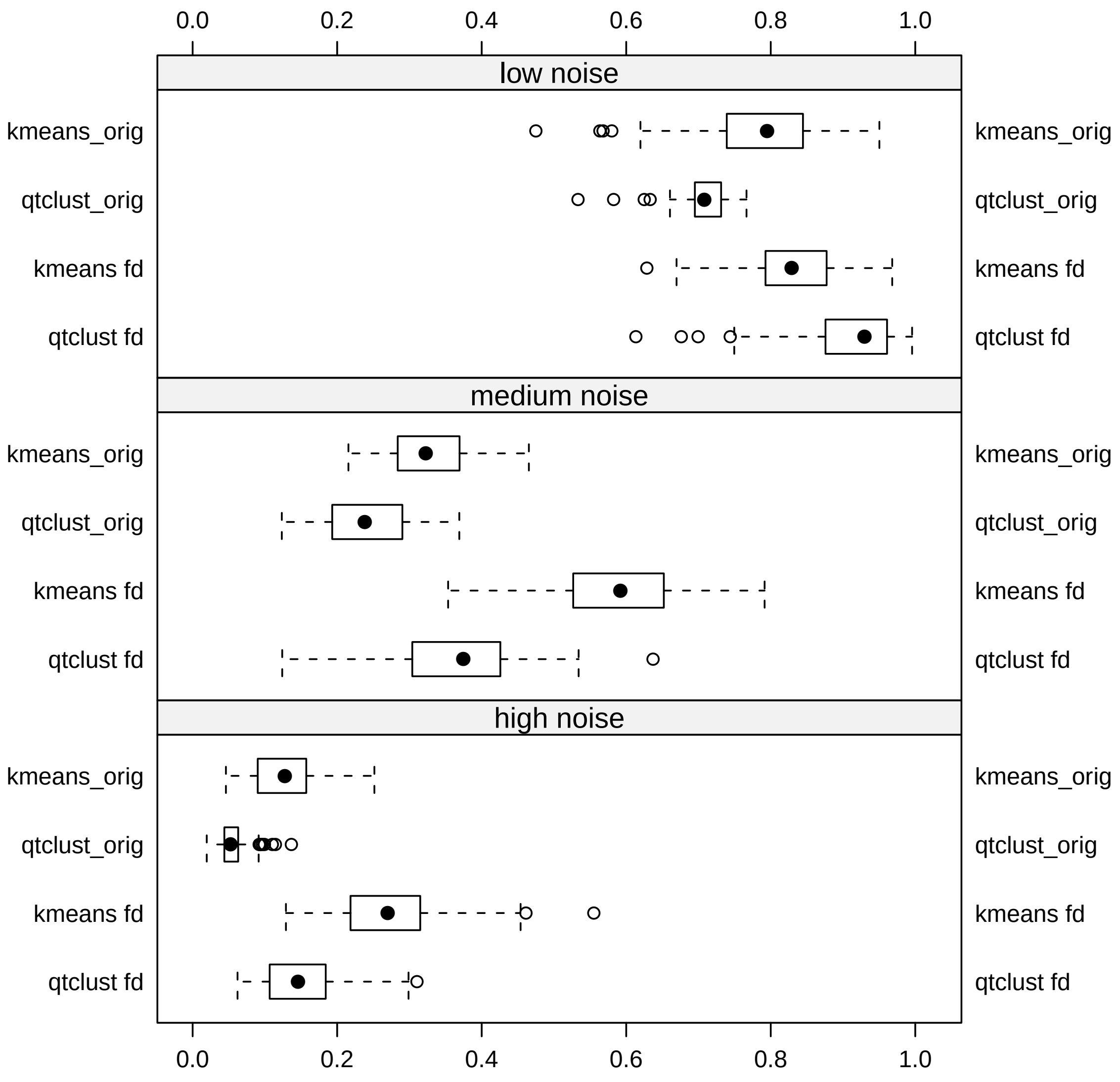


Figure 4.6: Comparison of clustering functional data vs. original data using the adjusted Rand index on 100 data sets with low, medium and high noise level.

of adding noise genes to the data sets is very small and all four methods perform very well (bottom left and right panel of Figure 4.7). If the noise genes are spread over the whole expression range (large mean of noise genes) QT–Clust on functional data performs best and if a large number of noise genes is present in the data K–Means slightly outperforms QT–Clust on functional data.

4.1.6 Model-based clustering: Evaluation of initialization and random effects

Mixtures of linear models

In this section the cluster results of the mixtures of linear models (mixtures of LMs) without RI are summarized for the different initialization strategies presented in Table 2.2. Figure 4.8 shows the adjusted Rand index of cluster solutions of the different initialization strategies and the true cluster membership when low, medium and high noise level is present in the data. For a low noise level starting in the true cluster solution yields the best results as expected. In this case sampling and spectral clustering are also good initialization strategies whereas CEM performs worst. For medium or high noise level the performance of all models is not good and even starting in the true solution yields adjusted Rand indices smaller than 0.5. For these noisy data sets spectral clustering outperforms model-based clustering.

In Figure 4.9 the adjusted Rand index is used to compare the performance of the different methods when only one type of noise is present in the data. The corresponding log-likelihoods and runtimes are displayed in Figures 4.10 and 4.11. Figure 4.10 shows that hardly any increase in log-likelihood is observed when starting EM in the solution of CEM, SEM or short runs of EM. Runtimes are only shown in Figure 4.11 for the three noise scenarios where the number of genes is equal, i.e., large SD of mean of genes, large SD of RI and large mean of noise genes. As the number of noise genes added to a data set is much larger in the forth noise scenario (yielding a total of 1630 genes) the longer runtimes cannot directly be compared to the other scenarios where the number of genes is always 730. Therefore the resulting runtimes are displayed in a separate plot (see Figure 4.12).

For a large SD of the mean of genes the true cluster solution is again the best starting partition, followed by SEM and sampling (see Figure 4.9). The overall performance is good. However, the runtimes of SEM and the incremental method are the longest, followed by sampling and random initialization.

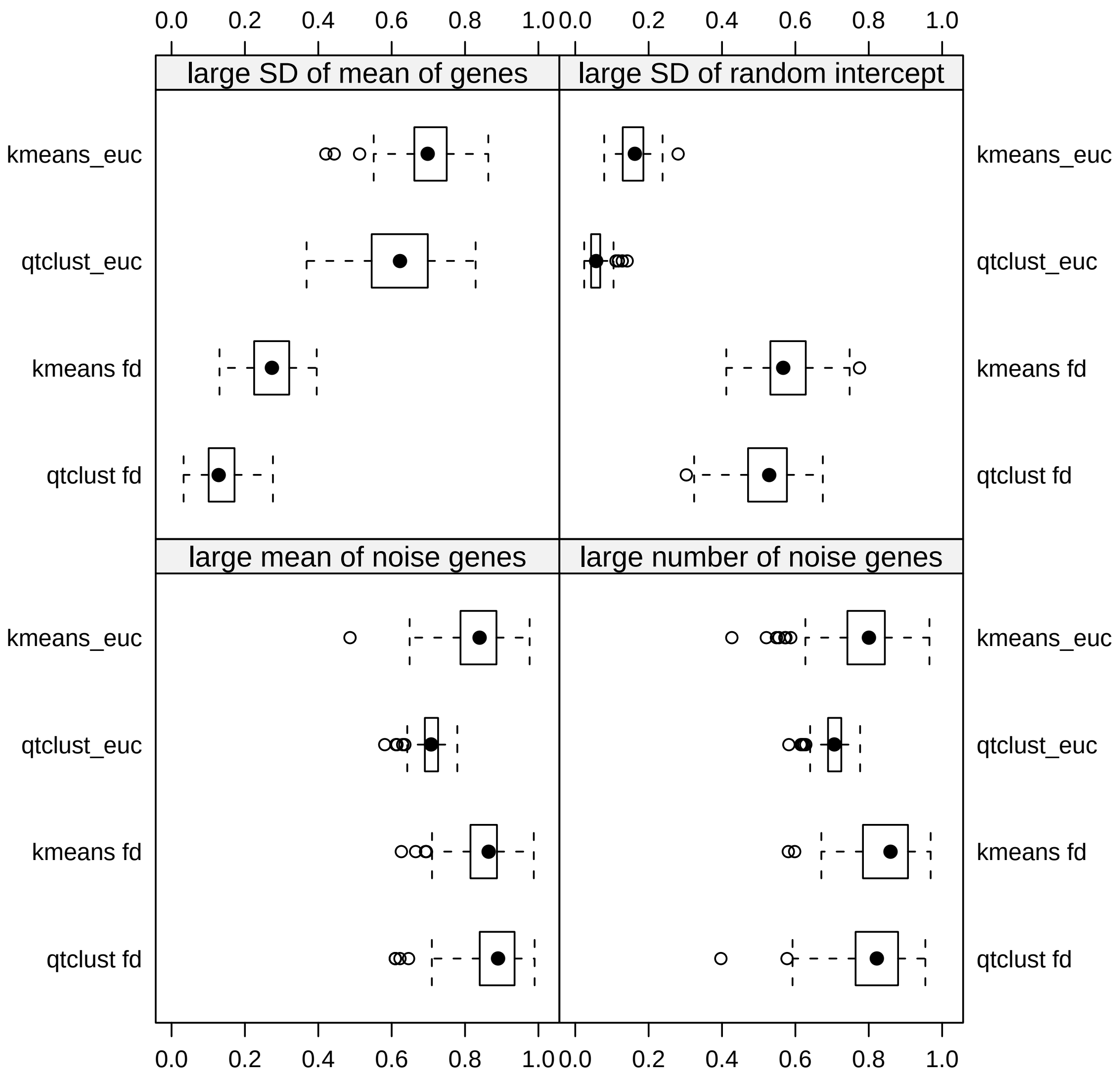


Figure 4.7: Comparison of clustering functional data vs. original data using the adjusted Rand index on 100 data sets when only one type of noise is present in the data.

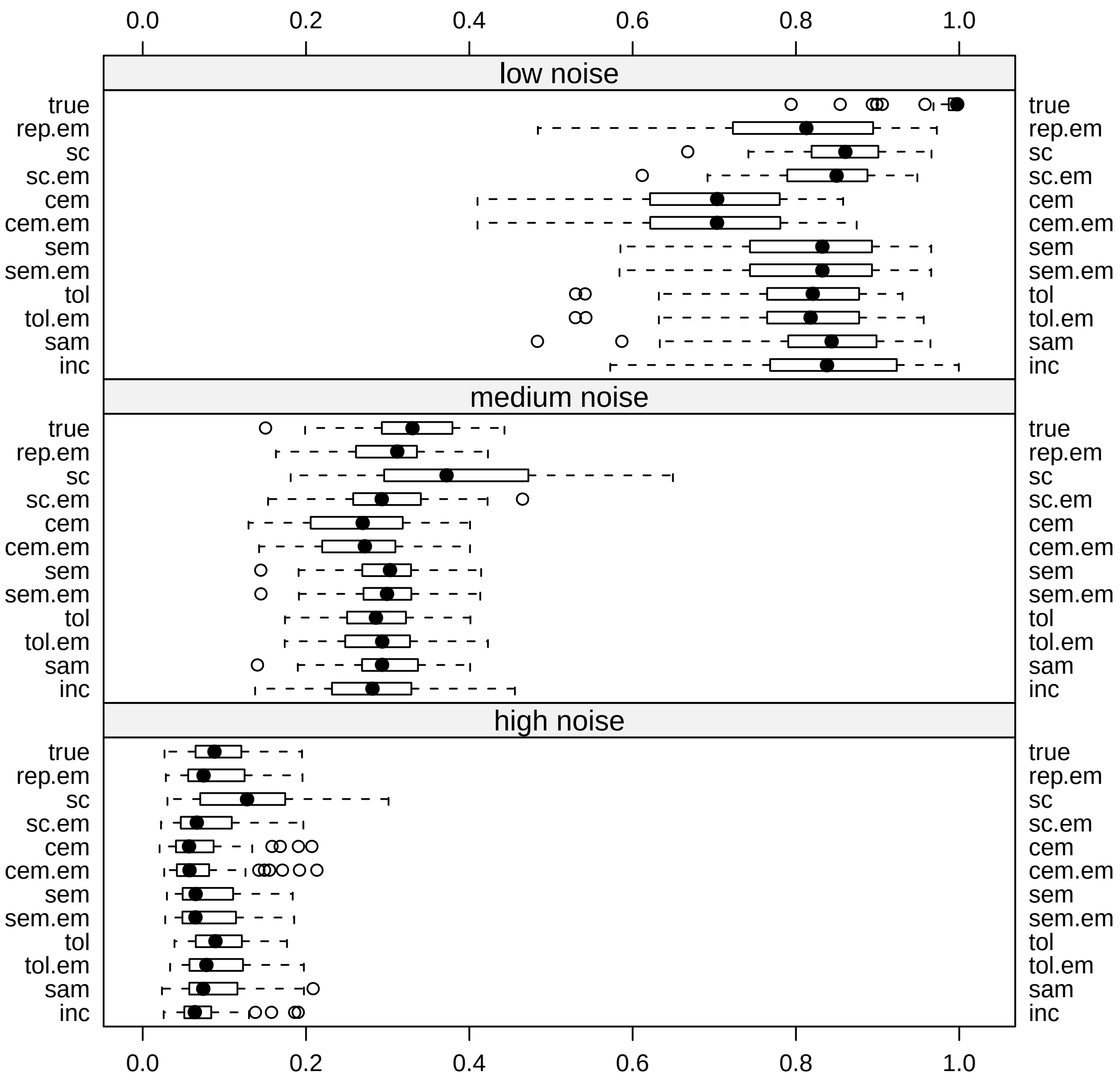


Figure 4.8: Adjusted Rand Index of the different initialization strategies for mixtures of LMs for low, medium and high noise level.

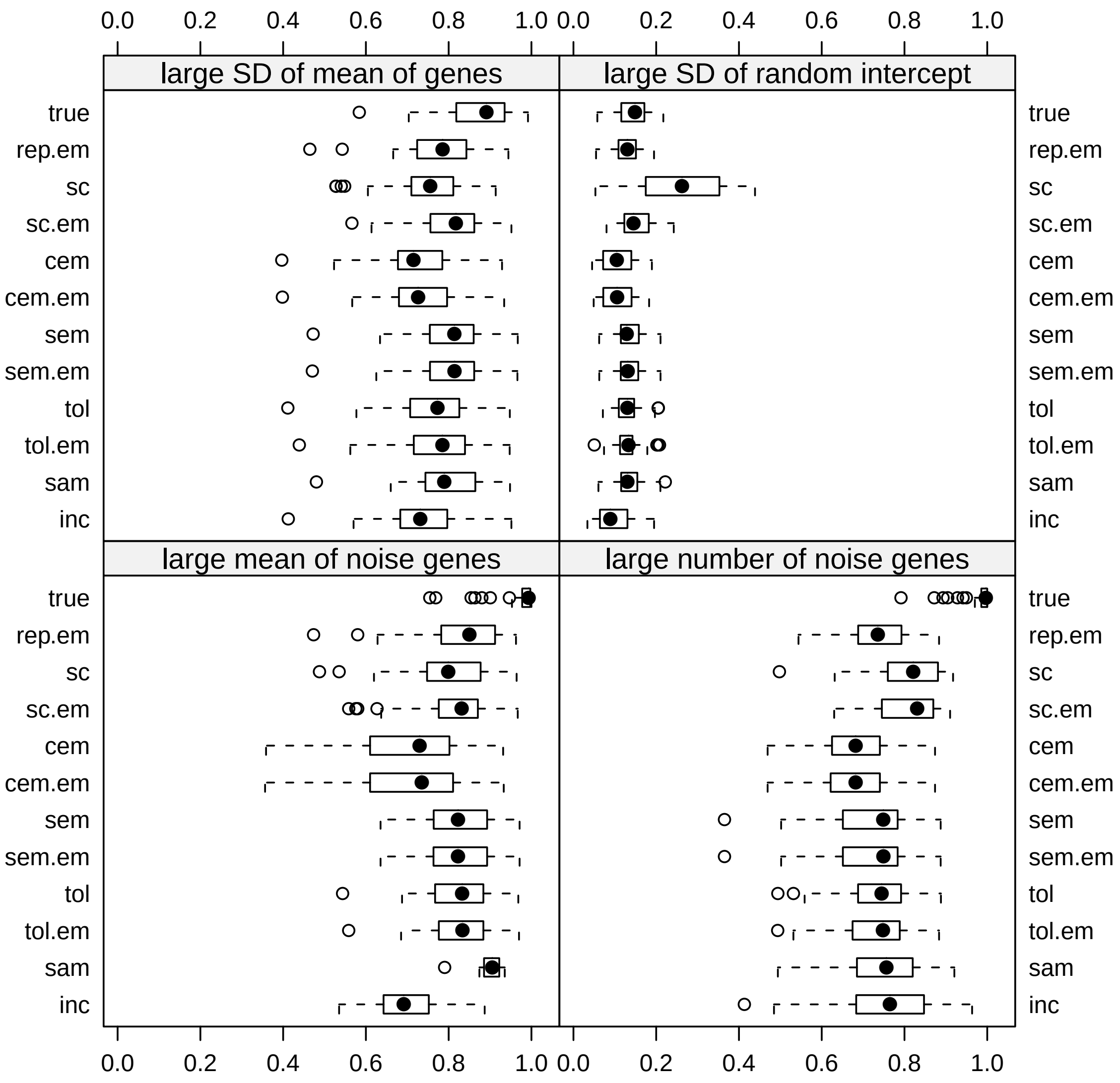


Figure 4.9: Adjusted Rand Index of the different initialization strategies for mixtures of LMs when only one type of noise is present in the data.

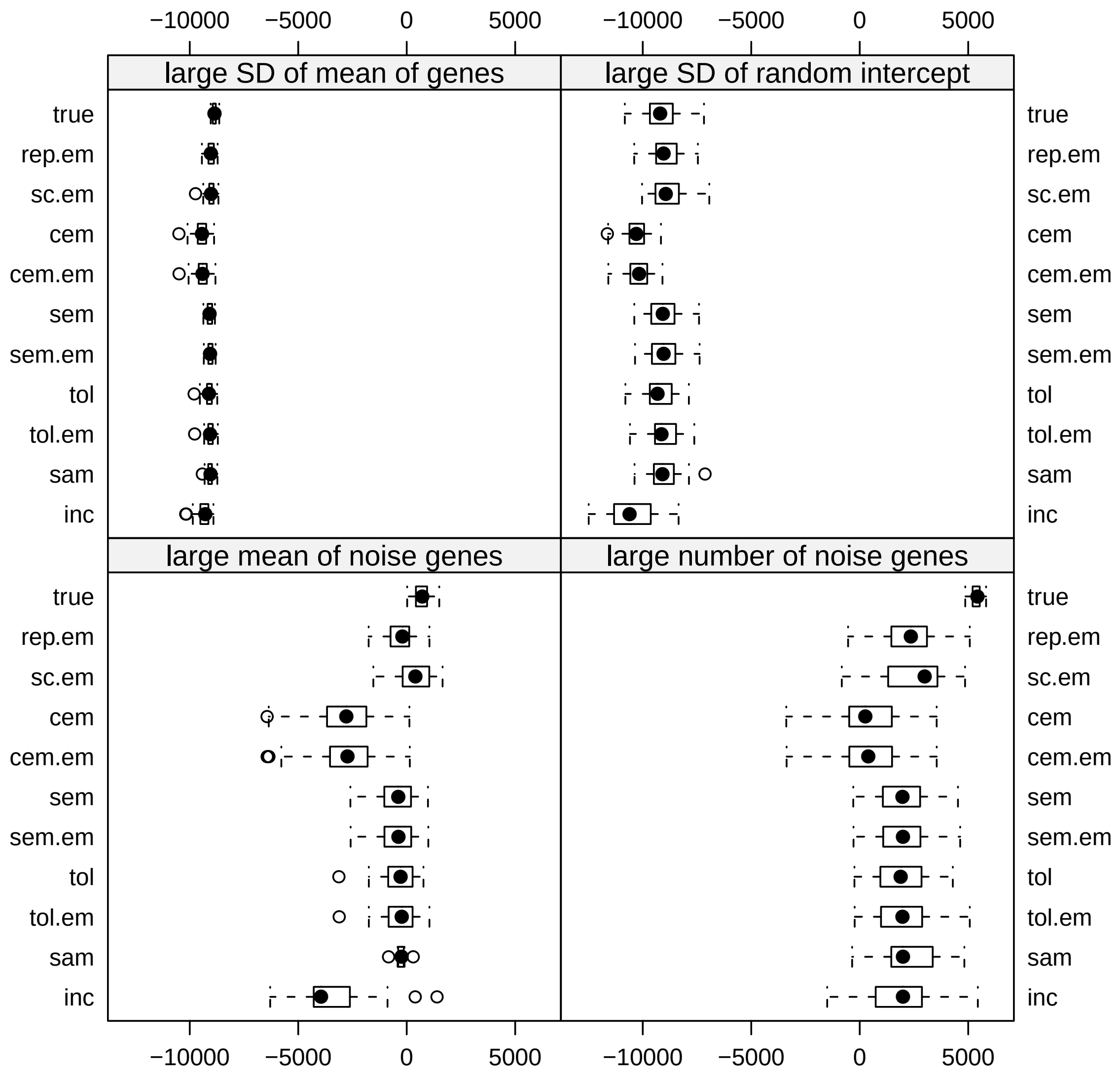


Figure 4.10: Log-likelihood of the different initialization strategies for mixtures of LMs when only one type of noise is present in the data.

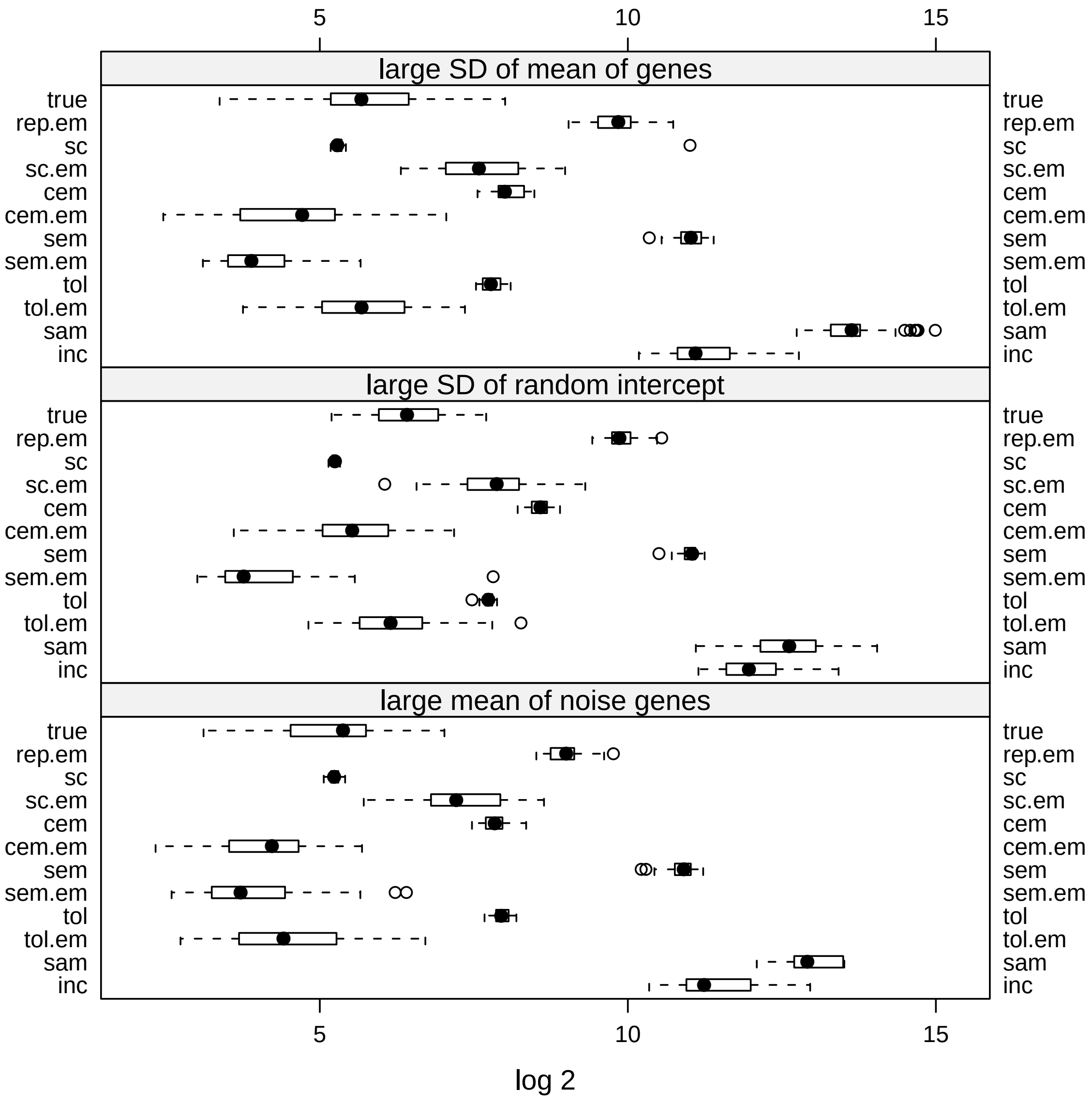


Figure 4.11: System time of the different initialization strategies for mixtures of LMs when only one type of noise is present in the data, i.e., large mean of noise genes, large SD of RI or large mean of noise genes.

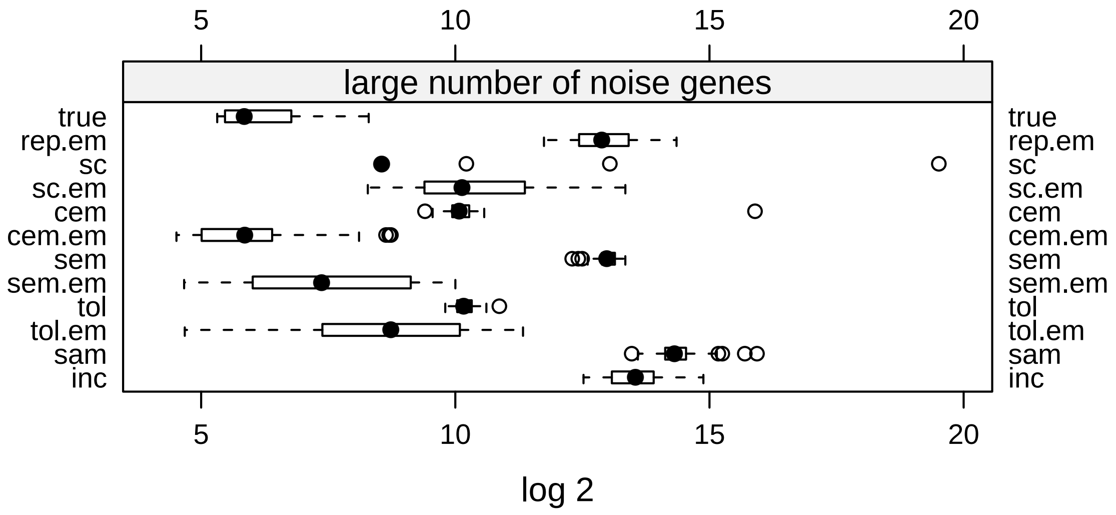


Figure 4.12: System time of the different initialization strategies for mixtures of LMs for a large number of noise genes.

In the case of a large SD of the random intercept the models without RI cannot identify the components no matter what initialization strategy is used. In this case the runtimes of the incremental method, SEM, sampling and random initialization are again very large. Furthermore, spectral clustering outperforms model-based clustering.

For clusters with large mean of the noise genes the agreement between the cluster solutions and the true cluster memberships is in general very high. This indicates that noise genes do not affect the clustering of differentially expressed genes. In this case the incremental method and CEM yield the worst results. Again, the incremental method and SEM have the longest runtimes.

Finally, in the case of a large number of noise genes the true cluster solution clearly outperforms the other initialization strategies but the performance of all methods is very good. However, due to the increase in data size from 730 for a small set of noise genes to 1630 for a large set of noise genes the runtimes of most methods increase dramatically, especially for sampling, the incremental method, random initialization and SEM.

Mixtures of linear mixed models

Next the cluster results of mixtures of linear mixed models (mixtures of LMMs) with RI are summarized in boxplots. Figure 4.13 shows the adjusted Rand index of cluster solutions of the different initialization strategies and the true cluster membership when low, medium and high noise level is present in the data. In contrast to the model without RI (Figure 4.8) where the quality of the cluster solutions decreases tremendously when medium or high noise is added to the data sets now the overall impression of the cluster solutions is much better. Even for high noise level the agreement between the cluster solutions and the true cluster membership is about 60%. CEM and the incremental method perform among the worst whereas starting in the true cluster solution and SEM yield the best results. Additionally mixture models with a random intercept clearly outperform spectral clustering.

The performance of the mixture of LMMs when only one type of noise is present in the data (see Figure 4.14 and 4.15 is also much better compared to the mixture of LMs (Figure 4.9). As expected the performance of all initialization strategies is very good for data generated with a large SD of the random intercept.

The big disadvantage of mixture models with random intercept are the long run times (see Figures 4.16 and 4.17) which are by a factor of 10 longer than the runtimes of the models without RI (see Figure 4.11). However, the trend is the same for models with and without RI. Random initialization, SEM, the sampling and the incremental method cannot be recommended due to the extremely long run times.

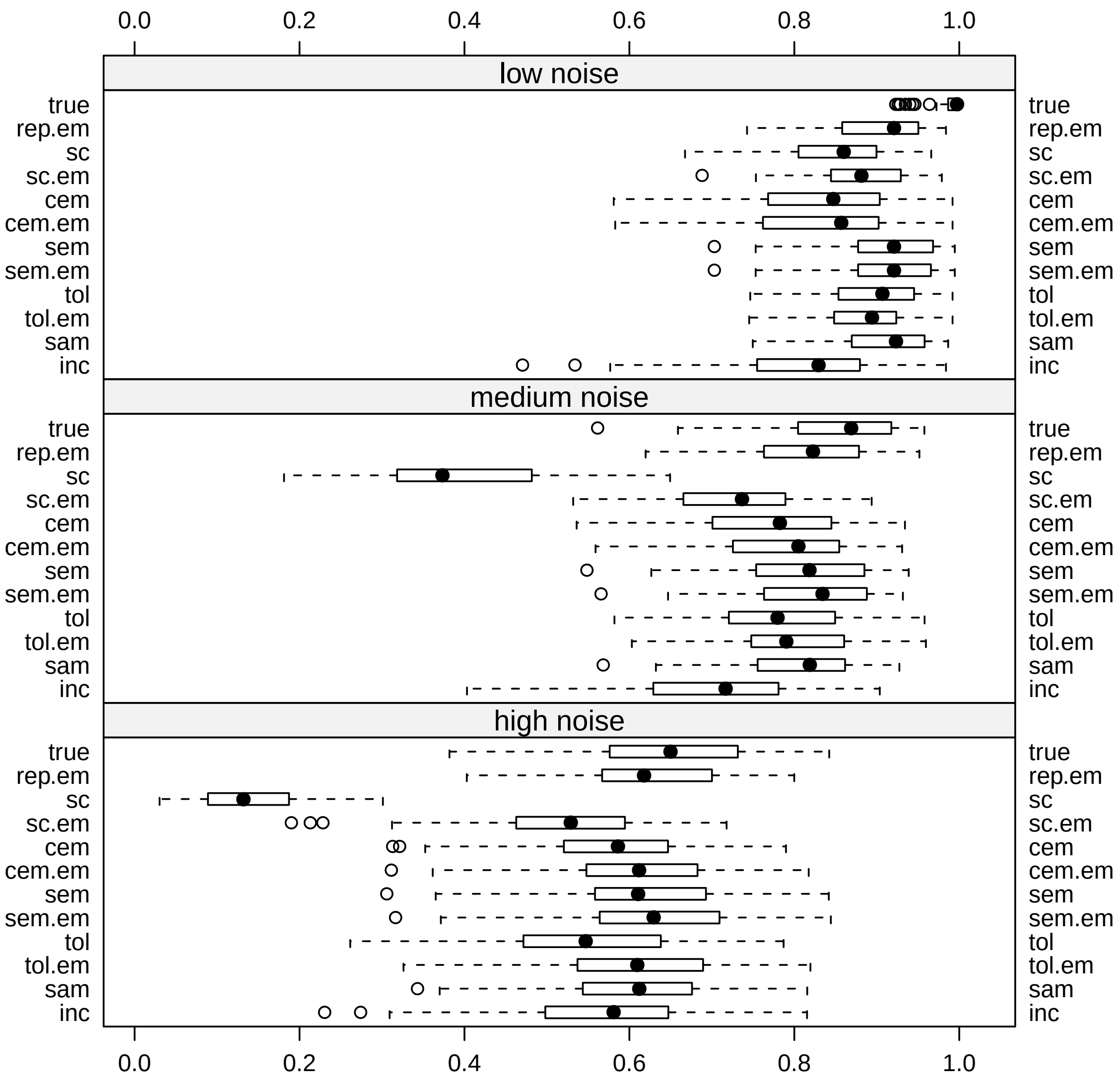


Figure 4.13: Adjusted Rand Index of the different initialization strategies for mixtures of LMMs for low, medium and high noise level.

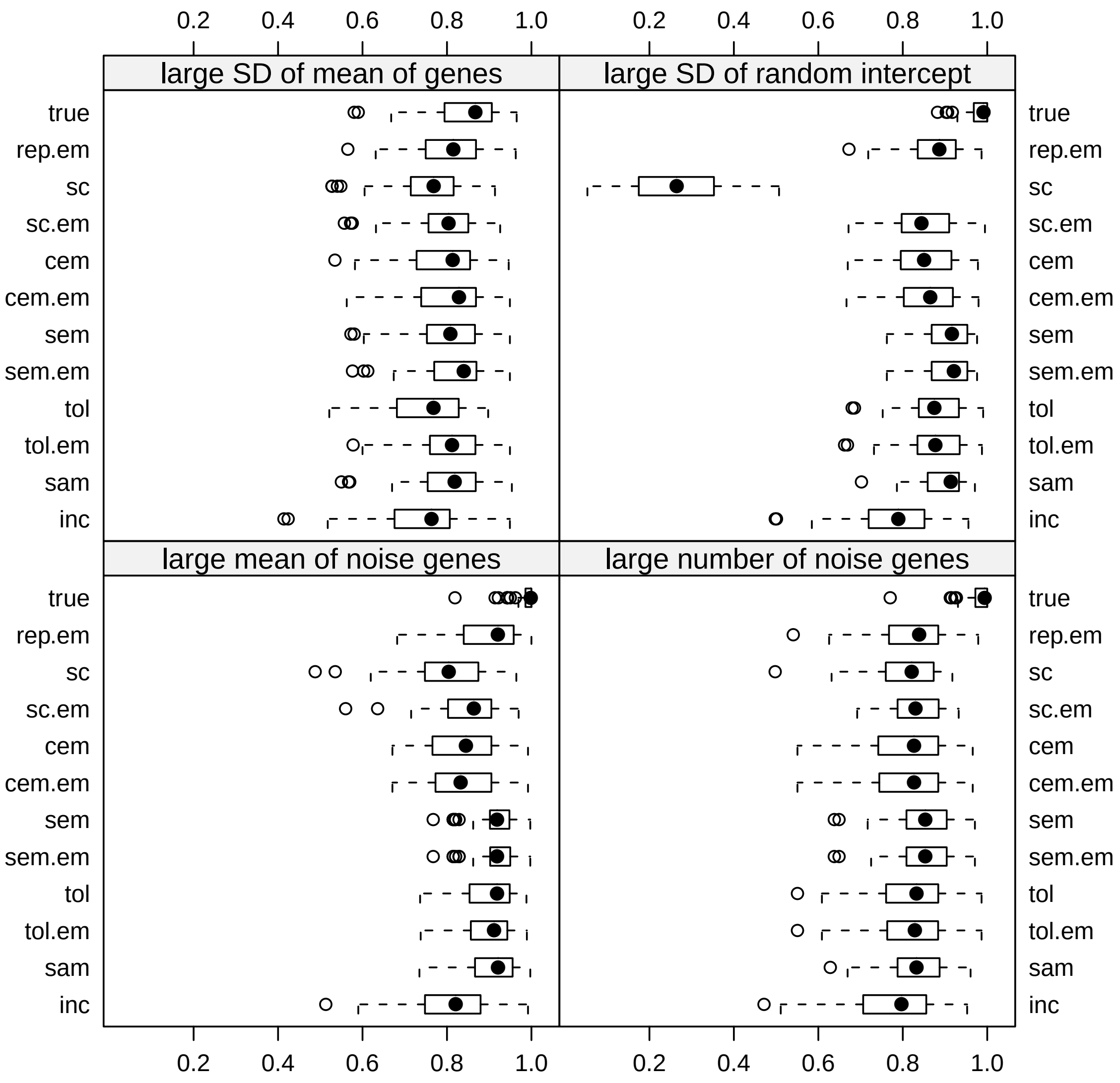


Figure 4.14: Adjusted Rand Index of the different initialization strategies for mixtures of LMMs when only one type of noise is present in the data.

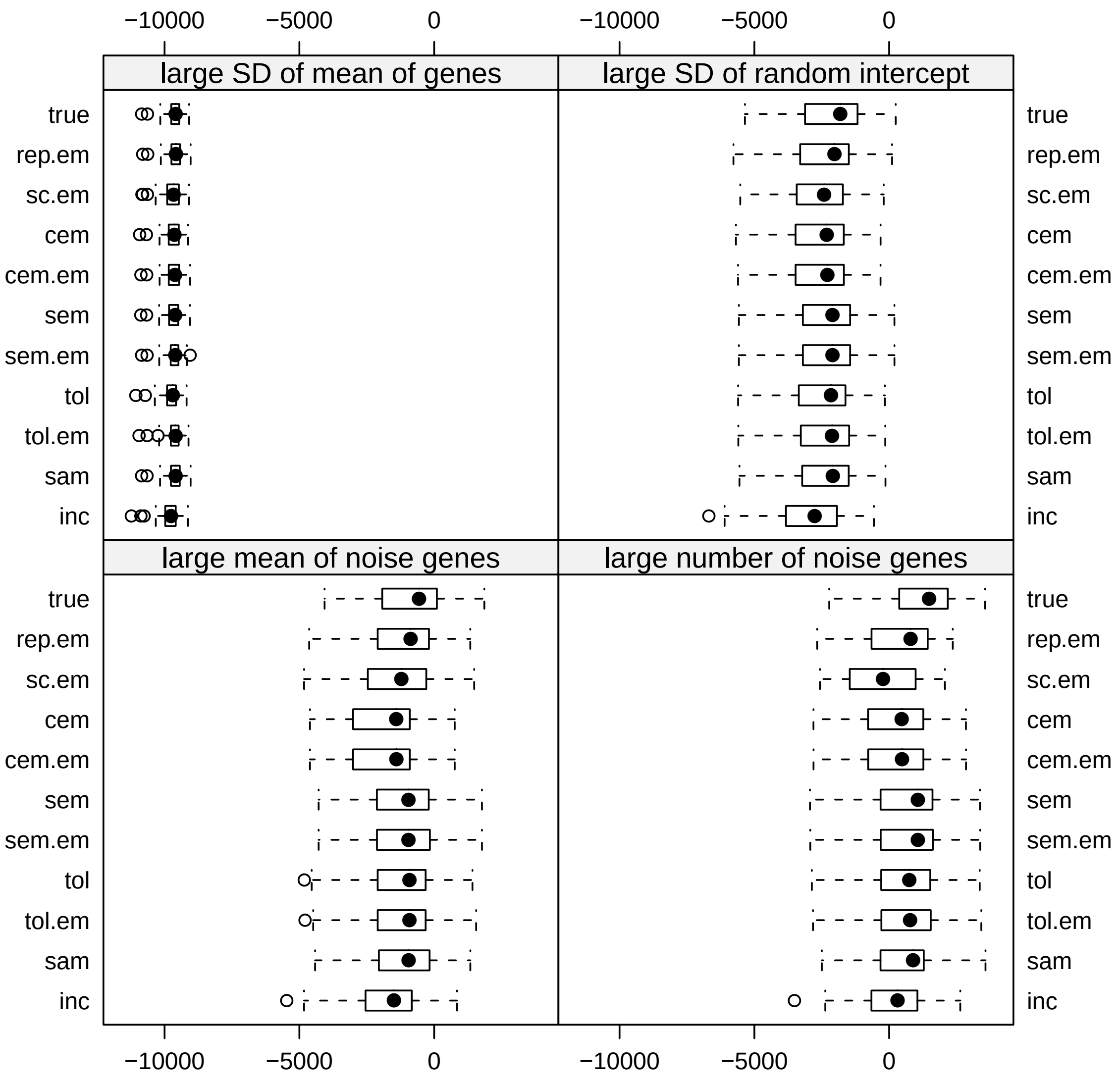


Figure 4.15: Log-likelihood of the different initialization strategies for mixtures of LMMs when only one type of noise is present in the data.

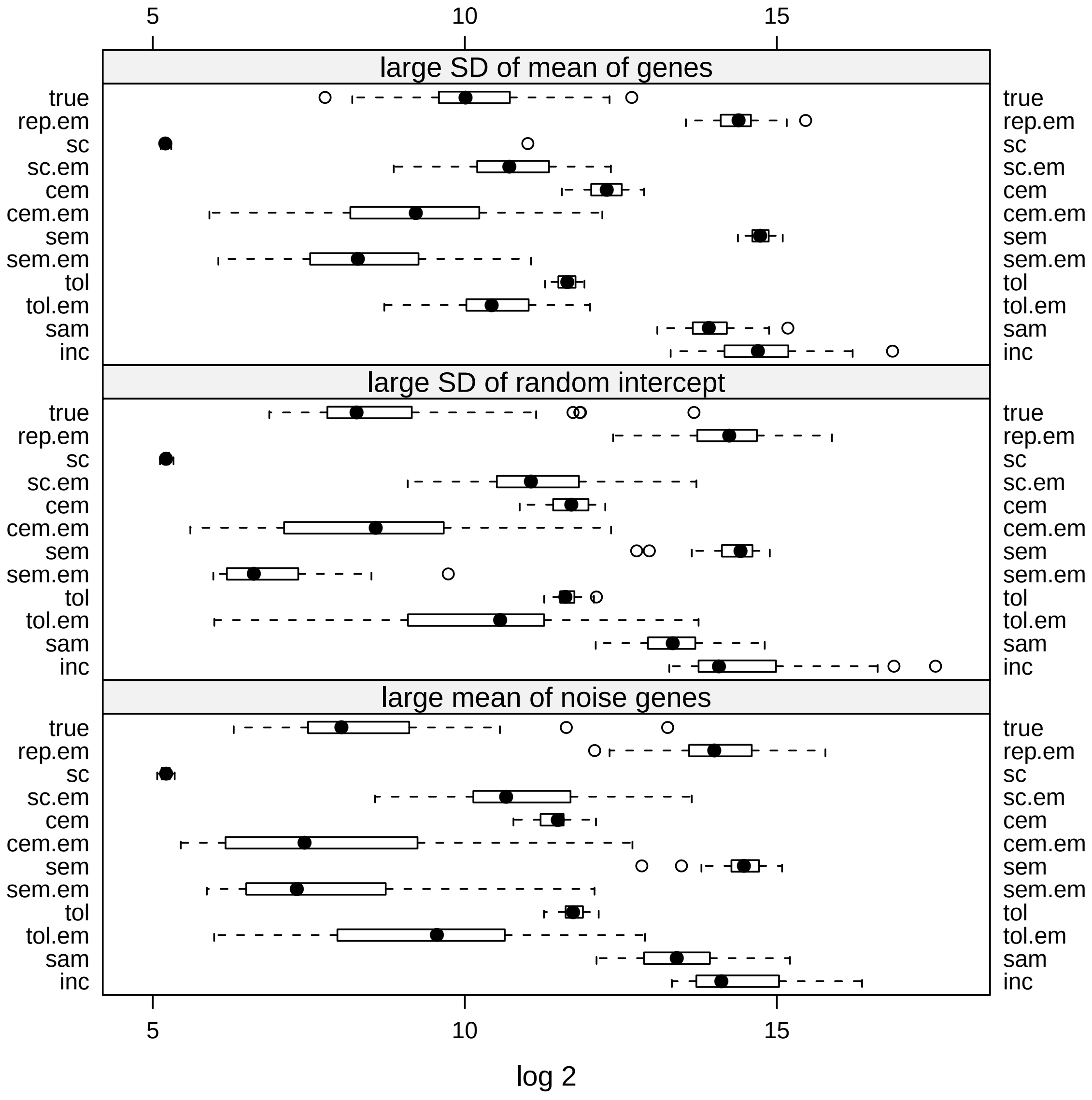


Figure 4.16: System time of the different initialization strategies for mixture models with RI when only one type of noise is present in the data, i.e., large mean of noise genes, large SD of RI or large mean of noise genes.

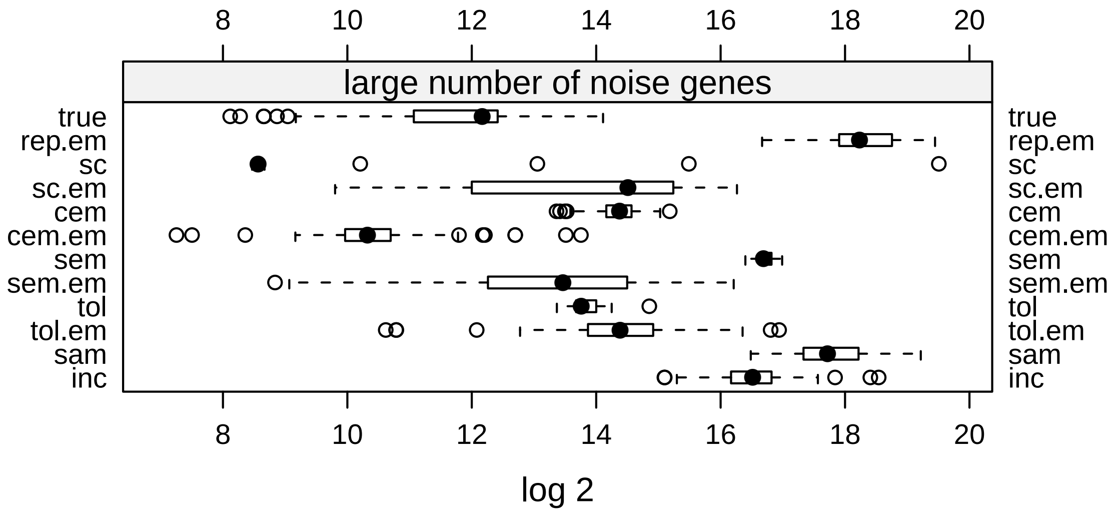


Figure 4.17: System time of the different initialization strategies for mixture models with RI for a large number of noise genes.

4.1.7 Overall comparison

Finally the best cluster methods found in the single comparisons are summarized in Figures 4.18 and 4.19. The selected methods are K–Means and QT–Clust clustering of the original data using Euclidean and "1-Correlation" distance, K–Means and QT–Clust clustering of the functional data as well as mixtures of linear mixed models using the cluster solution of short runs of EM as starting value. In Figure 4.18 the different methods are compared for low, medium and high noise level. For a low noise level QT–Clust on functional data as well as mixture of LMMs yield the best results. For medium and high noise level mixtures of LMMs and K–Means clustering using "1-Correlation" distance are the methods of choice whereas K–Means and QT–Clust clustering of the original data as well as functional data using Euclidean distance perform very poorly.

In Figure 4.19 the different methods are compared when only one type of noise is present in the data. For a large SD of the mean of genes (top left panel) the best method is mixtures of LMMs followed by the classical K–Means clustering of the original data using Euclidean distance. The methods performing worst are clustering the functional data using K–Means and QT–Clust. In the case of a large SD of the random intercept (top right panel) mixtures of LMMs are again the method of choice but K–Means clustering using "1-Correlation" distance yields almost as good results. In this noise setting Euclidean distance should not be used on both the original as well as the functional data.

For a large mean of the noise genes (bottom left panel of Figure 4.19) all methods perform very well. Mixtures of LMMs as well as clustering the functional data yield better results than clustering the original data using Euclidean distance. More or less the same is true for a large number of noise genes (bottom right panel).

4.1.8 Conclusions

In this simulation study several cluster methods for time course gene expression data were evaluated on artificial data sets with different types of noise. For the quality–based clustering approach QT–Clust the impact of the radius on the number of clusters, the number of outliers, the sum of within cluster distances and the Rand index between repeated calls of QT–Clust was investigated. A small radius yields a large number of outliers and a large number of clusters. On the other hand, for a larger radius the sum of within cluster distances increases and the Rand index decreases rapidly. The region of good values for the radius is rather small and needs to be investigated individually for each data set and distance measure used.

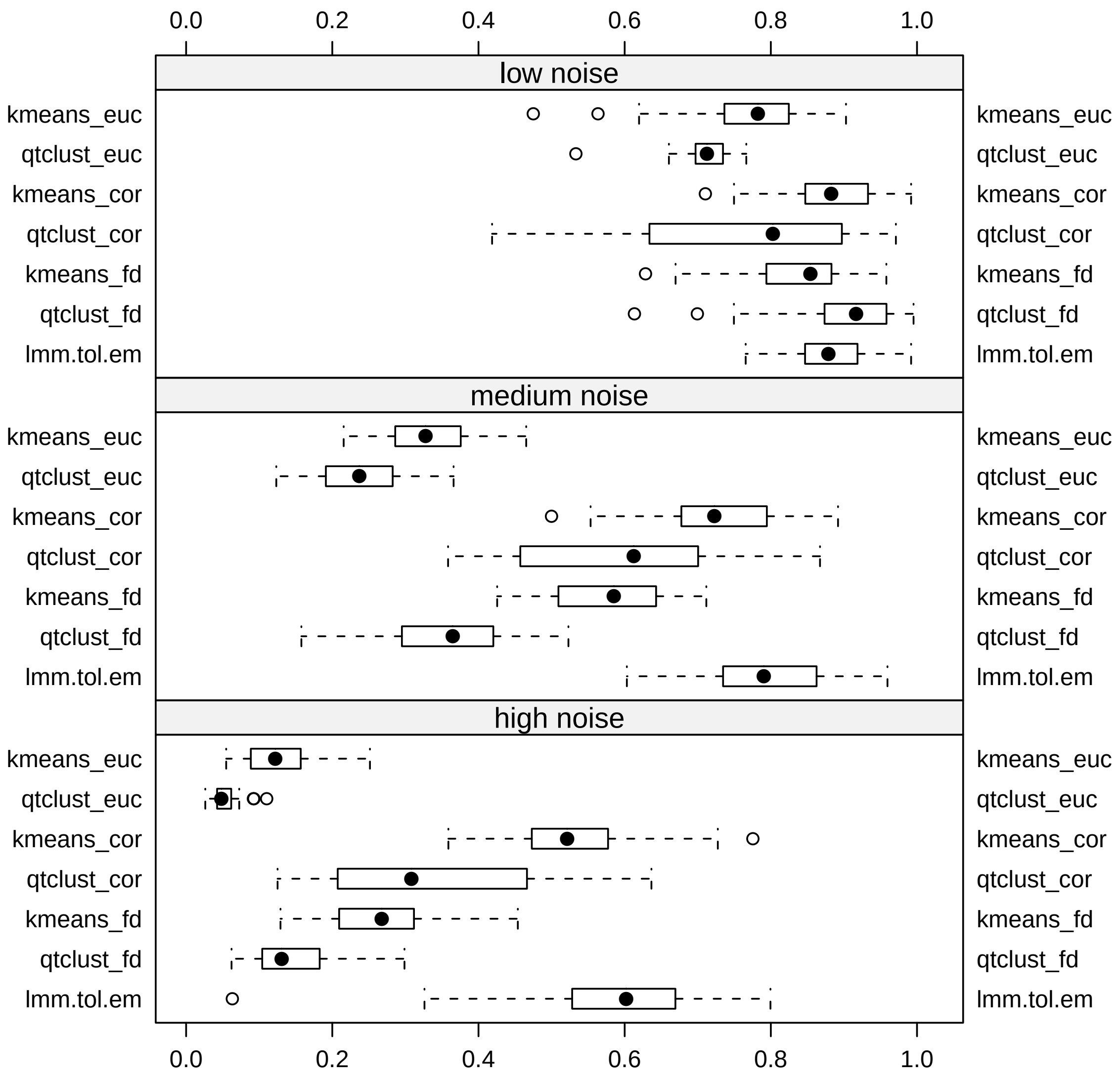


Figure 4.18: Comparison of selected methods using the adjusted Rand index on 100 data sets with low, medium and high noise level added to the cluster centers.

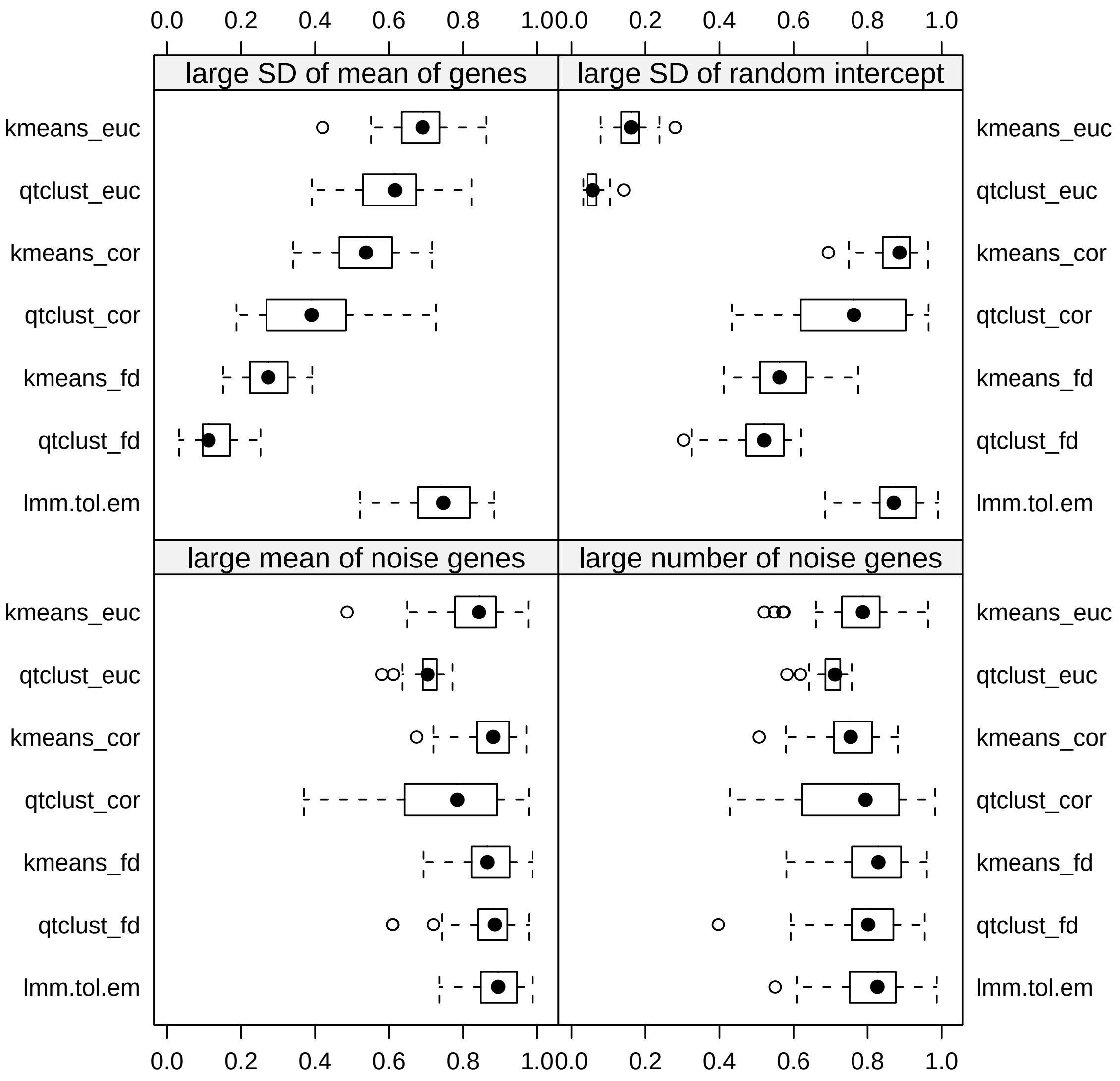


Figure 4.19: Comparison of selected methods using the adjusted Rand index on 100 data sets when only one type of noise is present in the data.

The behaviour of different versions of the adjusted Rand index was investigated and it was found that only a very large number of outliers has an impact on the modified Rand index.

Some general observations about model–based clustering were made for this type of data:

- For noisy data sets mixtures of LMs should not be used.
- Mixtures of LMMs clearly outperform mixtures of LMs and spectral clustering on noisy data sets. However, the user should be aware of the much longer runtimes.
- Running SEM, CEM or short runs of EM and using their best solution for the initialization of EM does hardly increase the performance already reached by these strategies. This was observed using the classification criterion as well as the likelihood criterion.
- Computationally intensive methods like the sampling or incremental method are hardly worth the effort.
- Random initialization yields very long runtimes compared to CEM or short runs of EM. However, the cluster results are similar.
- The impact of the cluster method used is much larger than the impact of the initialization strategy.
- For short runs of EM the tradeoff between the quality of the cluster solutions and runtime is very good.

The results of the overall comparison of cluster methods are the following:

- For a low noise level there was not so much difference between the methods.
- However, for a high noise level only K–Means clustering of the raw data using "1-Correlation" and mixtures of linear mixed models performed well and had an adjusted Rand index larger than 0.5.
- In the case of a large SD of the mean of genes the best method was mixtures of LMMs followed by the classical K–Means clustering of the original data using Euclidean distance. The methods performing worst were clustering the functional data using K–Means and QT–Clust.

- In the case of a large SD of the random intercept mixtures of LMMs were again the method of choice but K–Means clustering using “1-Correlation” distance yielded almost as good results. In this noise setting Euclidean distance should not be used on both the original as well as the functional data.

- For a large mean of the noise genes and a large number of noise genes all methods performed well. Mixtures of LMMs as well as clustering the functional data yield better results than clustering the original data using Euclidean distance.

4.2 Yeast data

For the simulations presented in this section a publicly available data set from yeast was used (in the following called the "yeast data set"). It is the seventeen time point mitotic cell cycle data (Cho et al. 1998) available at **http://genome-www.stanford.edu**. This data set was preprocessed adapting the instructions given by Heyer et al. (1999).

4.2.1 Evaluation of stability and variance of QT–Clust

In this section the sum of within cluster distances and stability of the stochastic variant of QT–Clust are investigated in order to compare it to the original algorithm. The yeast data was preprocessed removing the outlier time points 10 and 11 from the original 17 variables. Then genes that were either expressed at very low levels or did not vary significantly over the time points were removed. This procedure yields gene expression data on $G = 3722$ genes (observations) for $T = 15$ time points (variables).

100 replicates of QT–Clust each are computed for increasing values of the hyper parameter `ntry` between 1 and 3300. For `ntry` equal to the number of genes the algorithm is deterministic and equivalent to the original algorithm. Figure 4.20 shows boxplots of the sum of within cluster distances for all five distance measures after rescaling. On this data set the parameter `ntry` has major impact on the quality of the partition. Even though on this data set the variability is higher for small values of `ntry` this may lead to better results. Smaller values of the sum of within cluster distances can be obtained using small values of `ntry`. Additionally the different distance measures show different patterns and all suggest different values of `ntry`. Values between 1 and 100 always lead to to better results than the original algorithm on this data set. Figure 4.21 shows boxplots of all consecutive pairwise comparisons of cluster results for 100 replicates using the adjusted Rand index (Hubert and Arabie 1985). The stability of QT–Clust increases for increasing values of `ntry` except for "1-Jackknife Correlation". This distance measure is different to all others as it allows single outliers. It would be interesting to see if Jackknife versions of Euclidean, Manhattan and Maximum distance show similar characteristics. So if one is interested in clusters with small sum of within cluster distances stochastic QT–Clust performs better than the original algorithm on this data set. If reproducibility is important then the deterministic algorithm is preferable.

As there is no general "best" value for `ntry` different numbers have to be tested for each data set and each distance measure in order to find a suitable value. Now the simulations for `ntry` = 5, 100, 1000 and 3300 were compared for all distances. Figure 4.22 shows boxplots of the stability within a single value of `ntry` and between results for different values of `ntry`. "1–Jackknife Correlation" reveals fewer differences than the other distance measures. This may indicate that the instability originates from outliers in single coordinates. For all further simulations on this data set `ntry` = 5 is used to obtain cluster results with small sum of within cluster distances. Additionally the use of smaller values of `ntry` speeds up the procedure.

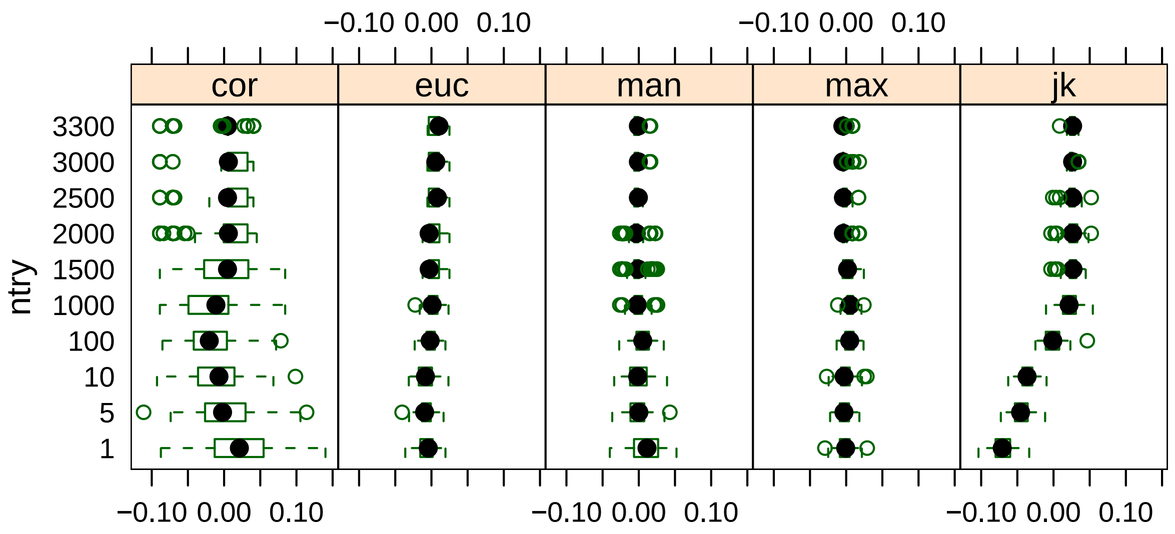


Figure 4.20: Sum of within cluster distances of QT–Clust for increasing values of the hyper parameter `ntry` after rescaling.

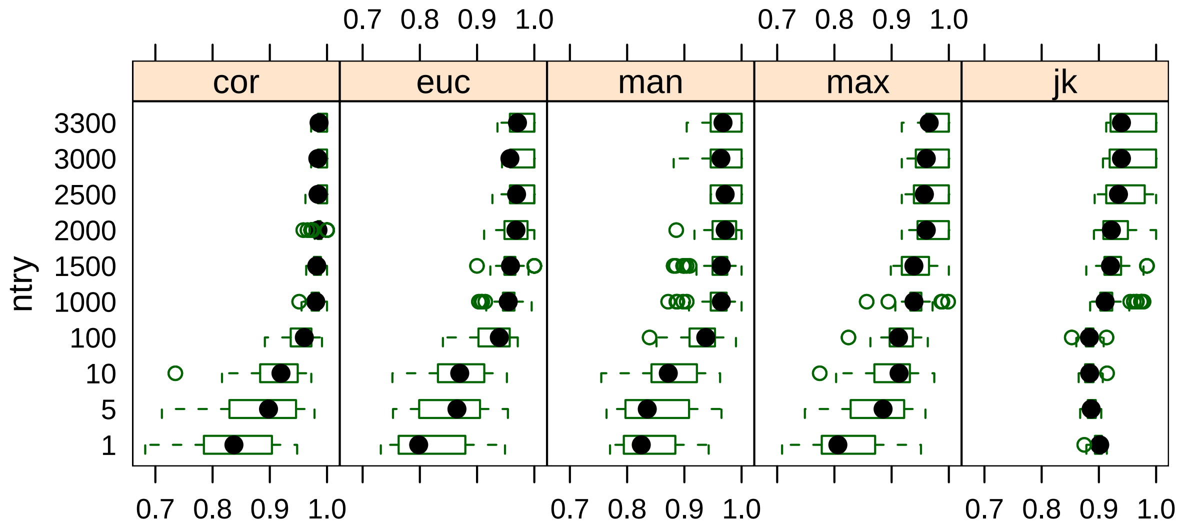


Figure 4.21: Stability of the stochastic approximation of QT–Clust for increasing values of the hyper parameter `ntry`. Pairwise comparison of the results using boxplots of the Rand indices.

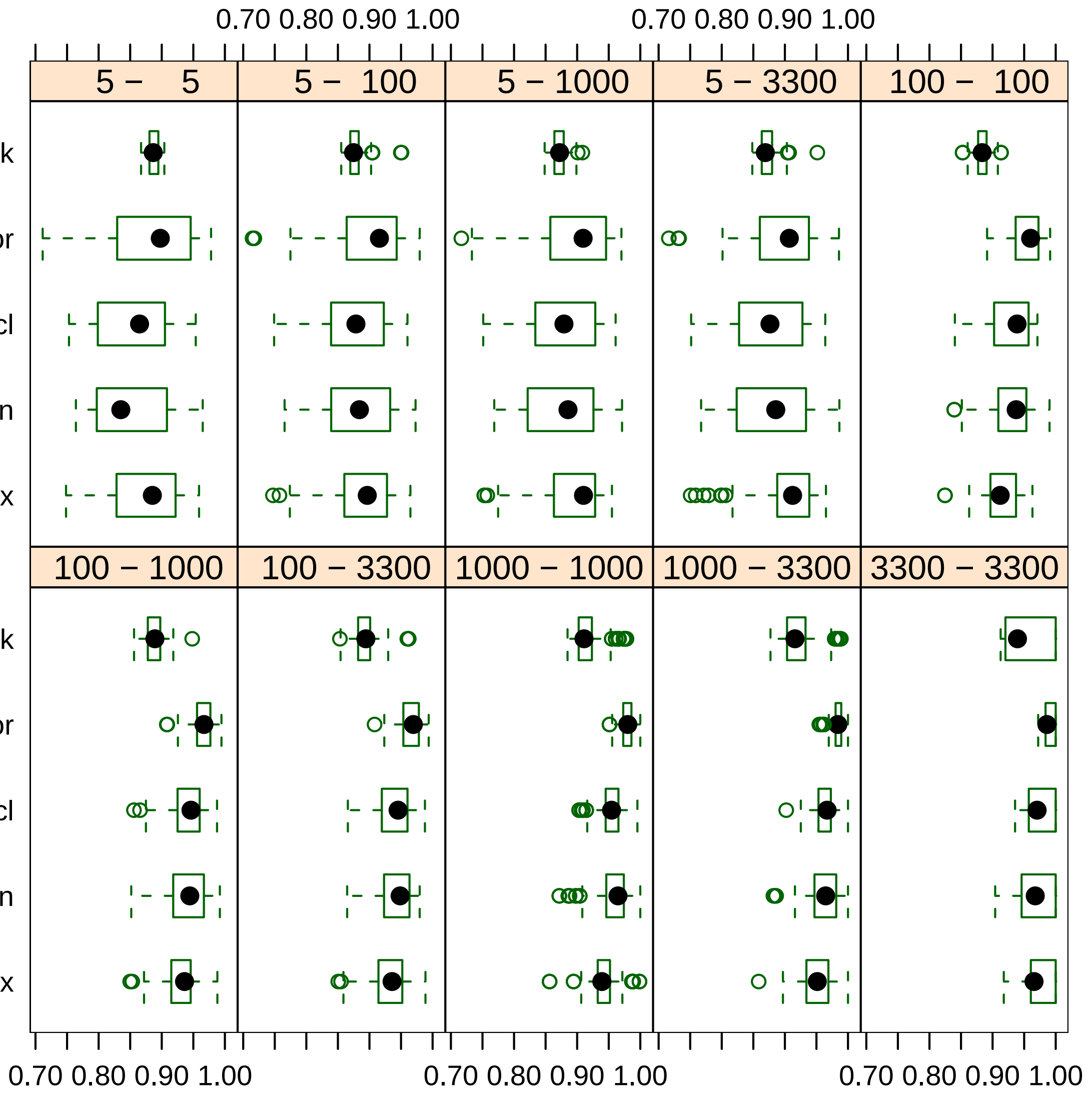


Figure 4.22: Rand index for pairwise comparison of `ntry` = 5, 100, 1000 and 3300 on 100 replicates within and between values of `ntry`.

4.2.2 The impact of Jackknife distance measures

In this section the new distance measure presented in Section 2.1.3 are evaluated. In this case the yeast data was preprocesses differently. The genes were again rescaled. Additionally genes that were expressed at very low levels and did not vary significantly over the time points were removed. This time the filtering was more stringent in order to get a smaller data set. This procedure yields gene expression data on $G = 2090$ genes (observations) for $T = 17$ time points (variables). As time point 10 was reported to be an outlier variable the simulations were conducted on the 17 time point data set as well as on a data set with time point 10 removed to investigate the functionality of the Jackknife distance measures.

The goal of this simulation study is to compare the four classical distance measures to the Jackknife distance measures using both K–Means and stochastic QT–Clust. For K–Means the following procedure is used

1. draw 100 bootstrap samples from the original data,
2. cluster them into 50 clusters using each of the distance measures, and
3. compare the obtained results using the sum of within cluster distances and the adjusted Rand index.

The sum of within cluster distances is computed as a measure of the quality of a partition. The Rand index is used as a measure of stability and reproducibility of the resulting clusters and the agreement between partitions.

The number of clusters for K–Means is chosen arbitrarily. As biologists prefer to work with smaller groups of genes to be able to take a closer look at the resulting clusters the genes were grouped into 50 clusters for K–Means. For QT–Clust 100 replicates of the algorithm were computed for each distance measure on the original data because the algorithm has no prediction step. It was tried to find an appropriate radius for QT–Clust to get a similar numbers of clusters like in K–Means. As observed in Section 4.1.3. The number of clusters of QT–Clust is changing with the diameter of the clusters and the minimal number of points that form a single cluster. Therefore the number of clusters of QT–Clust is varying between distance measures and even between replicates of QT–Clust using the same distance measure.

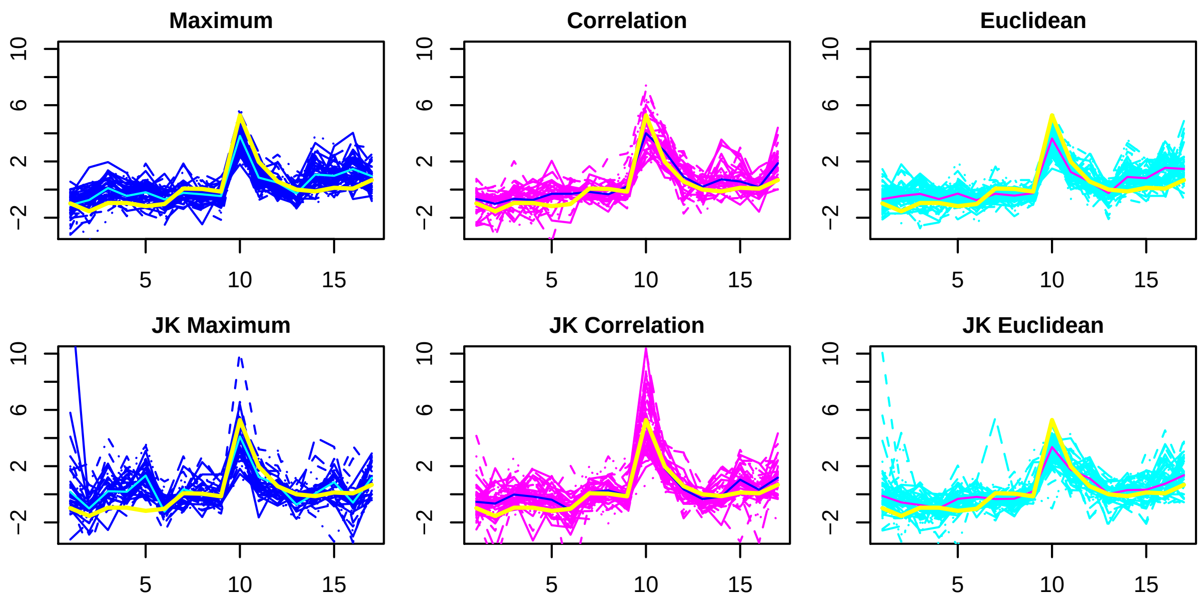


Figure 4.23: The clusters of gene YDL223C (yellow) for Maximum, "1 - Correlation" and Euclidean distance and their Jackknife versions using K–Means algorithm. Time points are shown on the x–axis and gene expression is shown on the y–axis.

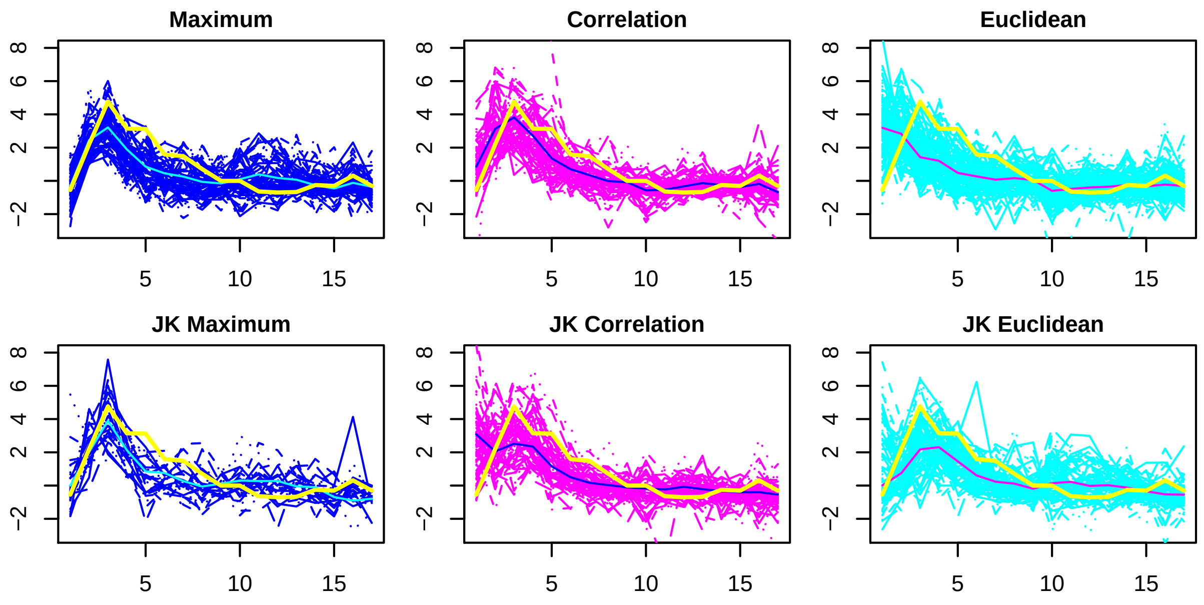


Figure 4.24: The clusters of gene YDR044W (yellow) for Maximum, "1 - Correlation" and Euclidean distance and their Jackknife versions using QT–CLUST algorithm.

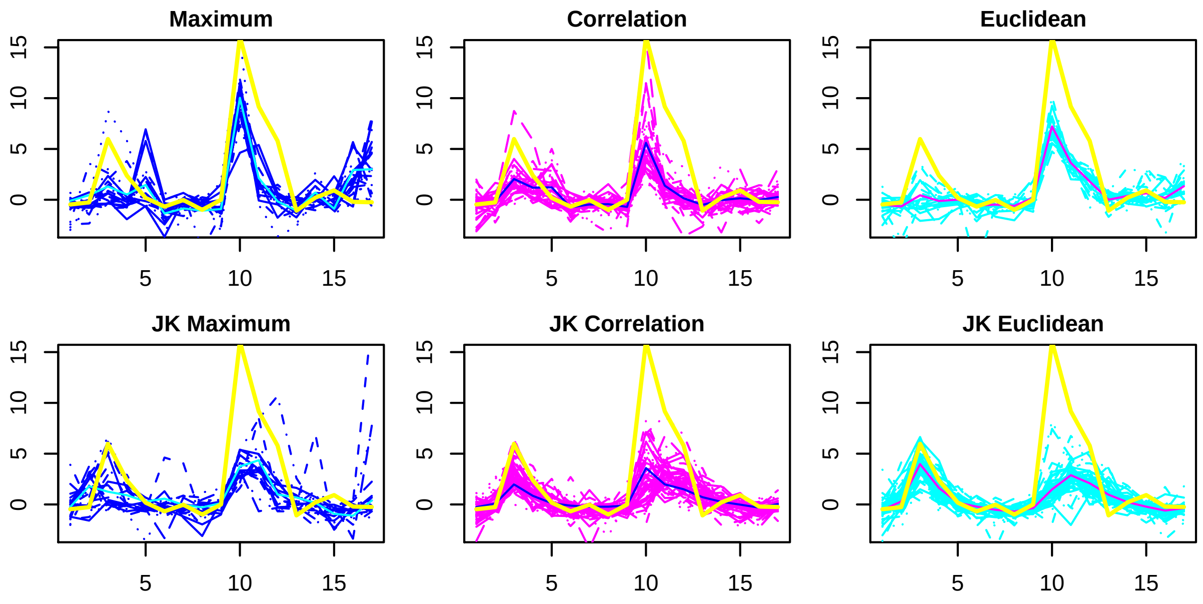


Figure 4.25: The clusters of gene YDR006C (yellow) for Maximum, “1 - Correlation” and Euclidean distance and their Jackknife versions using K–Means algorithm.

Some example clusters

Some example clusters of the yeast data can be seen in Figures 4.23, 4.24 and 4.25. The clusters of the genes YDL223C, YDR044W and YDR006C are observed for Maximum distance, “1 – Correlation” distance and Euclidean distance and their Jackknife versions using the two different cluster algorithms. Figure 4.23 shows gene expression profiles with a clear peak at time point 10. Figure 4.24 shows genes with high activity at time point 3 and Figure 4.25 contains genes with high gene expression at time point 3 as well as time point 10. As Figure 4.25 shows genes might fall into different clusters when allowing for outlier variables. Using Jackknife distance measures the clusters of gene YDR006C (Figure 4.25) have a stronger peak at time point 3 and a new peak at time point 11 instead of time point 10.

Stability of the resulting clusters

As researchers want to know how reliable the resulting clusters are the stability of the cluster results is investigated in this section. For that purpose all consecutive pairwise comparisons of cluster results are computed using the adjusted Rand index. 100 replicates on the original data for QT–Clust and 100 bootstrap samples of K–Means are used. Box-

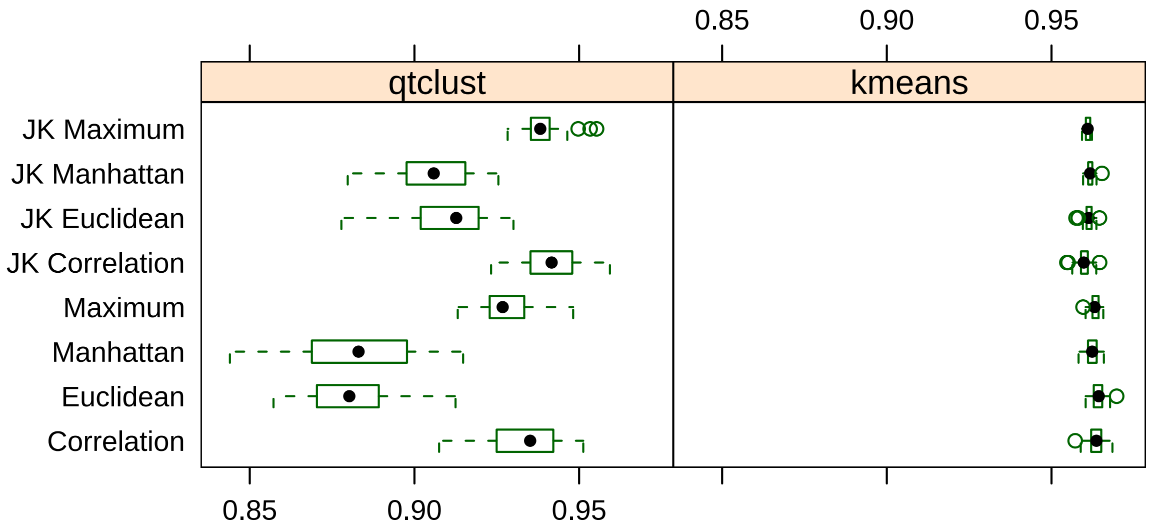


Figure 4.26: Boxplots of the Rand indices for all pairwise comparisons of 100 replicates of QT–Clust (left panel) and 100 bootstrap samples of K–Means (right panel) within different distance measures.

plots of the Rand index of all consecutive pairwise comparisons are shown in Figure 4.26.

It can be seen that the reproducibility of cluster results is in general very high for K–Means. The Rand index for consecutive comparisons within distance measures is over 0.95 for all distance measures. For QT–Clust the stability of cluster results is smaller that for K–Means. Here the stability is highest for “1 – Jackknife Correlation” distance followed by Jackknife Maximum distance. The reproducibility of cluster results using Manhattan and Euclidean distance is much lower on this data set. For QT–Clust it can be clearly seen that for all distance measures the stability of the Jackknife version of the distance measure is higher than the stability of the original distance measure.

Quality of the partitions

As a measure of the quality of a partition the sum of within cluster distances is used. It was computed for each distance measure for all of the 100 replicates on the original data using QT–Clust and 100 bootstrap samples of K–Means. Figure 4.27 shows boxplots of the sum of within cluster distances for all distance measures.

The sum of within cluster distances is not directly comparable between distance measures. But Figure 4.27 shows that using the Jackknife version of any of the four distance

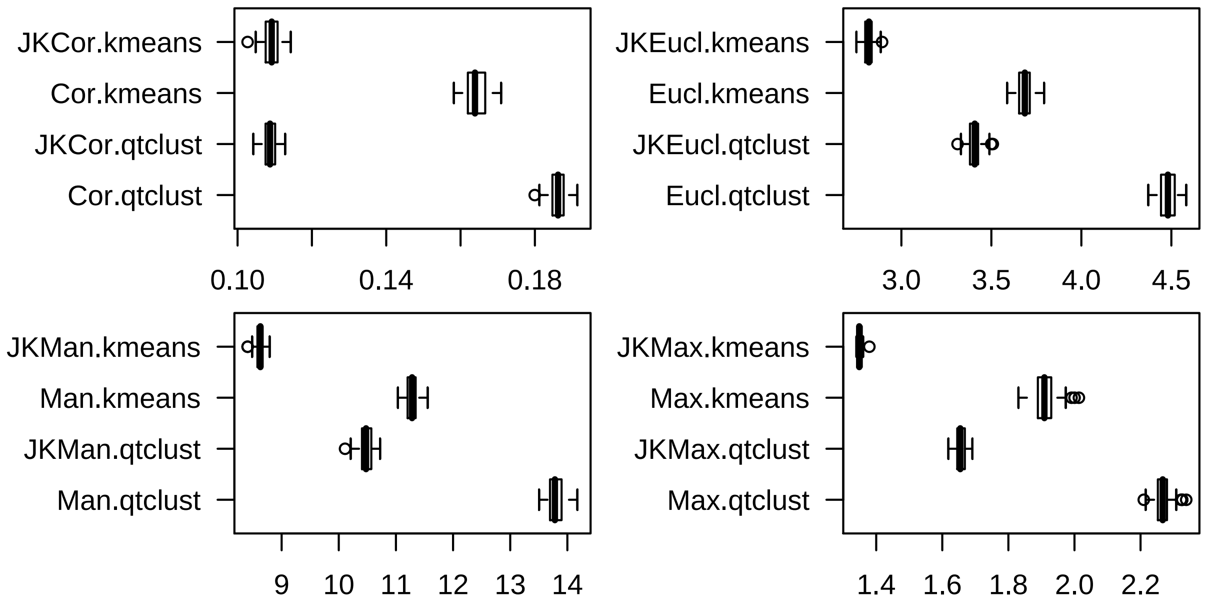


Figure 4.27: Boxplots of the sum of within cluster distances for 100 replicates of QT–Clust and 100 bootstrap samples of K–Means for the different distance measures.

measures leads to a smaller sum of within cluster distances for both QT–Clust and K–Means. This means that the similarity between genes within a cluster is higher using Jackknife distance measures. Additionally the sum of within cluster distances is smaller for K–Means than for QT–Clust for all distance measures.

Comparing different distance measures

Now the differences between cluster results of different distance measures are investigated. The agreement between partitions is computed using the Rand index. The results are shown in Figure 4.28.

For QT–Clust the agreement between "1 - Correlation" distance and Maximum distance and their Jackknife versions is very high as well as the agreement between these Jackknife versions. The results obtained using Manhattan and Euclidean distance disagree most.

For K–Means the impact of the distance measure used is much smaller on this data set. The agreement between cluster results using different distance measures is very high for this cluster algorithm. Here the partitions agree most for the Jackknife versions of Maximum distance and "1 - Correlation" distance and disagree most for Maximum distance and "1 - Correlation" distance.

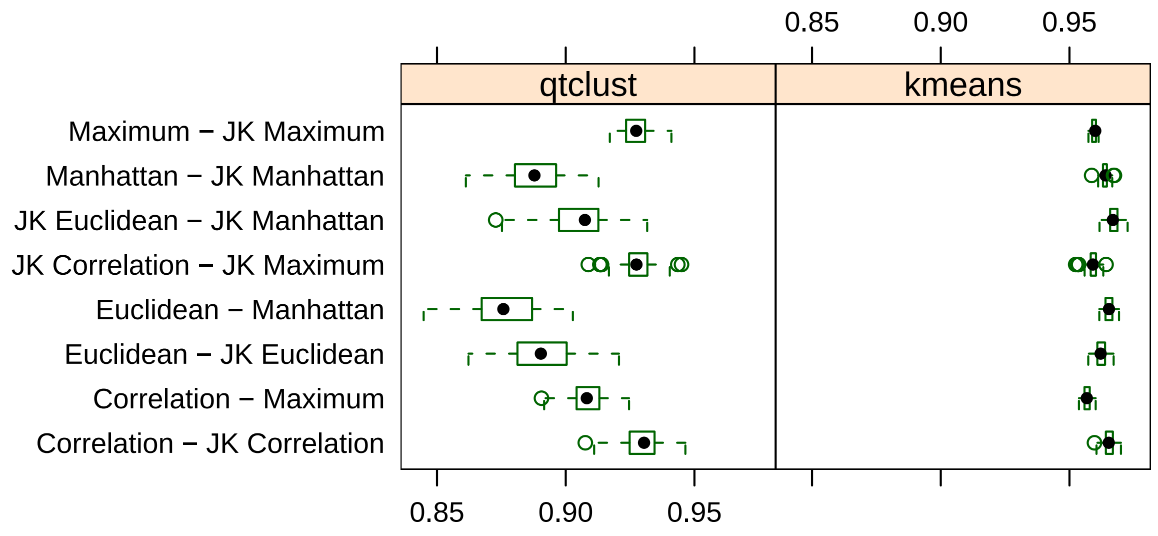


Figure 4.28: Rand index for pairwise comparison of 100 replicates of QT–Clust (left panel) and 100 bootstrap samples of K–Means (right panel) between different distance measures.

Excluding one outlier time point

As time point 10 was reported not to be dependable (Heyer et al., 1999) the simulations were repeated on a smaller data set with this time point deleted. Again the partitions are compared using the Rand index. First the cluster results obtained on each data set are compared for all distance measures. The results are shown in Figure 4.29. For K–Means the agreement between results on the full data set and on the data set with one time point removed is again very high (over 0.95). For QT–Clust the pattern of agreement is very similar to Figure 4.26. This means that the cluster results agree between the two data sets and excluding the outlier time point does not change the results significantly.

Finally we want to find out how good the Jackknife distance measures work and how frequently the outlier time point 16 is detected and excluded from calculating the distance between two gene expression profiles. For that purpose the original data set using Jackknife distance measures is compared to the 16 time point data set using the original distance measures (see Figure 4.30). It was found that on this data set the agreement between cluster results for Jackknife distance measures on the original data set and original distance measures on the data set with one outlier time point deleted is very high for Maximum distance and "1 - Correlation" distance using QT–Clust. Therefore Jackknife distance

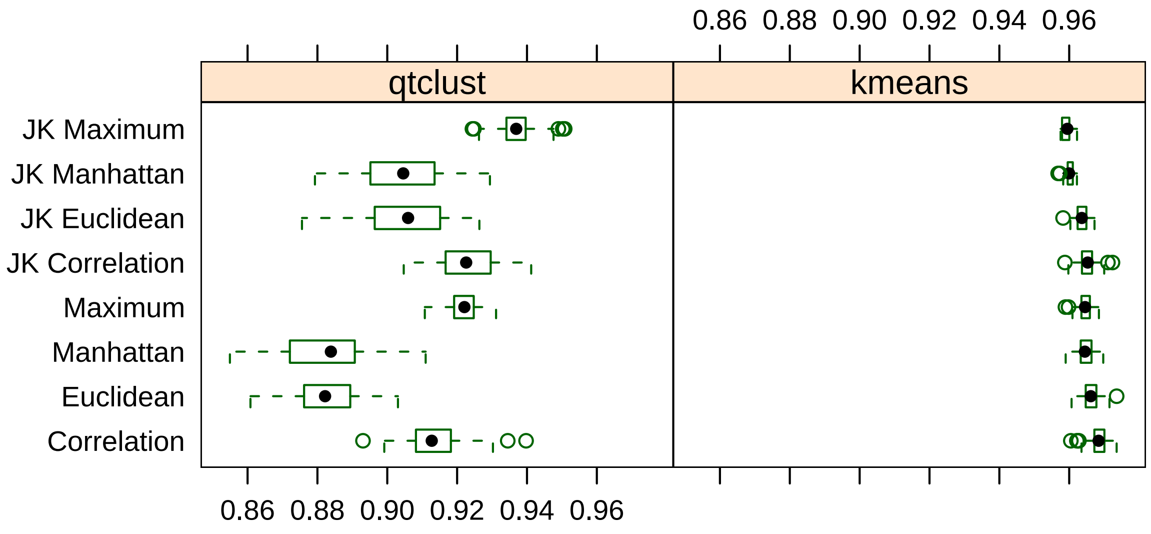


Figure 4.29: Rand index for pairwise comparison between the full data set and the 16 time point data set for QT–Clust (left panel) and K–Means (right panel) using different distance measures.

measures are applicable when outliers at special time points distort the gene expression data.

As the example clusters in Figure 4.24 show time point 10 is not the most extreme observation for all genes. For those genes other observations than time point 10 will be detected by the Jackknife distance measures.

4.2.3 Conclusions

On the yeast data set it was found that stochastic QT–Clust leads to better results than the original QT–Clust algorithm if one is interested in a small sum of within cluster distances. If stability and reproducibility are more important then the original algorithm is preferable even though it results only in a local optimum.

The impact of Jackknife distance measures is summarized as follows

- The distance measure used for clustering time–course gene expression data has major impact on the resulting clusters.
- The sum of within cluster distances is smaller for the Jackknife versions of the distance measures. This might be an indicator that Jackknife distance measures lead

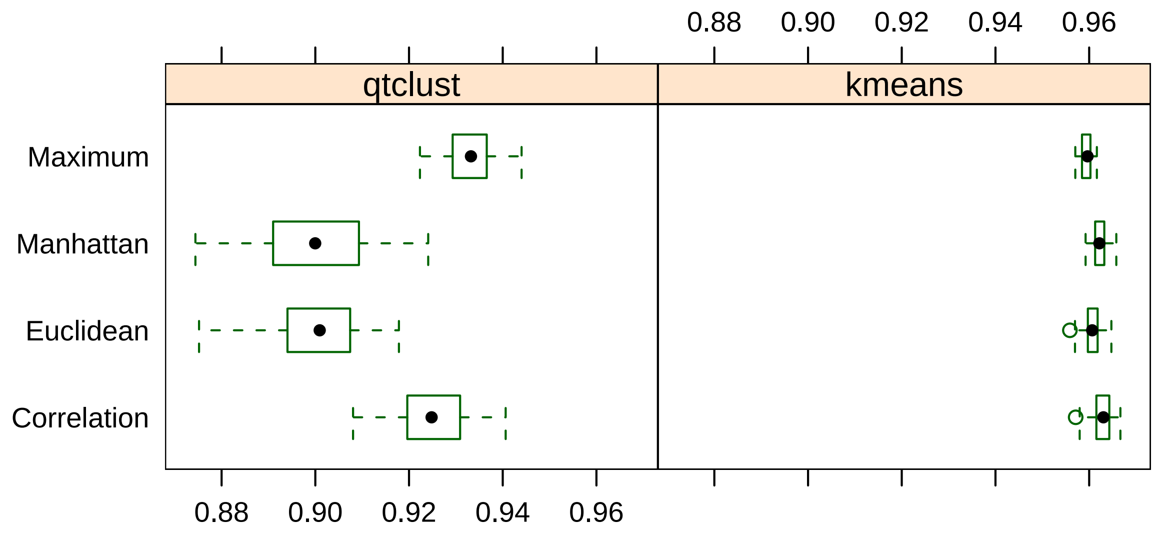


Figure 4.30: Rand index for pairwise comparison between the full data set using Jackknife distance measures and the 16 time point data set using the original distance measures for QT–Clust (left panel) and K–Means (right panel).

to tighter clusters with more similar gene expression profiles.

- The partitions on the original data set using Jackknife distance measures agree very well with the partitions on the smaller data set using the classical distance measures.
- The influence of the distance measure is much higher for QT–Clust than for K–Means.

Chapter 5

Implementation in R

All cluster algorithms and visualization methods used are implemented in R (R Development Core Team 2009), a free software environment for statistical computing and graphics.

5.1 flexclust

R package **flexclust** (Leisch 2006) is a flexible toolbox to investigate the influence of distance measures and cluster algorithms. It contains extensible implementations of the K–centroids and QT–Clust algorithm and offers the possibility to try out a variety of distance or similarity measures as cluster algorithms are treated separately from distance measures. New distance measures and centroid computations can easily be incorporated into cluster procedures. The default plotting method for cluster solutions in **flexclust** is the neighborhood graph.

Function `kcca` uses a family concept similar to the implementation of generalized linear models in S (Chambers and Hastie 1992). A KCCA family consists of the following two parts:

dist : A function taking N observations and K centroids as inputs and returning the NxK matrix of distances between all observations and centroids.

cent : An (optional) function computing the centroid for a given subset of the observations.

An example for a new distance measure is given by

```
distJackknife <- function(x, centers)
{
   m = array(dim=c(ncol(centers),nrow(x),nrow(centers)))
   for(i in 1:ncol(centers)){
     m[i,,] = distCor(x[,-i,drop=FALSE],centers[,-i,drop=FALSE])
   }
   apply(m,2:3,min)
}
```

For Jackknife Correlation a closed form for centroid computation does not exist up to our knowledge. If `cent` is not specified a general purpose optimizer is used (at some speed and precision penalty). The canonical centroids cluster–wise means and cluster–wise medians are used for Euclidean distance and for Manhattan distance. The KCCA family objects are also used for function `qtclust`.

5.2 flexmix

The EM algorithm for ML estimation of finite mixture models is for example implemented in the R package **flexmix** (Leisch 2004; Grün and Leisch 2008). The E-step is treated as fixed whereas arbitrary models can be fitted by modifying the M-step. For mixtures of linear mixed models `FLXMRlmer()` and for mixture of linear models with smoothing splines `FLXMRsmooth.spline()` are used as model drivers for the M-step.

5.3 gcExplorer

R package **gcExplorer** (Scharl and Leisch 2009a) is a new visualization toolbox for the interactive exploration of cluster solutions. **gcExplorer** is based on infrastructure from R package **flexclust** (Leisch 2006). Arbitrary centroid–based cluster solutions using for example the standard functions `kmeans` or `pam` can also easily be used as input by converting them to **flexclust** objects. Bioconductor packages **graph** and **Rgraphviz** (Gentleman et al. 2005) are used for non–linear arrangement of the graph nodes. **Rgraphviz** is an interface to the open source software project GraphViz (`http://www.graphviz.org`).

gcExplorer offers several possibilities for the interactive exploration of gene clusters. The functionality of the package includes the visualization of the cluster structure in form of neighborhood graphs, the display of gene clusters in graphics or HTML tables, highlighting additional properties of the clusters as well as test procedures to judge the quality of cluster solutions. There are different methods to include information about the clusters in the representation of nodes by using color coding defined by argument `node.function`. External information about pathways or functional groups (e.g., gene ontology terms) can be included in the call to `gcExplorer`. This allows to search for accumulation of functionally related genes in clusters. The association between clusters can be explored by using different cutoff values for the edges to be drawn. This can be used to find the appropriate number of clusters for a given problem.

The most important feature is that the data points in a given cluster can be explored interactively using panel functions. When clicking on the nodes of the neighborhood graph, the `panel.function()` is executed for the observations in the corresponding cluster. Users can define arbitrary new panel functions, or use the ones already defined in the package. Any R graphic can be used as panel function, the example below shows gene expression profiles. Another possibility are HTML tables of all genes in a cluster with links to databases.

5.3.1 Artificial data generator

R package **gcExplorer** contains functionality to generate arbitrary time course gene expression data. The idea is to start with a set of cluster centers which are created by some data generating process. The following data generating processes are possible:

- simulate from a normal distribution,
- simulate from an integrated autoregressive process, and
- manually define patterns.

The gene cluster simulator `gcSim()` is used as follows to generate the set of centers

```
cent <- gcSim(sim = "arima", time = 16, sd = 0.1,
              sd.ri = 0, size = 1, n = 15)
```

where

- `sim`: data generating process
- `time`: number of time points
- `sd`: SD of the measurement error

- `sd.ri`: SD of the RI
- `size`: number of genes in a cluster
- `n`: number of clusters

A set of centers can be used to form different data sets using various combinations of noise parameters, e.g.,

```
data1 <- gcData(
   gcSim(sim = "pattern", cent = cent,
         sd = 0.1, sd.ri = 0.1,
         size = rep(c(10, 20, 30, 50, 100),
         each = 3)),
   gcSim(sim = "noise", time = 16, size = 100))
```

where `cent` is the set of centers

5.3.2 Non–linear layout algorithms

A linear projection of the data into 2 dimensions using for example linear discriminant analysis (LDA) has the advantage that the lengths of edges in the graph are directly interpretable. However, LDA does not scale well in the number of clusters, and relationships between the centroids of more than 15 clusters can hardly be displayed in the plane. As shown in Scharl and Leisch (2008b) linear methods cannot be used for high–dimensional gene expression data and a large number of clusters. R package **gcExplorer** (Scharl and Leisch 2009a) uses non–linear layout algorithms implemented in the open source graph visualization software Graphviz (`http://www.graphviz.org/`) for the display of neighborhood graphs. Bioconductor packages **graph** and **Rgraphviz** (Carey et al. 2005) provide tools for creating, manipulating, and visualizing graphs in R as well as an interface to Graphviz. **Rgraphviz** returns the layout information for a graph object, x- and y–coordinates of the graph's nodes as well as the parameterization of the trajectories of the edges. Several layout algorithms can be chosen.

dot: hierarchical layout algorithm for directed graphs

neato and fdp: layout algorithms for large undirected graphs

twopi: radial layout

circo: circular layout

The default layout algorithm in **gcExplorer** is “dot”. Even though distances between nodes and length of edges are no longer interpretable when using non–linear layout algorithms the increase in readability and clear arrangement is obvious.

The latest release of **gcExplorer** is always available at the Comprehensive R Archive Network CRAN: `http://cran.R-project.org/package=gcExplorer`. Details on how to use the **gcExplorer** and how to perform the analysis described in Sections 5.4 and 5.5 can be found in the Appendix A (Scharl et al. 2009d) which will also be contained in the package as a vignette. Appendix B contains the help pages of the functions.

5.4 Exploratory analysis of cluster solutions using the gcExplorer

The PS19 data (described in Section 1.3) is now used to demonstrate the functionality of **gcExplorer**. The data is clustered using stochastic QT–Clust (Scharl and Leisch 2006b) yielding a cluster object which consists of 14 clusters.

In Figure 5.1 the cluster solution is displayed using LDA. Nodes correspond to cluster centroids and the thickness of an edge between two clusters is proportional to their similarity. Data points are displayed as well as cluster hulls. This is the display method provided in **flexclust**. For this data set linear projection of the data into two dimensions works quite well. However, the nodes corresponding to cluster centroids are already very small and will become even smaller for more than 14 clusters.

The neighborhood graph shown in Figure 5.2 uses the non–linear layout algorithm "dot" and allows a detailed view on the cluster structure even for a large number of clusters. Again the nodes in the graph correspond to cluster centroids and the shadow values between clusters defined in Section 2.3 are used as edge weights. The thickness of an edge between two clusters is proportional to their similarity. Related clusters are not forced to lie next to each other in the graph as edges can have various lengths. For example cluster 13 located at the right end of the graph is related to cluster 1 located in the top of the graph. Several groups of clusters can be found. The clusters in the bottom left corner of the graph (e.g., clusters 3, 6, 12 and 14) are not connected to the clusters in the right part of the graph (e.g., clusters 5, 9, 10 and 13) indicating that the corresponding genes show very different expression profiles over time.

5.4.1 Interactive exploration

An schematic view of the interactive usage of the **gcExplorer** is given in Figure 5.3. By clicking on the nodes of the neighborhood graph new graphics devices pop up showing the corresponding cluster by using the stated panel function. In this example clusters 3, 4, 7, 13 and 14 are visualized by plotting the corresponding gene expression profiles and cluster 3 is also displayed in form of an HTML table.

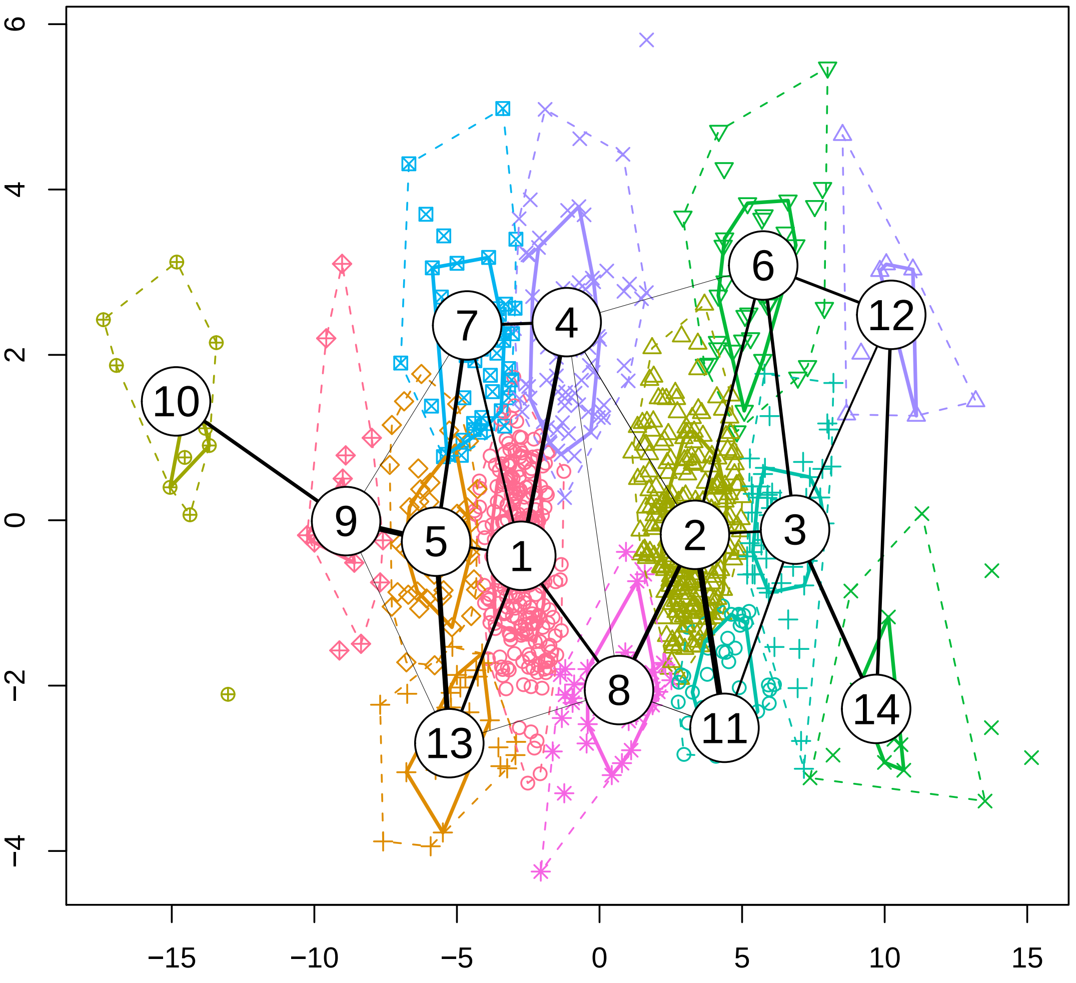


Figure 5.1: Neighborhood graph of a cluster solution of the PS19 data where nodes correspond to cluster centroids and the thickness of an edge between two clusters is proportional to their similarity. Data points are displayed as well as cluster hulls.

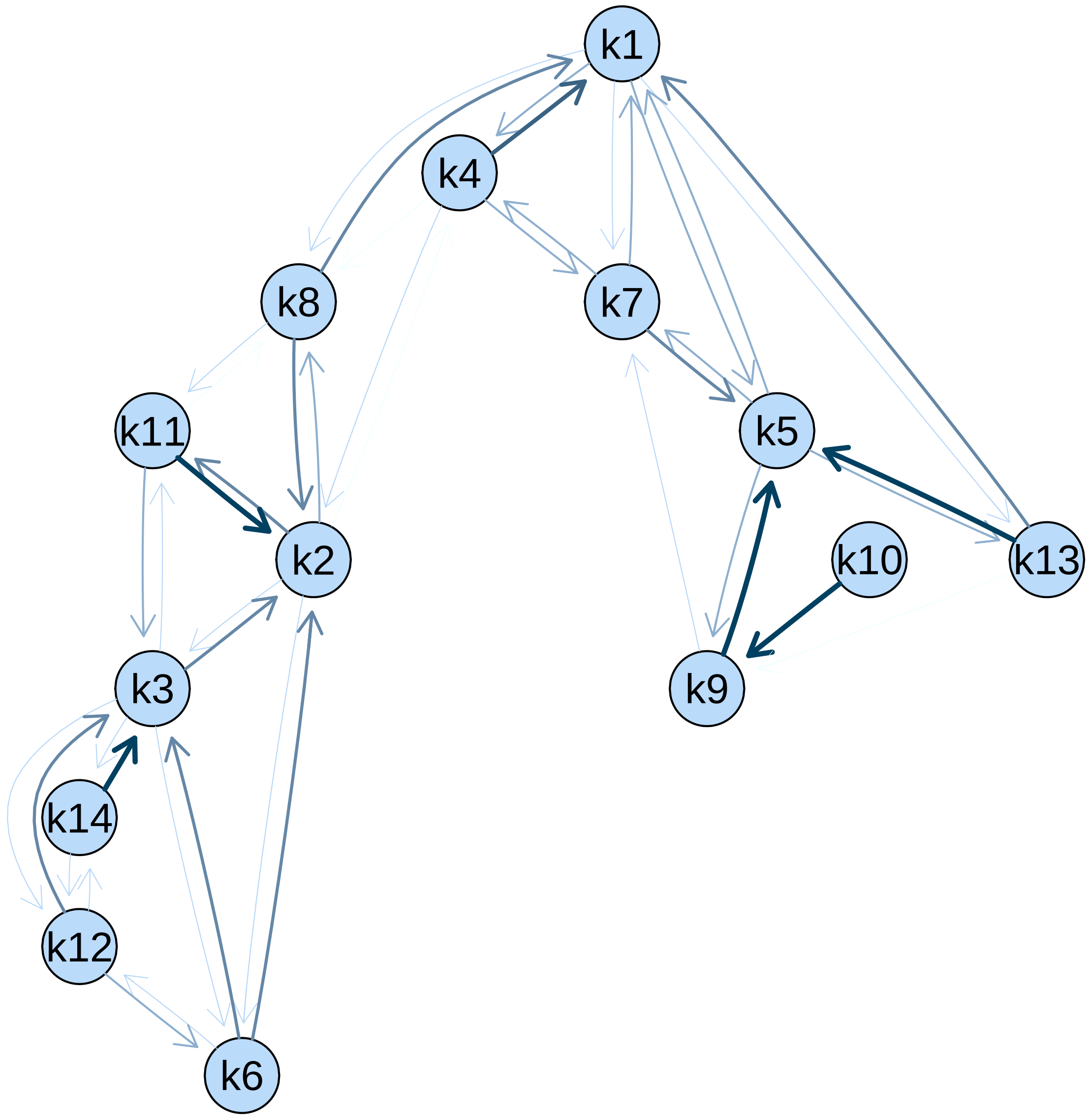
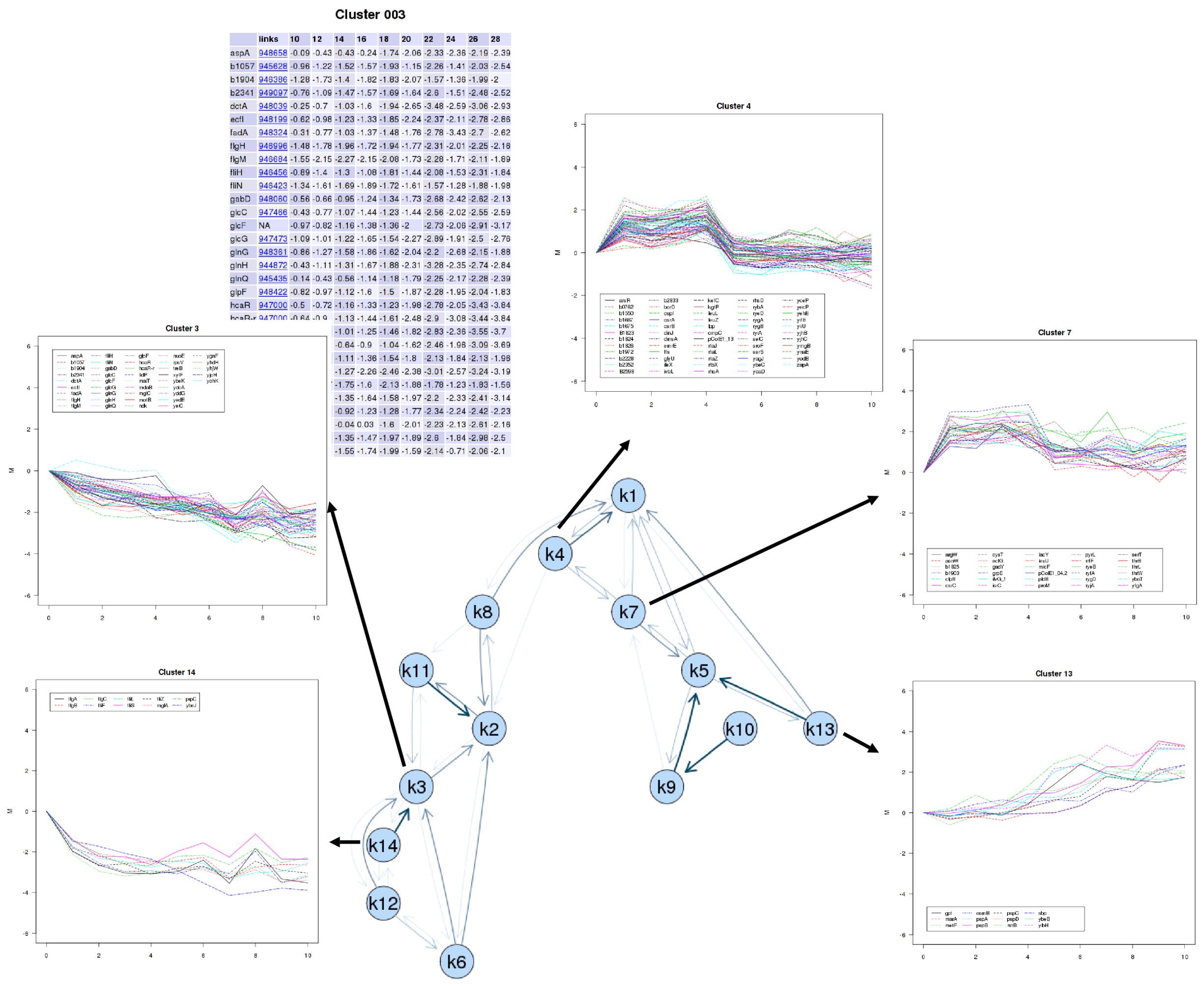


Figure 5.2: Neighborhood graph of a cluster solution of the PS19 data where nodes correspond to cluster centroids and the thickness of an edge between two clusters is proportional to their similarity.



Cluster 003

	links	10	12	14	16	18	20	22	24	26	28
aspA	948658	-0.09	-0.43	-0.43	-0.24	-1.74	-2.06	-2.33	-2.36	-2.19	-2.39
b1057	945628	-0.96	-1.22	-1.52	-1.57	-1.93	-1.15	-2.26	-1.41	-2.03	-2.54
b1904	946386	-1.28	-1.73	-1.4	-1.82	-1.83	-2.07	-1.57	-1.36	-1.99	-2
b2341	949097	-0.76	-1.09	-1.47	-1.57	-1.69	-1.64	-2.6	-1.51	-2.48	-2.52
dctA	948039	-0.25	-0.7	-1.03	-1.6	-1.94	-2.65	-3.48	-2.59	-3.06	-2.93
ecfI	948199	-0.62	-0.98	-1.23	-1.33	-1.85	-2.24	-2.37	-2.11	-2.78	-2.86
fadA	948324	-0.31	-0.77	-1.03	-1.37	-1.48	-1.76	-2.78	-3.43	-2.7	-2.62
flgH	946996	-1.48	-1.78	-1.96	-1.72	-1.94	-1.77	-2.31	-2.01	-2.25	-2.16
flgM	946684	-1.55	-2.15	-2.27	-2.15	-2.08	-1.73	-2.28	-1.71	-2.11	-1.89
fliH	946456	-0.89	-1.4	-1.3	-1.08	-1.81	-1.44	-2.08	-1.53	-2.31	-1.84
fliN	946423	-1.34	-1.61	-1.69	-1.89	-1.72	-1.61	-1.57	-1.28	-1.88	-1.98
gabD	948060	-0.56	-0.66	-0.95	-1.24	-1.34	-1.73	-2.68	-2.42	-2.62	-2.13
glcC	947466	-0.43	-0.77	-1.07	-1.44	-1.23	-1.44	-2.56	-2.02	-2.55	-2.59
glcF	NA	-0.97	-0.82	-1.16	-1.38	-1.36	-2	-2.73	-2.06	-2.91	-3.17
glcG	947473	-1.09	-1.01	-1.22	-1.65	-1.54	-2.27	-2.89	-1.91	-2.5	-2.76
glnG	948361	-0.86	-1.27	-1.58	-1.86	-1.62	-2.04	-2.2	-2.68	-2.15	-1.88
glnH	944872	-0.43	-1.11	-1.31	-1.67	-1.88	-2.31	-3.28	-2.35	-2.74	-2.84
glnQ	945435	-0.14	-0.43	-0.56	-1.14	-1.18	-1.79	-2.25	-2.17	-2.28	-2.39
glpF	948422	-0.82	-0.97	-1.12	-1.6	-1.5	-1.87	-2.28	-1.95	-2.04	-1.83
hcaR	947000	-0.5	-0.72	-1.16	-1.33	-1.23	-1.98	-2.78	-2.05	-3.43	-3.84
hcaR-r	947000	-0.64	-0.9	-1.13	-1.44	-1.61	-2.48	-2.9	-3.08	-3.44	-3.84
				-1.01	-1.25	-1.46	-1.82	-2.83	-2.36	-3.55	-3.7
				-0.64	-0.9	-1.04	-1.62	-2.46	-1.96	-3.09	-3.69
				-1.11	-1.36	-1.54	-1.8	-2.13	-1.84	-2.13	-1.98
				-1.27	-2.26	-2.46	-2.38	-3.01	-2.57	-3.24	-3.19
				-1.75	-1.6	-2.13	-1.88	-1.78	-1.23	-1.83	-1.56
				-1.35	-1.64	-1.58	-1.97	-2.2	-2.33	-2.41	-3.14
				-0.92	-1.23	-1.28	-1.77	-2.34	-2.24	-2.42	-2.23
				-0.04	0.03	-1.6	-2.01	-2.23	-2.13	-2.61	-2.16
				-1.35	-1.47	-1.97	-1.89	-2.6	-1.84	-2.98	-2.5
				-1.55	-1.74	-1.99	-1.59	-2.14	-0.71	-2.06	-2.1

Figure 5.3: Neighborhood graph with some of the clusters displayed using panel functions `gcProfile` and `gcTable`.

5.4.2 Node functions

Color coding

In the graph shown above one single kind of node symbol is used for all nodes. This way no information about the different clusters is revealed. There are several possibilities how to include additional information in the representation of nodes. The most simple method is to use color coding, e.g., to color nodes by size or tightness of the corresponding clusters. In this case the color of a node depends on the distribution of a certain property over all nodes where the maximum will get the darkest and the minimum will get the brightest color. Usually the smaller or tighter clusters are more interesting and can more easily be explored.

The percentage of genes in a cluster assigned to a functional group under investigation can also be used for color coding. The visualization of functional groups in the graph is not only a validation of the cluster method. It is also a very helpful tool for practitioners to quickly find subgroups of genes related to specific functions under study.

Some examples of color coding are shown in Figure 5.4. In the top left panel cluster size is highlighted, i.e., dark node symbols indicate large clusters and light node symbols indicate small clusters. In the top right panel cluster tightness is used where dark nodes correspond to tight clusters which usually correspond to groups of genes with clearly defined gene expression profiles. In the bottom left and right panels two functional groups are investigated. In the bottom left panel clusters with accumulation of σ_{32}–regulated genes are highlighted which are related to heat shock. In the bottom right panel the GO term “flagellar motility” is shown which is part of the biological process classification.

Flagellar motility is an example of a functional group where the corresponding genes have similar expression profiles and are therefore grouped into similar clusters (i.e., clusters 11, 3 and 14) which are connected by edges in the neighborhood graph. In the case of σ_{32}–regulated genes (bottom left panel) there is no clear relationship between the cluster solution and the functional group as the corresponding genes are located in various clusters.

Node symbols

The second option for adding further information to the display of the neighborhood graph is to use different graphical symbols for the representation of nodes. For that purpose **gcExplorer** makes use of R package **symbols** (Voglhuber (2008), `http://r-forge.r-project.org/projects/symbols`). **symbols** is based on Grid (Murrell 2006), a very

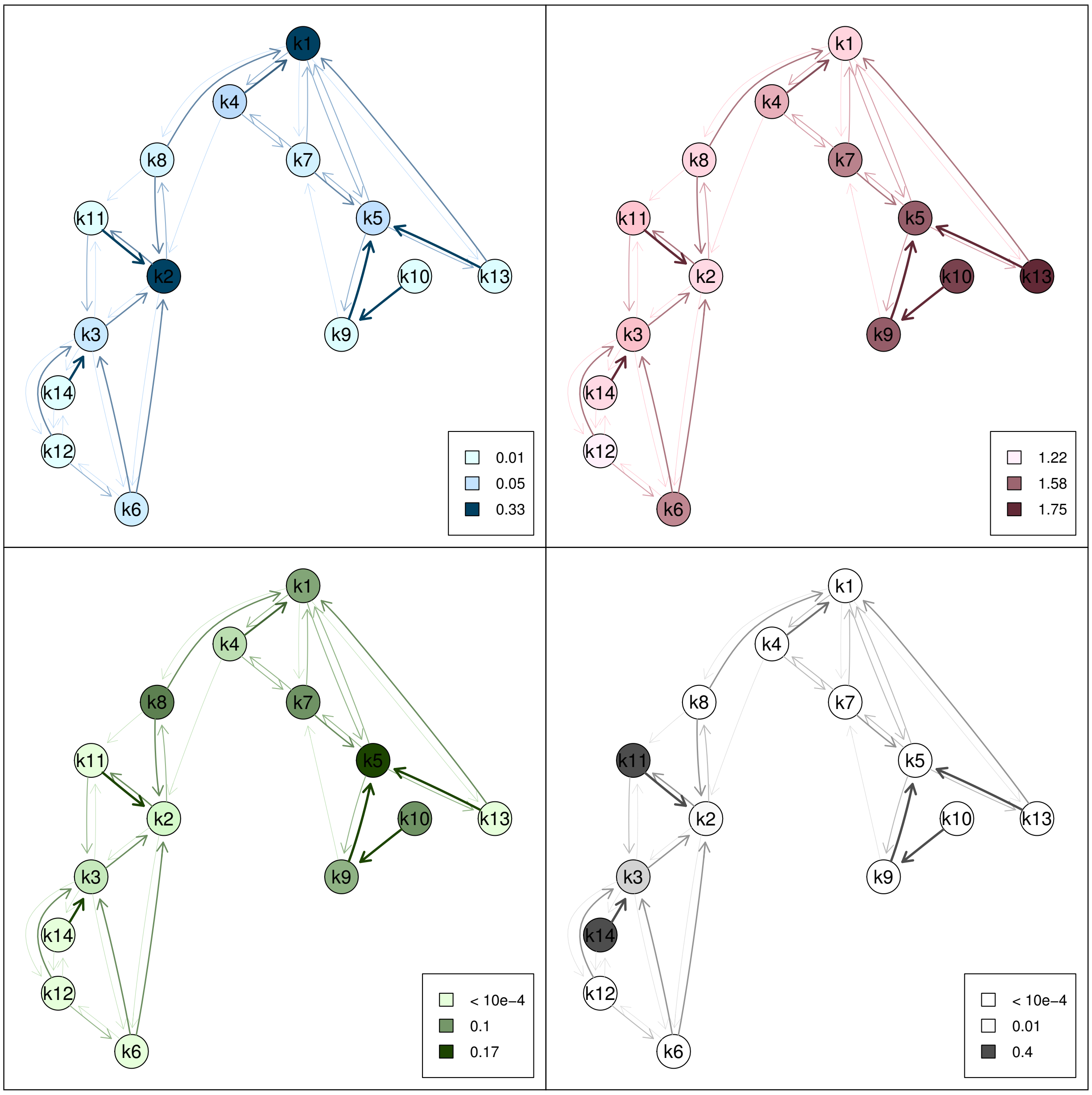


Figure 5.4: Different options for color coding. Top left panel: cluster size, top right panel: cluster tightness, bottom left panel: Sigma 32 regulated genes, bottom right panel: genes involved in flagellar motility.

flexible graphics system for R. Grid features viewports, i.e., rectangular areas allowing the creation of plotting regions all over the R graphic device. Due to the layout algorithms used in the **gcExplorer** nodes remain quite large allowing large viewports for the visualization of nodes. Several grid–based functions are implemented in package **symbols** which can directly be used as node functions in the **gcExplorer**.

The most natural node symbols in the case of time–course gene expression data are line plots showing the gene expression profiles over time for either the cluster centroids or the whole group of genes in a certain cluster. Figure 5.5 gives a very good overview of the cluster solution and the single gene clusters where similarities in gene expression profile can directly be investigated. It can be seen that clusters containing down–regulated genes are located in the bottom left part of the graph whereas up–regulated genes are located in the right part of the graph. Further, there are no edges between clusters of up- and down–regulated genes.

In order to visualize group memberships pie charts are frequently used. Figure 5.6 leftn panel shows the portion of genes with F statistic (F) > 20 and F ≤ 20 respectively. In the right panel of Figure 5.6 boxplots of the log F statistic are shown.

5.4.3 Edge options

Directed vs. undirected graph

The neighborhood graph is a directed graph as the similarity of cluster 1 to cluster 4 is different from the similarity of cluster 4 to cluster 1 and so on. Besides plotting the original directed graph there are several options how to plot edges taking into account for instance the mean, minimum or maximum of the similarities between two clusters. In practice the mean similarity is frequently used especially when testing the functional relationship between clusters (see Figure 5.9).

5.4.4 Graph modifications

The non–linear layout algorithms implemented in Graphviz are optimized for the given set of nodes and edges. Removing an edge or a node will result in a different graph which makes comparisons between graphs rather complicated. R package **gcExplorer** contains the function `gcModify` which allows to modify a given graph without changing the original layout. There are several possibilities how to modify a given graph. However, it is only possible to remove nodes and edges from a larger graph. Adding new nodes and edges is

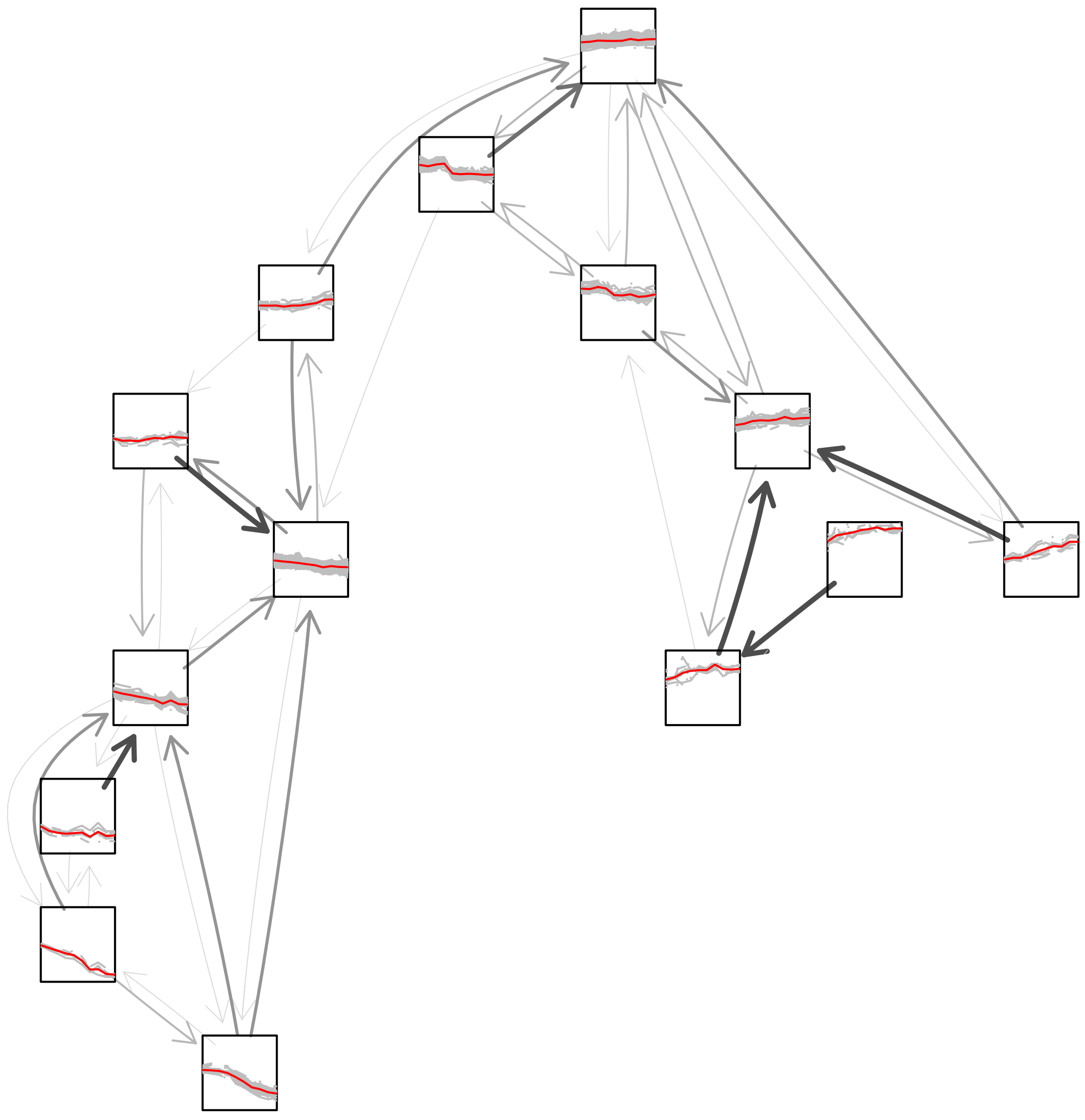


Figure 5.5: Neighborhood graph using line plots as node symbols where the genes expression profiles are plotted in grey and the cluster centroids are plotted in red.

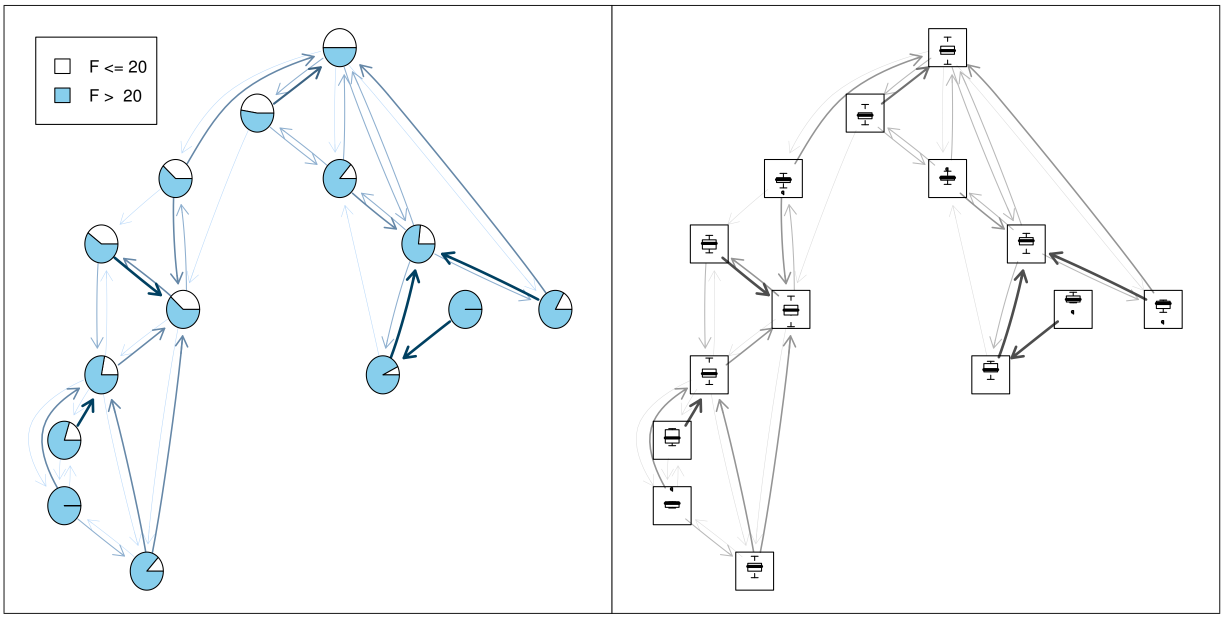


Figure 5.6: Neighborhood graph using pie charts (left panel) and boxplots (right panel) as node symbols.

not allowed. The node symbols are independent of the graph structure so different node functions can be used in each modified graph.

Node modifications

Sometimes only a subgraph of the original graph is of interest, e.g., clusters of all up–regulated genes. A subgraph can be created specifying either the set of nodes which should remain in the graph or by specifying the nodes which should be removed from the graph. In the next step manual or automatic zooming can be used to enlarge certain parts of the plot. An example of a subgraph is given in Figure 5.7.

Edge modifications

Filtering by cluster similarity can be used to simplify the original neighborhood graph. Edges between nodes are only drawn if the similarity between clusters is above a certain threshold, e.g., at least 10%. This prevents the graph from being too complex. Examples of the neighborhood graph where different cutoff values for drawing edges are shown are given in Figure 5.8.

Comparisons of different cutoff values as shown in Figure 5.8 are only possible when

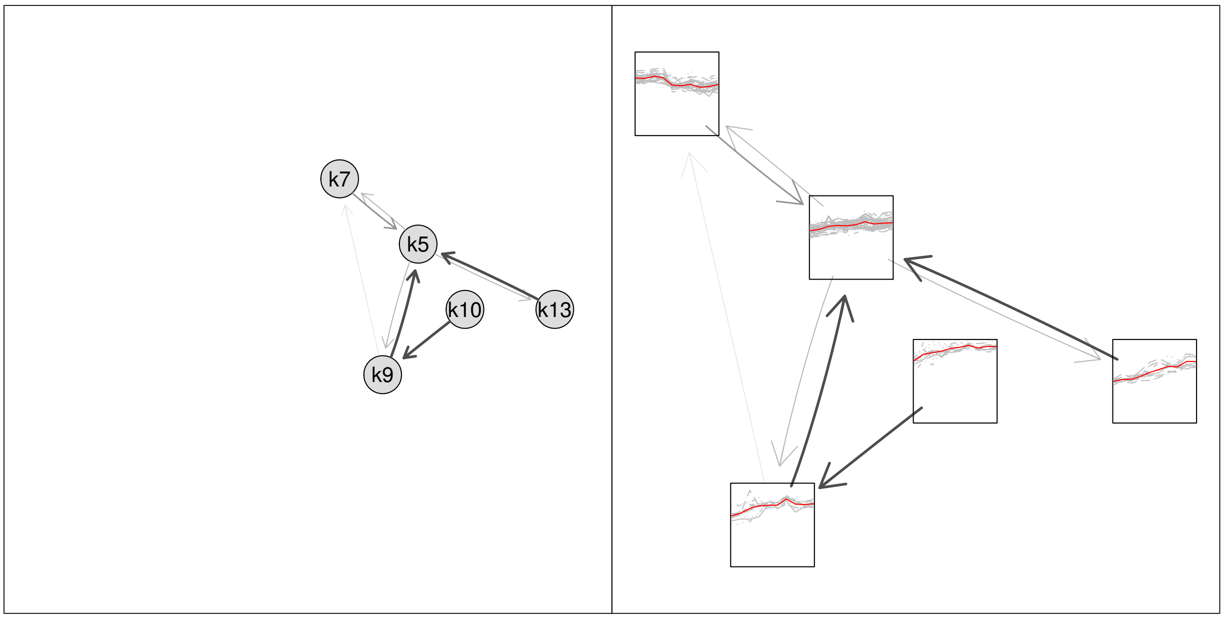


Figure 5.7: A subgraph of the neighborhood graph before zooming without specified node function (left panel) and after zooming with a node function (right panel).

starting with the largest set of edges.

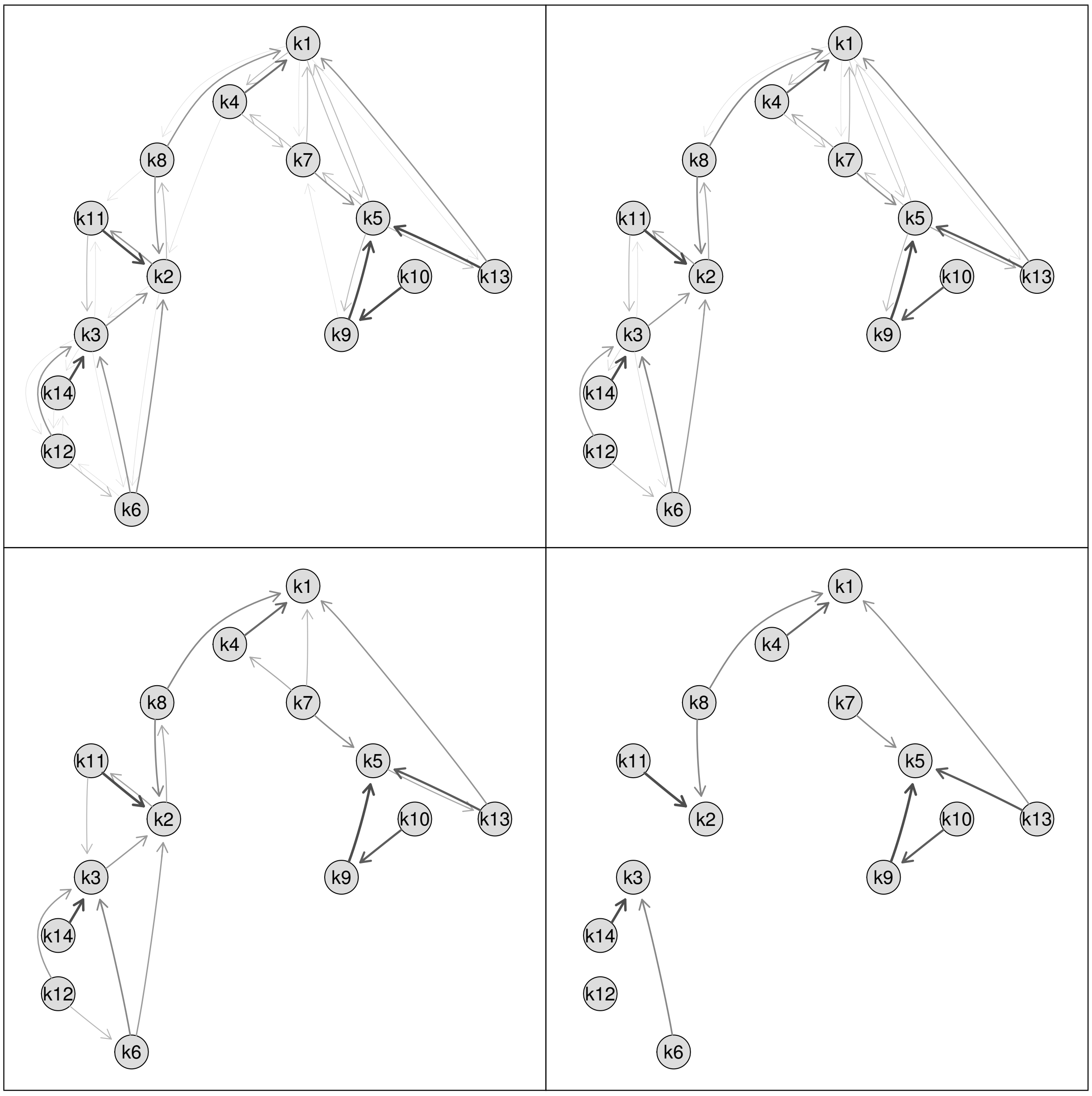


Figure 5.8: Use of different cutoff values for drawing edges in the neighborhood graph. Top left panel: all edges, top right panel: similarity > 10%, bottom left panel: similarity > 20%, bottom right panel: similarity > 30%.

5.5 Inferential analysis of cluster solutions with the gcExplorer

5.5.1 Compare cluster solutions

In this section the goodness of the cluster solution of the PS19 data investigated in Section 5.4 is judged based on its validity when applied to the PS17 experiment (see Section 1.3) where the same set of genes was exposed to different experimental conditions. Table 5.1 gives the results of the `comp_test` described in Section 3.3.2 consisting of cluster size, observed average within cluster distance, the 5% quantile of the permuted average distances and the probability of observing a lower within cluster distance by randomly assigning the genes to clusters. In this case 10 out of 14 clusters have a significantly smaller within cluster distance when using the cluster solution of the PS19 experiment compared to random assignment. In other words these 10 groups of genes form clusters under different experimental conditions and are more likely to contain co–regulated genes.

Table 5.1: Judge the validity of the PS19 cluster solution for the PS17 data using the comp_test.

	size	obs.av.dist	5%quantile.perm	p.val.lower
1	302	0.58	0.95	**0.00**
2	299	0.55	0.94	**0.00**
3	41	0.65	0.83	**0.00**
4	59	0.62	0.85	**0.00**
5	52	0.73	0.84	**0.00**
6	31	0.61	0.79	**0.00**
7	30	0.66	0.78	**0.00**
8	26	0.82	0.77	0.10
9	14	0.52	0.68	**0.00**
10	10	0.38	0.62	**0.00**
11	10	0.70	0.63	0.12
12	5	0.49	0.45	0.07
13	12	0.96	0.66	0.53
14	10	0.62	0.63	**0.04**

5.5.2 Functional relevance test

For the functional relevance test presented in Section 3.3.1 another *E. coli* experiment was used where various mutants were investigated under oxygen deprivation (Covert et al. 2004). The mutants were designed to monitor the response from *E. coli* during an oxygen shift in order to target the a priori most relevant part of the transcriptional network by using six strains with knockouts of key transcriptional regulators in the oxygen response. These experiments provide expression profiles for 4205 genes derived from the original data set downloaded from the Gene Expression Omnibus (Barrett et al. 2007) with accession GDS680 by applying the altering steps described in Castelo and Roverato (2009).

Another possibility for external validation of a cluster solution is to test the functional relevance of single edges, i.e., to test the relationship between a functional grouping and a cluster solution. In this example the *E. coli* oxygen data set (Covert et al. 2004) is used and the GO term GO:0009061 (anaerobic respiration) is investigated. The accumulation of genes involved in anaerobic respiration is displayed in Figure 5.9 left panel. In the case of edge tests undirected graphs are used instead of the original directed graphs as each pair of nodes is only tested once.

The output of function `edgeTest` (see Table 5.2) gives detailed information about the tested edges, i.e., the corresponding cluster sizes, the difference in proportions and the p–value. Additionally, function `edgeTest` gives the 95% quantile of the maxima of the permuted average distances which is 0.22 in this case. The p-values are now used to form a new similarity matrix using function `newclsim`. If the p–value of an edge is smaller than 0.05 the edge weight is set to 0. This new similarity matrix based on the p–values of the functional relevance test is finally used to draw a modified neighborhood graph where significant edges are removed. In this case 11 edges have significant p–values and differences in proportions larger than 0.23. In Figure 5.9 right panel the modified neighborhood graph is displayed. It can be seen that clusters 32, 43, 36, 34, 21 and 22 contain most of the genes involved in anaerobic respiration and form a disconnected subgraph after testing the functional relevance of the edges.

5.5.3 Power simulations for the functional relevance test

The power of the functional relevance test is simulated on artificial cluster solutions. For defined

- datasize

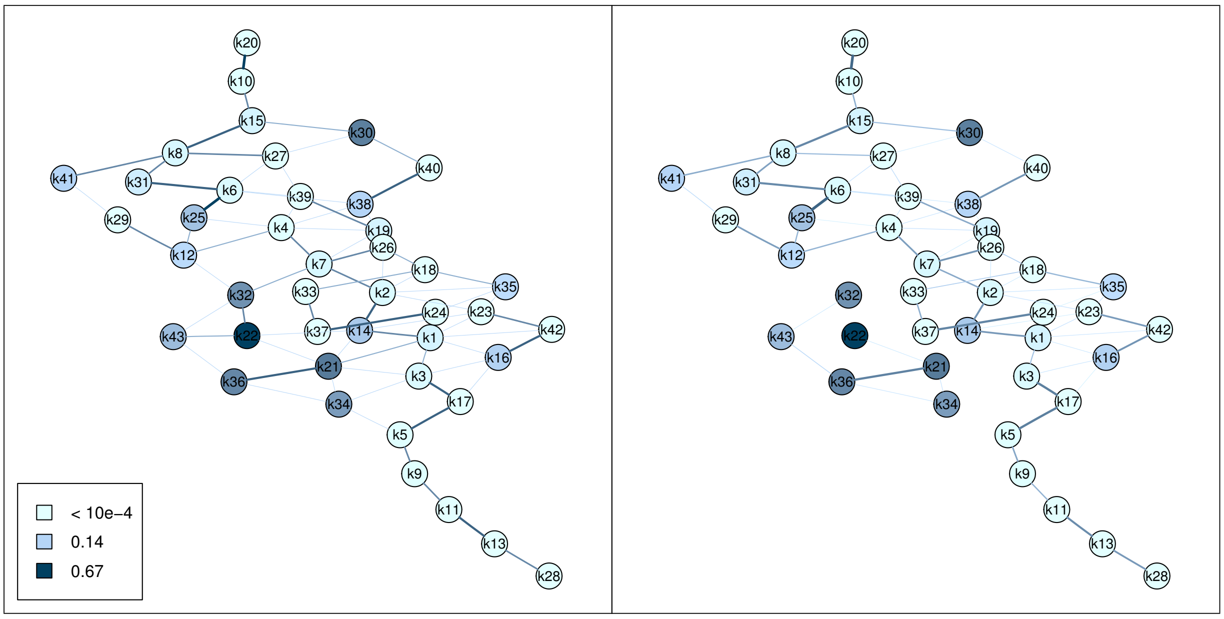


Figure 5.9: Left Panel: Neighborhood graph of the oxygen data set where the mean edge method is used. Right Panel: Neighborhood graph where significant edges are removed using the functional relevance test.

- number of clusters
- difference in proportions between cluster 1 and 2
- proportion of grouped genes in cluster 1
- proportion of grouped genes in the total data set

a cluster solution is simulated where the difference in proportions between clusters 1 and 2 is fixed and the remaining proportions are random. For a given setup the functional relevance test is run 1000 times where only the power for the edge between clusters 1 and 2 is observed (see Table 5.3). It can be seen that the test performs best if the proportion of grouped genes in cluster 1 is large and the proportion of grouped genes in the total data set is small.

Table 5.2: Functional relevance test of the *E. coli* oxygen data for functional group GO:0009061 (anaerobic respiration). The 95% quantile of the maxima of the permuted average distances is 0.22.

	Clsize1	Clsize2	Diff.in.Prop.	P-value
1˜2	671	526	0.02	1.00
1˜3	671	424	0.01	1.00
4˜6	378	209	0.02	1.00
2˜7	526	121	0.01	1.00
4˜7	378	121	0.02	1.00
6˜8	209	108	0.01	1.00
4˜12	378	16	0.11	0.59
1˜14	671	33	0.14	0.51
2˜14	526	33	0.16	0.50
1˜16	671	13	0.11	0.59
3˜16	424	13	0.12	0.57
1˜21	671	9	0.40	**0.00**
3˜21	424	9	0.41	**0.00**
14˜21	33	9	0.26	**0.05**
14˜22	33	12	0.48	**0.00**
21˜22	9	12	0.22	0.13
4˜25	378	10	0.19	0.29
6˜25	209	10	0.17	0.34
12˜25	16	10	0.08	0.93
2˜32	526	11	0.34	**0.01**
7˜32	121	11	0.33	**0.03**
12˜32	16	11	0.24	**0.05**
22˜32	12	11	0.30	**0.03**
3˜34	424	6	0.30	**0.03**
5˜34	263	6	0.33	**0.03**
21˜34	9	6	0.11	0.77
2˜35	526	17	0.09	0.81
21˜36	9	5	0.04	1.00
34˜36	6	5	0.07	0.94
22˜43	12	9	0.44	**0.00**
32˜43	11	9	0.14	0.51
36˜43	5	9	0.18	0.33

Table 5.3: Power simulations for the functional relevance test using differences in proportion between 0.05 and 0.4. The number of clusters is 10.

Data size	prop.c1	prop.all	d 0.05	d 0.1	d 0.15	d 0.2	d 0.25	d 0.3	d 0.35	d 0.4
100	0.50	0.50	0	0.000	0.000	0.000	0.004	0.043	0.062	0.108
100	0.50	0.33	0	0.000	0.000	0.000	0.010	0.044	0.095	0.179
100	0.50	0.25	0	0.000	0.000	0.000	0.011	0.074	0.129	0.229
100	0.50	0.20	0	0.000	0.000	0.001	0.018	0.078	0.186	0.300
100	0.33	0.50	0	0.000	0.001	0.005	0.033	0.051	0.033	0.029
100	0.33	0.33	0	0.000	0.000	0.006	0.035	0.068	0.071	0.044
100	0.33	0.25	0	0.000	0.000	0.013	0.049	0.065	0.074	0.062
100	0.33	0.20	0	0.000	0.001	0.020	0.064	0.087	0.088	0.080
500	0.50	0.50	0	0.000	0.010	0.084	0.276	0.653	0.999	1.000
500	0.50	0.33	0	0.000	0.015	0.137	0.442	0.918	1.000	1.000
500	0.50	0.25	0	0.000	0.010	0.180	0.606	0.996	1.000	1.000
500	0.50	0.20	0	0.000	0.025	0.248	0.700	1.000	1.000	1.000
500	0.33	0.50	0	0.001	0.026	0.159	0.384	0.747	0.764	0.450
500	0.33	0.33	0	0.001	0.069	0.242	0.551	0.978	0.889	0.669
500	0.33	0.25	0	0.002	0.074	0.301	0.733	1.000	0.909	0.905
500	0.33	0.20	0	0.000	0.098	0.414	0.903	1.000	0.935	0.976

Chapter 6

Applications of cluster methods to *E. coli* data

6.1 Comparison of induction strategies

In this section the utility of the interactive visualization toolbox **gcExplorer** is demonstrated for the interpretation of *E. coli* microarray data (Scharl et al. 2009b). The data sets used derive from two independent fedbatch experiments conducted in order to investigate the impact of different induction strategies on the host metabolism and product yield. The goal of the comparison is to identify genes and pathways that act similar in both settings and more importantly to identify groups of genes with differential reaction to the two induction strategies. For this reason cluster analysis followed by comparative graphical investigation of the different groups of genes is performed. The graphical exploration of clusterings is applicable to arbitrary partitioning cluster solutions. In this case the stochastic quality cluster algorithm QT–Clust (Scharl and Leisch 2006b) is used. The data sets used are described in Section 6.1.1. In the following several steps of the analysis of the given data sets are presented including the visualization of the cluster structure and the direct graphical comparison of these two experiments. It is shown that the identification of potentially interesting gene candidates or functional groups is substantially accelerated and eased.

6.1.1 Data

The *E. coli* cultivation data were collected at the Department of Biotechnology at the University of Natural Resources and Applied Life Sciences in Vienna. Two recombinant

E. coli processes with different induction strategies were conducted in order to evaluate the influence of the expression level of the inclusion body forming protein N^{pro}GFPmut3.1 on the host metabolism. The standard strategy with a single pulse of inducer yielding in a fully induced system (in the following called experiment A or PS17) was compared to a process with continuous supply of limiting amounts of inducer resulting in a partially induced system (in the following called experiment B or PS19) (Striedner et al. 2003). The time point of induction of the partially induced system was set one doubling past feed start. The bioreactor, the used equipment as well as the on- and offline analysis was published in detail by Achmüller et al. (2007). The resulting process data shown in Figure 6.1 clearly emphasize the central impact of induction strategies on the cellular response of strong expression systems and their behavior in production processes. The product formation rate triggered by full induction is too high and the thereby provoked metabolic overload impedes cellular growth. The increase in the total cell dry weight (CDW) attained past induction was mainly caused by the formation of the recombinant protein. This means that growth and product formation were decoupled completely. In consequence of these reactions product formation and process control were maintained only for a short period. However, in the experiment with limited induction cells were able to cope with the metabolic load triggered by the recombinant gene expression level for more than one doubling. Product formation was tightly coupled to cellular growth but approximately 9 hours past induction the metabolic load level exceeded the cellular capacities. The glucose yield coefficient ($Y_{X/S}$) decreased and the cells lost their ability to divide. The net cell mass generated in this phase was channeled into cell size and the cells entered a similar state as in the process with full induction.

In order to analyze the cellular response to different induction strategies on the transcription level two independent DNA microarray experiments were performed. A dye–swap design was used and the cells in the non-induced state of each experiment were compared to samples past induction. Since the production period of the fully induced system was limited to approximately one generation (7h at a growth rate of $0.1h^{-1}$) samples were drawn in a frequency of $1h^{-1}$. To cover the production period of the process with limited induction the sampling frequency was reduced to one sample every two hours. The used microarrays were epoxide-coated slides (Corning® Epoxide Coated Slides) with selective probes (50-mer oligos) for all 4289 open reading frames of the *E. coli* K12 genome (MWG *E. coli* K12 V2 oligo set; MWG Biotech AG, Germany) spotted in duplicates. The two experiments (including all processing protocols) have been loaded into ArrayExpress (`http://www.ebi.ac.uk/microarray-as/ae/`). The ArrayExpress accession number of

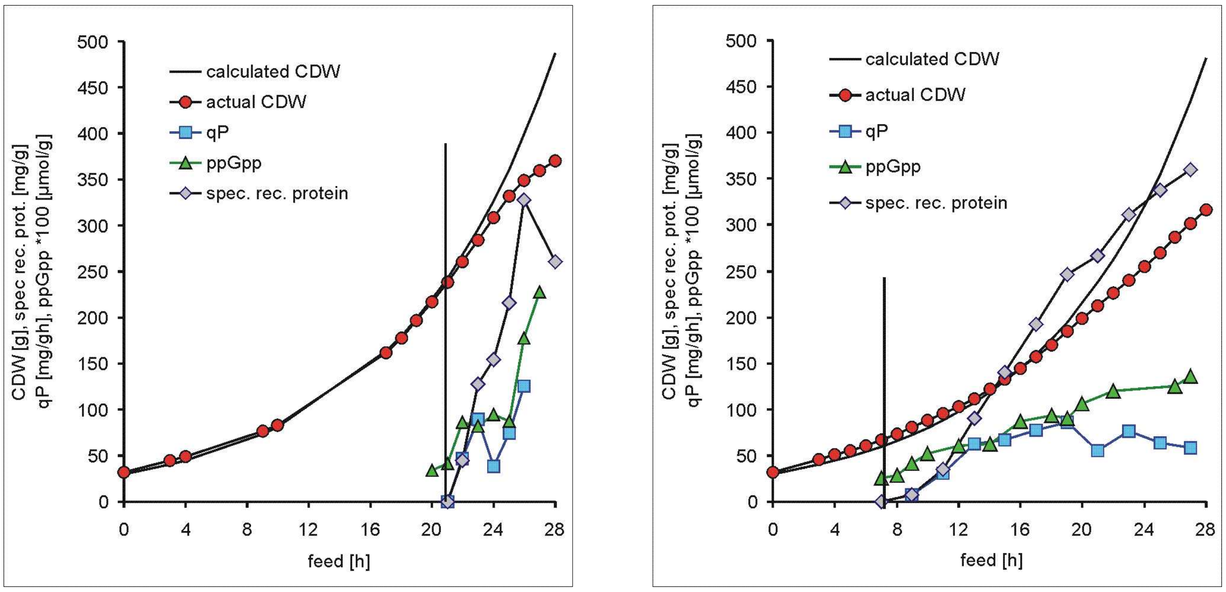


Figure 6.1: Protein production process with *E. coli* HMS174(DE3)(pET30aNProGFP). Fully induced system (experiment A, left panel) and partially induced system (experiment B, right panel). Vertical line indicates time point of induction.

the array design is A-MARS-10. The experiment with fully induced *E. coli* expression system (experiment A) has accession number E-MARS-16 and the experiment with partially induced system (experiment B) has accession number E-MARS-17. For standard low level analysis the data were preprocessed using print–tip loess normalization. Differential expression estimates were calculated using Bioconductor (Gentleman et al. (2005), `http://www.bioconductor.org`) package **limma** (Smyth 2005). The two data sets were filtered by excluding genes expressed at a very low level (average $\log_2$ intensity smaller 8), genes not showing differential expression (log–ratio M smaller ± 1.5) at least at one time point and genes with p-value of the corresponding F-statistic smaller 0.05. After filtering the data acquired from the experiment with a fully induced *E. coli* expression system (experiment A) consists of 733 genes and the data acquired from the process with limited induction (experiment B) consists of 429 genes where 311 genes are differentially expressed in both experiments. The filtered data sets were clustered using stochastic QT–Clust and further analysis and visualization was conducted using the **gcExplorer**.

6.1.2 Cluster visualization and interpretation

The major goal of this study is to identify differences between two independent microarray experiments which cannot be compared directly. For this purpose the two data sets are

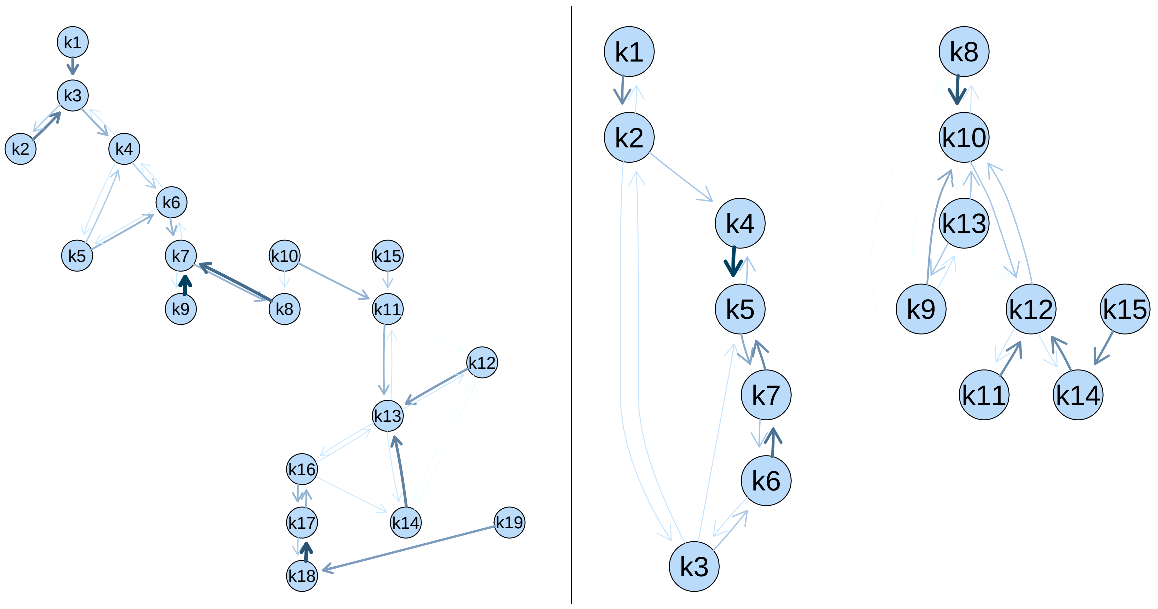


Figure 6.2: Neighborhood graph of the QT–Clust cluster solutions for experiment A (left panel) and experiment B (right panel).

clustered into small and tight subgroups of genes with common expression pattern which can easily be investigated. The diameter of the clusters is tuned in such a way to get in the range of 15 clusters and 10 outliers. The minimum number of points that form a single cluster is set to 2. These parameter settings lead reasonable cluster solutions that can directly be interpreted. The data sets of experiments A and B were separated into 19 and 15 clusters respectively with 20 and 9 outliers. Next these two cluster solutions are investigated independently and combined in the following section. In case of very similar clusters the neighborhood graph can be used to combine the clusters after proofing the similarity. However, in this exploratory approach it is advantageous to merge similar clusters than to split large ones.

The resulting cluster solutions are visualized as neighborhood graphs in Figure 6.2 using the **gcExplorer** where nodes correspond to cluster centroids. In the two graphs relationships between clusters can easily be explored as similar clusters are connected by edges. The thicker and darker an edge is drawn the more similar two clusters are. Several groups of clusters can be found. In the neighborhood graph of experiment A the clusters in the top left corner (e.g., 1,2,3) are not connected to the clusters in the bottom right corner (e.g., 17,18,19) indicating that the corresponding genes show very different expression profiles. This can be confirmed by looking at the expression profiles of the corresponding genes of experiment A (see Figure 6.3).

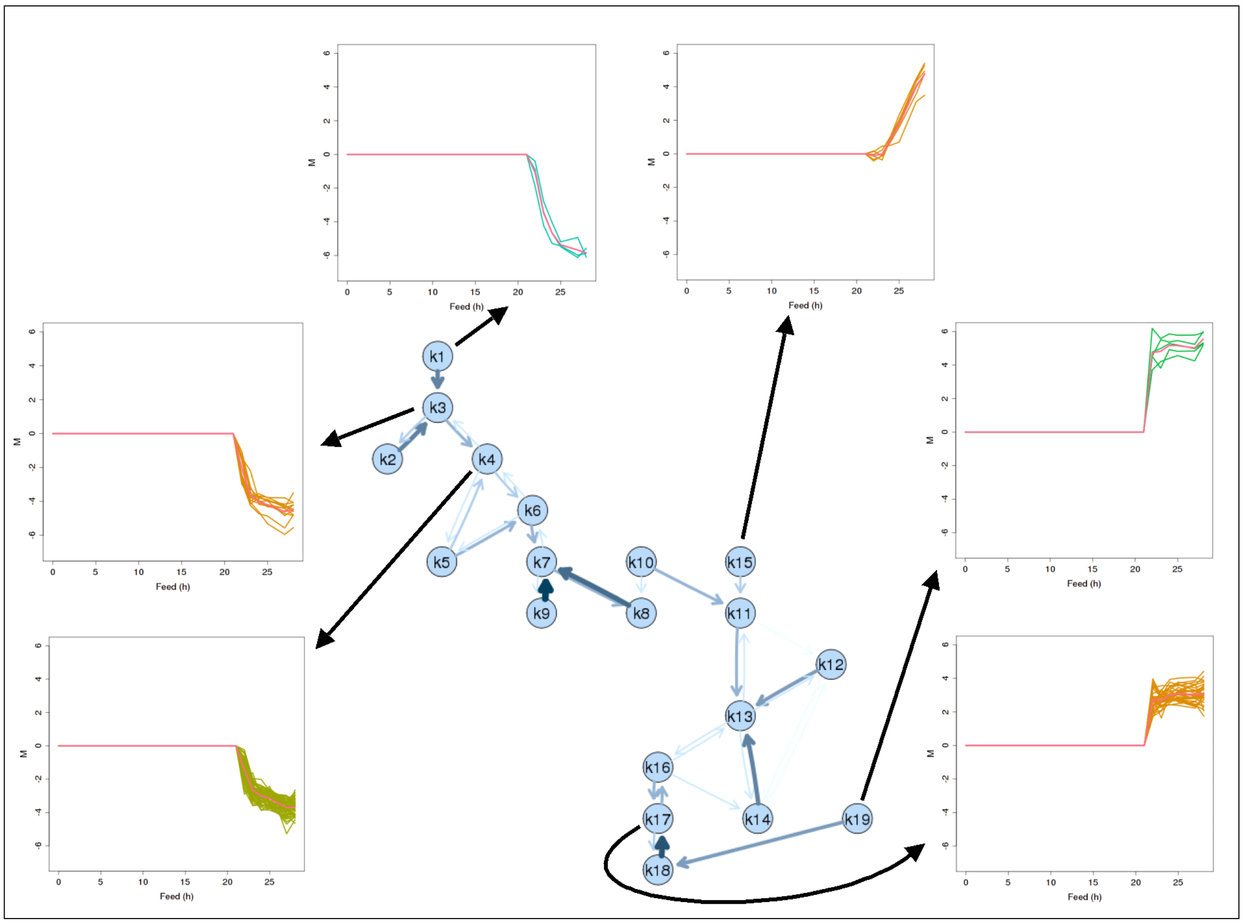


Figure 6.3: Neighborhood graph of experiment A with selected gene expression profiles displayed.

The genes in the bottom right clusters are all up–regulated (e.g., clusters 17 and 19) whereas the genes in the top left clusters are down–regulated (e.g., clusters 1, 3, and 4). The obtained results clearly show that the information gain of this work benefits from splitting the data sets in many small clusters at the beginning. For example, cluster 17, 18 and 19 contain genes with similar expression profiles. However, the level of up–regulation is much higher in cluster 19. If interpretation of a general trend is required these small clusters can be treated as a large one as it is often easier to investigate the smaller ones.

The cluster profiles with immediate and stern up or down regulation followed by constant values for the rest of the process definitely reflect the macroscopic outcome of the experiment with full induction. The irreversibility of the cellular response to the applied load level is mirrored in the transcriptome data. The only exception are the transcription profiles of genes related to phage shock grouped in cluster 15 which show continuously increasing gene expression until the end of the process.

A more detailed view on the cluster solution of experiment B is given in Figure 6.4. The neighborhood graph of this cluster solution consists of two unconnected subgraphs and shows a higher degree of differentiation.

The subgraph on the left contains all down–regulated genes whereas the subgraph on the right contains all clusters with up–regulated genes. Beside clusters with direct response to induction (e.g cluster 2 and 15) additional cluster profiles with immediate up–regulation followed by a downregulation after 9 hours past induction (e.g. cluster 9 and 13) or profiles with a 9 hour–delayed response to induction (e. g. cluster 6 and 3) were obtained. Furthermore, a large number of genes belongs to clusters with continuously increasing or decreasing trends past induction. These findings distinctly contradict the results of experiment A where only few genes show such a behavior. Again, the transcription data precisely reproduce the major changes in the experiment, the induction and the incipient metabolic overload.

The new visualization toolbox offers various possibilities for the analysis of microarray data which cannot all be shown here. In the graphs shown so far simple node symbols are used including the number of the corresponding cluster but there are several possibilities how to include additional information in the representation of nodes. The most simple method is to use color coding, e.g., to color nodes by size or tightness of the corresponding clusters. Another possibility is to use different shapes or symbols for nodes representing clusters with specific properties. The neighborhood graph is implemented in an interactive way and gene clusters can be investigated by clicking on the nodes. Plots of the expression profiles of the corresponding genes pop up and HTML tables giving further information

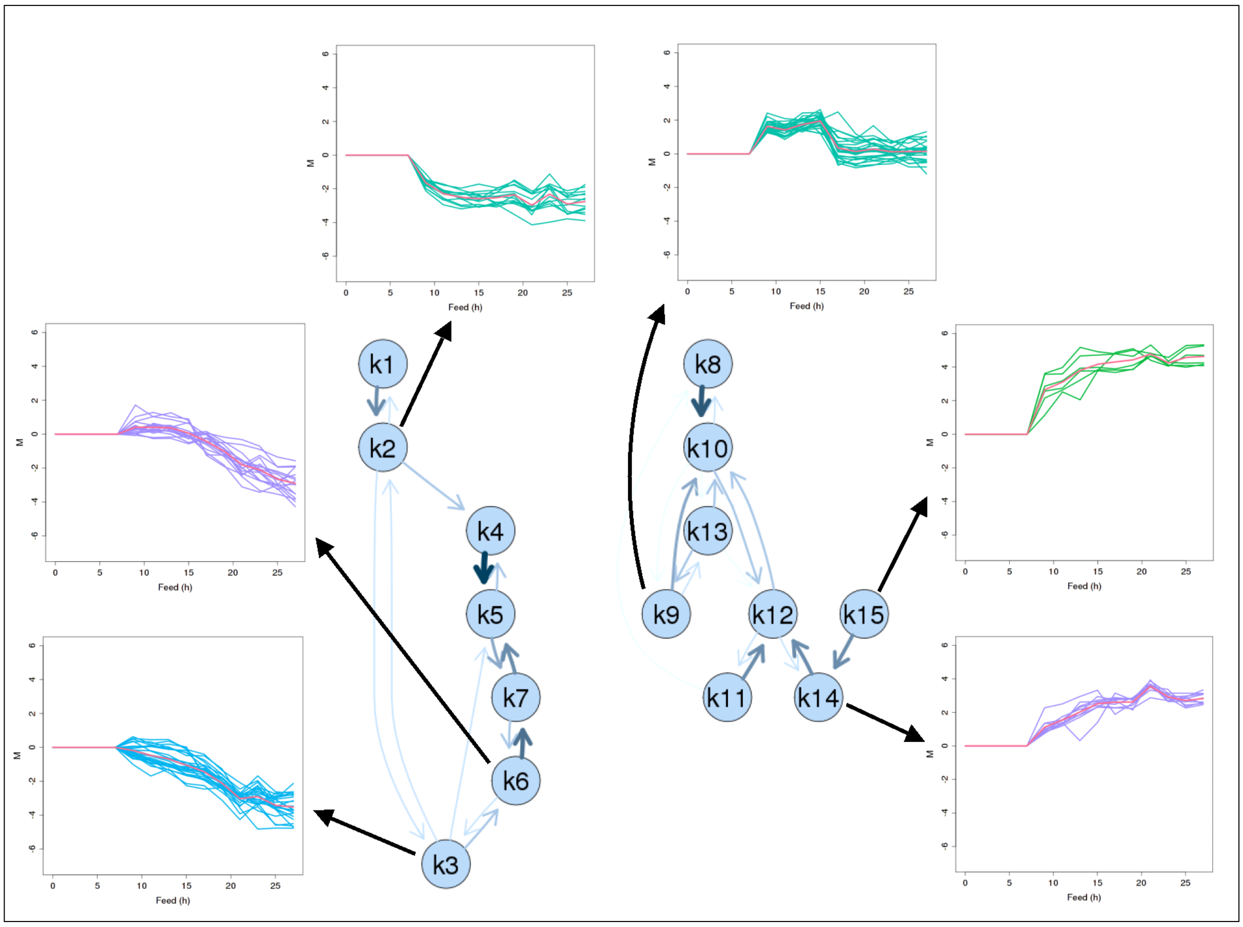


Figure 6.4: Neighborhood graph under limited conditions (experiment B) with selected gene expression profiles displayed.

about the genes link to databases like Ecocyc (`http://ecocyc.org/`). The **gcExplorer** is applicable up to a very high number of clusters. Related clusters are not forced to lie next to each other in the graph as edges can have various lengths (e.g., the edge between clusters 18 and 19 in the left panel or the edge between clusters 2 and 3 in the right panel).

6.1.3 Functional grouping

Cluster analysis is used to find groups of co–regulated genes in the microarray data without prior knowledge about the gene functions. However, by clustering expression profiles of co–expressed genes groups of genes with similar function are found. External information about the annotation of genes to functional groups can easily be included in the neighborhood graph, e.g., the accumulation of gene ontology (GO, The Gene Ontology Consortium (2000)) classifications in certain gene clusters can be highlighted in the node representation. For *E. coli* GO classifications about biological process (GOBP), molecular function (GOMF) and cellular component (GOCC), the GenProtEC (Serres et al. (2004), `http://genprotec.mbl.edu/`) classification system for cellular and physiological roles of *E. coli* gene products and the RegulonDB (Salgado et al. (2006), `http://regulondb.ccg.unam.mx/`) providing information about operons and regulatory networks were implemented. These knowledge–based functional mappings can be used to study cellular functions in individual clusters.

In the left panel of Figure 6.5 clusters of experiment A with genes controlled by σ^{32}, the main regulator of heat shock response are highlighted. In the right panel gene expression profiles of the closely related clusters 16 and 17 are displayed. 21 of 66 genes of the two clusters are under control of σ^{32}. Further functional characterization of these two clusters using GOMF yields the assignment of 26 genes to the GO–term GO:0005515 (protein binding) and of 16 genes to GO:0005524 (ATP binding). GOBP maps 11 genes to GO:0006950 (response to stress) and 10 genes to GO:0006457 (protein folding). On the other hand, a considerable number of 18 genes of these clusters is not mapped by the GO classification system as their molecular function is unknown or uncertain. Their cluster membership provides hints how these genes are embedded in the regulatory network of the cell and suggests potential cellular functions. A good example is *ybb*N, a thioredoxin–like protein with chaperone properties recently demonstrated in in–vitro experiments (Caldas et al. 2006; Kthiri et al. 2008; Soni et al. 2007). The relevance of the thus determined properties for cell physiology is still unknown but the cluster result strongly supports the suggested function as chaperone. Construction of a *ybb*N deletion mutant, a clone with

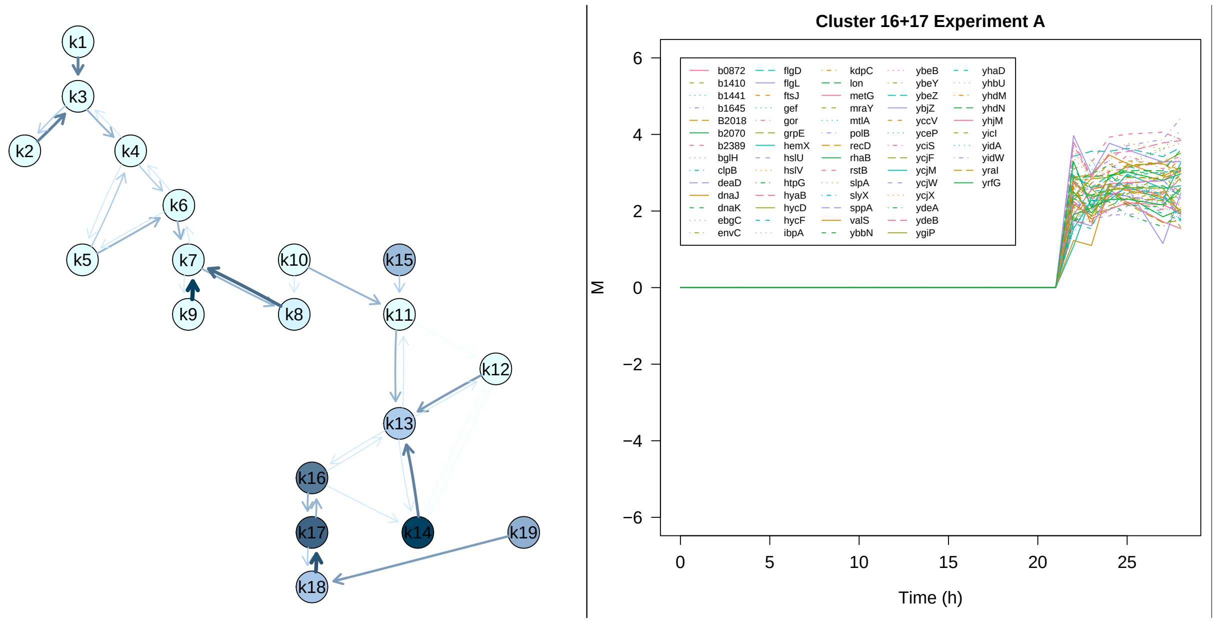


Figure 6.5: Neighborhood graph of experiment A where clusters containing σ^{32}–regulated genes are highlighted (left panel). Gene expression profiles of the corresponding genes of clusters 16 and 17 are shown in the right panel.

plasmid encoding *ybb*N and conduction of experiments similar to the described cultivations will provide the information which is required to confirm these assumptions.

6.1.4 gcExplorer - a tool for comparative graphical analysis of microarray experiments

One typical application of the **gcExplorer** is the comparative graphical analysis of different and independent μ–array experiments. It is exemplified in the following workflow. A cluster solution of a single experiment (e.g., experiment A) can easily be compared to other experiments (e.g., experiment B) in order to find genes or groups of genes with similar as well as different behavior. This is achieved by clustering the genes of experiment A and using this partition to investigate experiment B. This procedure helps to quickly identify groups of genes that cluster in both experiments and on the other hand to reveal differences between the experiments. An example of a gene cluster which is very similar between the two experiments is shown in the top panels of Figure 6.6. In the top left panel cluster 15 of experiment A is shown for the full induction data. In the top right cluster the same set of genes is shown under limited conditions. An example of a tight cluster of genes showing a strong and direct down–regulation in response to induction is given by cluster

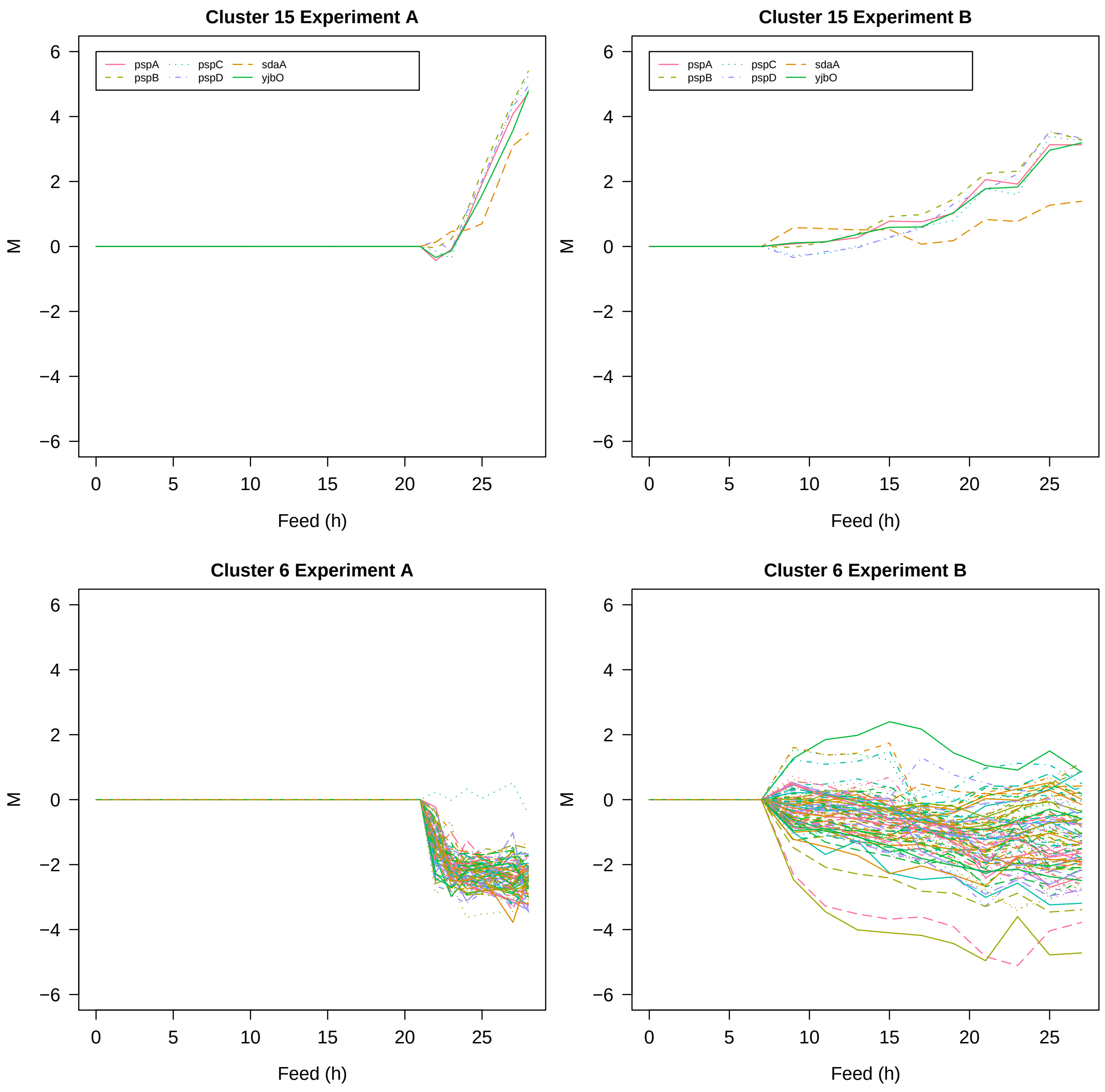


Figure 6.6: Comparison of cluster 15 and 6 of experiment A under fully induced conditions (left panels) and limited conditions (right panels).

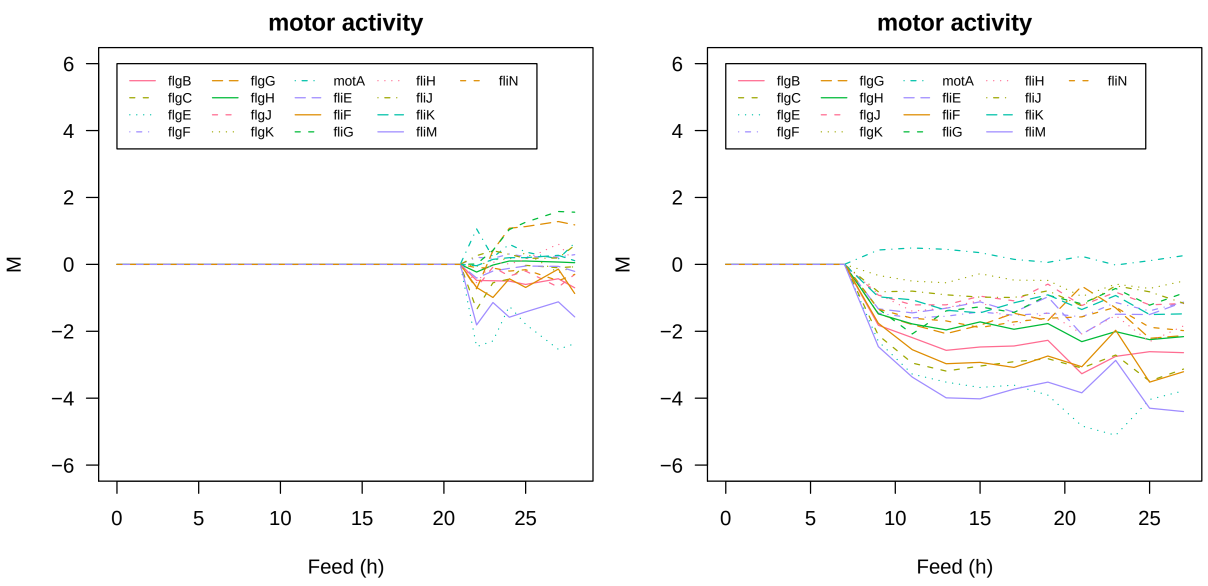


Figure 6.7: Motor Activity (GO:0003774) in experiment A (left panel) and experiment B (right panel).

6 (see Figure 6.6 bottom left panel). In experiment B the majority of genes grouped to cluster 6 of experiment A show a delayed rather than a direct down regulation in response to induction (bottom right panel of Figure 6.6) whereas a considerable number of genes shows no common behavior. In the following *fli*E, *fli*A and *lpp*, three genes with deviating profiles were selected to be examined in more detail.

*fli*A and *flg*E are the only genes of cluster 6 showing strong and direct down–regulation under limited induction conditions. These genes belong to the GO group GO:0003774 (motor activity). The expression patterns of all genes of this group are shown in Figure 6.7. In the experiment with limited induction (right panel) all these genes were down regulated in contrast to experiment A (left panel) where no common response was detected. A possible explanation of these findings is that cells exposed to high but tolerable induction levels (experiment B) were able to compensate for depletion of cellular resources and capacities by reduction or cessation of non essential branches of the metabolism. In the defined environment of a bioreactor motility provides no benefits but demands energy and metabolites. Consequently, the cells cut down these expenses to maintain central cellular functionality.

Another interesting transcription profile in cluster 6 of experiment A is given by the murein lipoprotein *lpp*. Under fully induced conditions this gene is down–regulated whereas

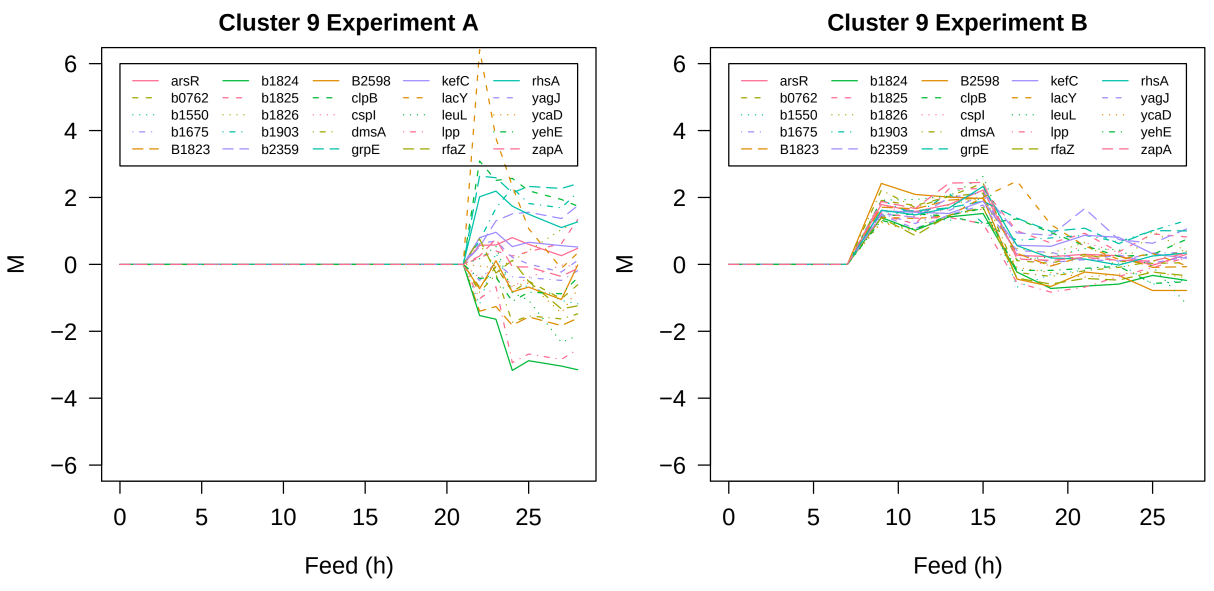


Figure 6.8: Cluster 9 of experiment B under fully induced conditions (left panel) and limited conditions (right panel).

under limited induction conditions the expression level of this gene follows a distinct up- and down trend coinciding with the process phenomenons described in Section Data. In the cluster solution of experiment B *lpp* is assigned to a cluster of 25 genes (Figure 6.8) comprising genes involved in membrane lipid synthesis (*gns*B, *rfa*Z), membrane sysnthesis (*rhs*A), cell division (*zap*A) cold shock response (*csp*BCI) but also 8 predicted genes of unknown function. Lpp is the major lipoprotein of the outer membrane and one of the most abundant proteins in *E. coli.* It is essential for the stabilization and integrity of the bacterial cell envelope (Hirashima et al. 1974). The *gns*B gene increases the membrane fluidity and flexibility (Sugai et al. 2001). Cells activate an energy demanding protective strategy by synthesis and translocation of Lpp which is in contrast to the cut–back strategy described above.

This comparative analysis of the two experiments clearly reveals the irreversible overload of metabolism in the experiment with full induction. Cells were not able to respond in a concerted and accurate way. On the other hand, cells exposed to limited induction of recombinant gene expression cope with emerging stress by different strategies in order to survive. The described cellular responses are similar to transitional changes of cells entering the stationary phase. This spore–like multiple–stress resistance state enables maintenance of viability under bad conditions (Ramirez Santos et al. 2005). The identified genes involved in these defense mechanisms are potential candidates for indepth investigation and provide clues about the regulatory mechanisms involved.

6.1.5 Conclusions

The interactive visualization tool **gcExplorer** was developed in order to make cluster analysis useful for practitioners. It allows not only to visualize the cluster structure, beyond that the gene clusters are plotted or shown in HTML tables with links to databases. Additional properties of the clusters like cluster size or cluster tightness can be highlighted as well as external information like functional grouping. Furthermore **gcExplorer** provides functions for comparative graphical analysis of different μ–array experiments. **gcExplorer** is a userfriendly software tool for the analysis of gene expression data and very helpful for practitioners to get an overview on the output of μ–array experiments.

In this study microarray data from two processes with a strong recombinant *E. coli* expression system were analyzed. Neighborhood graphs enable the investigation of the underlying cluster structure and relationships between clusters. The implemented features for functional grouping allowed the assignment of cellular functions to clusters and provided hints about the functionality of other genes belonging to a certain cluster. Comparative graphical analysis of these two experiments resulted in the identification of differences in the cellular response and a number of interesting gene candidates involved. It was shown that the cellular strategies are different in the two DNA–μ–array experiments. Useful information was extracted for the further advancement of the expression system by means of genetic engineering or by means of process engineering.

6.2 Comparison of cluster algorithms for *E. coli* BL21 data

6.2.1 Data

The goal of this experiment is the detailed investigation of the cellular response of *E. coli* BL21(DE3) to high level expression of recombinant human super–oxide–dismutase (SOD) on the transcriptional level. Three biological replicates were generated by using a carbon limited exponential feedbatch cultivation similar to industrial setups for large scale production. For induction of the system a single pulse of IPTG yielding in a fully induced system is applied one doubling past feed start. In order to achieve a proper time–resolution of cellular responses sampling starts with a high frequency for the first hours past induction and decreases in the course of the experiment. The resulting process data shown in Figure 6.9 demonstrate that the experimental setup provides highly reproducible samples optimally suited for the intended application in DNA-microarray experiments. The deviation of the total cell dry weight from the calculated course is caused by the rapidly increasing specific content of recombinant protein and the thereby triggered metabolic load on the cellular system. The recovery of growth around eight hours past induction is due to the proliferation of a non-producing plasmid free cell population (Figure 6.9 A). Figure 6.9 B shows that the capacity of the cellular protein processing machinery gets overstrained and bit by bit more of the recombinant protein accumulates in miss-folded form in protein aggregates (inclusion bodies).

The data consists of 530 genes at 16 time points after filtering genes not differentially expressed at least at one time point (p-value < 0.05, log ratio $M > 2$ and average intensity $A > 8$).

In the case of time course microarray data the definition of clusters is not clear and therefore the quality of a cluster solution is difficult to evaluate. Even the number of components is hard to specify as practitioners usually prefer small clusters which can easily be investigated. However, too many cluster are even harder to control.

Biologically meaningful partitions are usually those where co–expressed genes of operons are grouped together. For that purpose a set of 77 genes was identified containing selected subsets of operons. The subset is summarized in Table 6.1 and displayed in Figure 6.10. In this case the adjusted Rand index is calculated on this control set of 77 genes and not on the complete partitions.

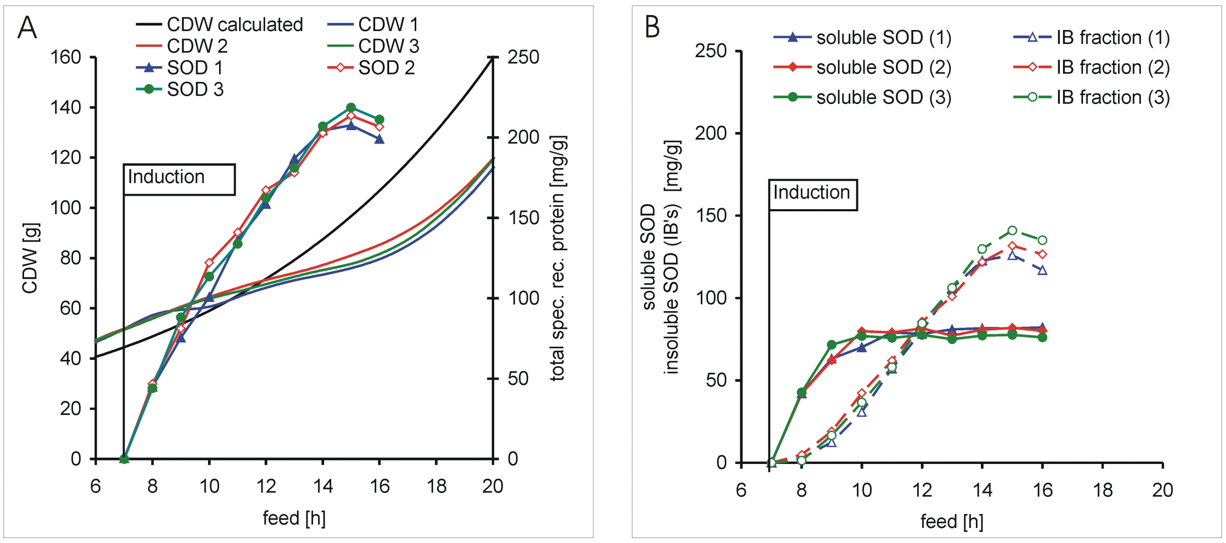


Figure 6.9: Protein production process with *E. coli* BL21(DE3). Vertical line indicates time point of induction.

Table 6.1: A–priori grouping of 77 genes of the *E. coli* data set using operon information.

Operon Name	Number of Genes
atpIBEFHAGDC	8
nuoABCEFGHIJKLMN	13
dnaKJ	2
thiCEFSGH	6
pspABCDE	4
flgBCDEFGHIJ	3
gatYZABCD	5
cyoABCDE	4
manXYZ	3
sdhCDAB–sucABCD	8
malEFG	3
malK-lamB-malM	3
malXY	1
dppABCDF	2
mraZW-ftsLI-murEF-mraY-murD-ftsW-murGC-ddlB-ftsQAZ	1
rpsJ-rplCDWB-rpsS-rplV-rpsC-rplP-rpmC-rpsQ	1
yceD-rpmF-plsX-fabHDG-acpP-fabF	1
thrS-infC-rpmI-rplT-pheMST-ihfA	2
ilvLG_1G_2MEDA	2
arnBCA-yfbH-arnT-yfbWJ	2
sufABCDSE	2
hinT-ycfLM-thiK-nagZ-ycfP	2

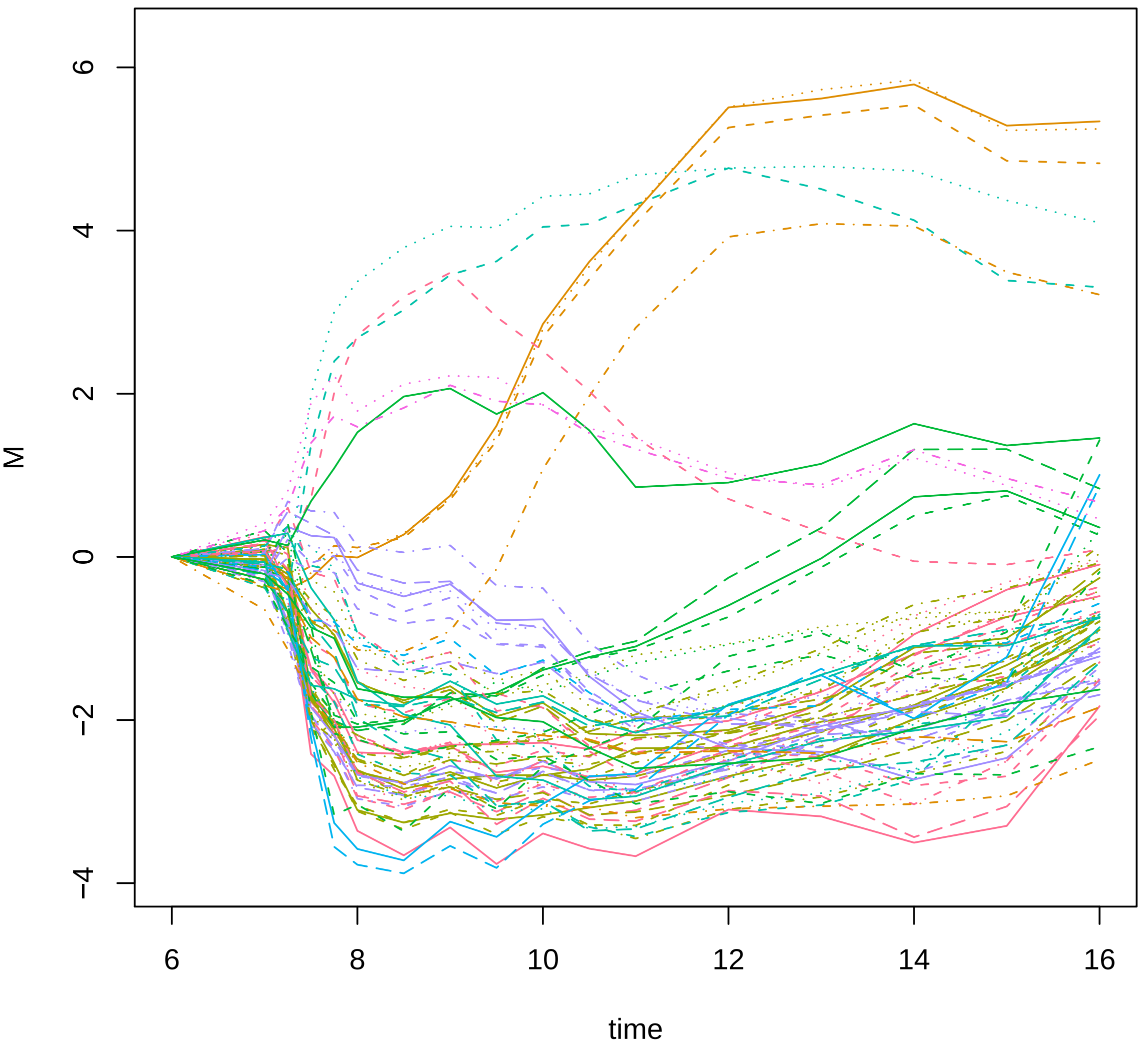


Figure 6.10: Expression patterns of the control set of 77 genes colored by operon.

Table 6.2: Results of the partitioning methods on the *E. coli* data set.

algorithm	dist	cent	type	k	NA	time	crand
kmeans	eucl	mean	orig	30	0	0	0.31
kmeans	man	median	orig	30	0	0	0.26
kmeans	cor	optim	orig	30	0	0.32	**0.51**
kmeans	max	optim	orig	30	0	1.63	0.39
qtclust	eucl	mean	orig	24	27	0	0.15
qtclust	man	mean	orig	25	32	0.02	0.26
qtclust	cor	mean	orig	21	12	0	0.21
qtclust	max	mean	orig	22	31	0.06	0.19
kmeans	euc	spline	orig	30	0	0.15	0.32
kmeans	euc	mean	fd	30	0	0	**0.45**
qtclust	euc	mean	fd	21	26	0.01	0.15

6.2.2 Partitioning methods

Cluster solutions of the partitioning methods are summarized in Table 6.2. For each method the algorithm is started 10 times keeping the best solution. Table 6.2 gives the number of clusters ("k"), the number of outliers identified by QT–Clust ("NA"), the system time and the adjusted Rand index. It can be seen that no method is able to correctly group all the operons into separate clusters. However, K–Means clearly outperforms QT–Clust. The best agreement between the a–priori grouping and a cluster solution is found for K–Means and "1-Correlation" distance yielding an adjusted Rand index of 0.51. The second best method is K–Means clustering of the functional data with an adjusted Rand index of 0.45. The worst methods in this comparison are QT–Clust using Euclidean distance on the original data as well as on the functional data where the Rand index is only 0.15.

6.2.3 Model-based clustering

Different cluster solutions using random initialization were compared starting with 5 to 60 components and yielding up to 29 components (see Figure 6.11). The likelihood criterion selects the model starting with $k = 58$ where 27 components are found.

The data was clustered using the methods investigated in the simulation study using 58 components in model-based clustering and 30 centers in spectral clustering. Cluster results using the different initialization strategies are given in Table 6.3 where k is the number of components found. The different cluster solutions are compared using the log-likelihood,

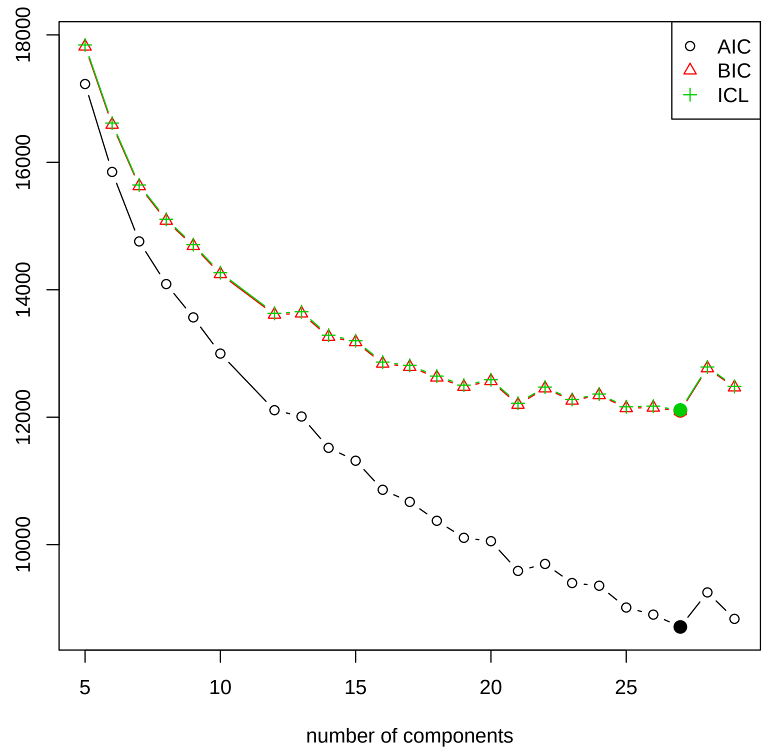


Figure 6.11: Number of components for the *E. coli* data when using random initialization for the initialization of mixtures of LMs.

BIC and AIC. Again the adjusted Rand index is computed only for the subset of 77 genes. For mixtures of LMs the number of components found is between 15 (sampling method) and 31 (short runs of EM). Initializing EM in the solution of short runs of EM is also the method with the largest log-likelihood and smallest AIC. BIC selects the solution of SEM where 23 clusters are found. The agreement between the cluster solution and the a–priori grouping of the 77 genes is highest for random initialization (0.38).

In the case of mixture models with RI the number of components found varies between 10 (incremental method) and 35 (random initialization and short runs of EM). AIC, BIC and log-likelihood select the results of random initialization as the best solution where the agreement between the a–priori grouping and the cluster solution is 0.39. The classification criterion selects the cluster solution of short runs of EM as the best partition with an adjusted Rand index of 0.48 which is slightly smaller than the Rand index of the best partitioning method.

6.2.4 Conclusions

The different cluster methods were applied to real microarray data from *E. coli.* For the real data set a subset of 77 genes where the grouping to operons is known a–priori was used to evaluate the goodness of the partitions. The results agree very well with simulation results on artificial data where high noise level is present in the data. K–Means clustering of the original data with "1-Correlation" distance, K–Means clustering of the functional data as well as mixtures of LMMs yield the best results.

Table 6.3: Results of the initialization strategies used on the *E. coli* data set using mixtures of LMs and mixtures of LMMs when starting with $k = 58$ components.

	RI	k	time	iter	logLik	BIC	AIC	crand
sc	N	30	16	-	-	-	-	0.27
rep.em	N	28	1305	39	-4045	12603	9088	**0.38**
sc.em	N	30	104	48	-3814	12464	8698	31
cem	N	18	305	21	-5080	12500	10678	0.29
cem.em	N	18	99	68	-4954	12800	10547	0.32
sem	N	23	1638	96	-4437	**12038**	9573	0.29
sem.em	N	23	16	9	-4403	12502	9623	0.29
tol	N	**31**	260	9	-3893	12765	8886	0.33
tol.em	N	**31**	155	74	**-3683**	12347	**8468**	0.31
sam	N	15	24410	47	-5303	13010	11138	0.32
inc	N	18	4357	20	-5202	13243	11031	0.27
rep.em	Y	**35**	31666	37	**-3113**	**8748**	**6783**	0.39
sc.em	Y	30	2142	53	-3455	9072	7388	0.38
cem	Y	23	5115	21	-3799	9253	7964	0.29
cem.em	Y	23	2969	88	-3717	9090	7800	0.44
sem	Y	31	25464	96	-3267	8769	7029	0.38
sem.em	Y	31	568	14	-3261	8757	7017	0.36
tol	Y	**35**	5302	10	-3256	9036	7071	**0.48**
tol.em	Y	**35**	2251	48	-3124	8772	6806	0.42
sam	Y	13	11434	90	-4559	10049	9323	0.19
inc	Y	10	23870	12	-4990	10695	10139	0.24

Chapter 7

Conclusions

Clustering gene expression profiles is a helpful tool for finding biologically meaningful groups of genes without prior information from databases. In this thesis several cluster methods for time course gene expression data are evaluated on artificial data sets where the true cluster membership is known in order to find the most appropriate methods for real microarray data. The different methods are compared when different types of noise are present in the data as technical problems and measurement errors can easily distort the data. As expected the performance of the algorithms differs the most when high noise is added to the data sets. In this case only K–Means clustering of the original data using "1 - Correlation" distance and mixtures of linear mixed models yield reliable results.

On a real microarray data set the true cluster membership is unknown. Therefore the likelihood criterion is used to evaluate the cluster solutions. Additionally a subset of 77 genes was selected where the grouping to co–regulated operons is known. The results on simulated data agree very well with the real world example. Again K–Means clustering of the original data using "1 - Correlation" distance and mixtures of linear mixed models yield the best results.

Beside the evaluation of different cluster methods tools for the interactive exploration of gene cluster solutions are presented in this thesis. The interactive visualization toolbox **gc-Explorer** allows not only to visualize the cluster structure in form of neighborhood graphs, beyond the gene clusters are plotted or shown in HTML tables with links to databases. Examples of the utility of this package for practitioners are presented in several examples. The functionality of the package includes different node representations using node coloring and the choice of node symbols. Additional properties of the clusters like cluster size or cluster tightness are highlighted as well as external information like functional grouping.

Graphs can be modified by removing nodes and edges or by zooming into a subgraph of interest. Further, the functional relevance test is presented which can be used to test the association of a functional grouping and cluster solution. Finally, the validity of a cluster solution is judged based on its performance on another data set where the same set of genes is investigated under different experimental conditions.

Furthermore **gcExplorer** provides functions for comparative graphical analysis of different microarray experiments. This resulted in the identification of differences in the cellular response and a number of interesting gene candidates involved. It was shown that the cellular strategies are different in the two DNA microarray experiments. Useful information was extracted for the further advancement of the expression system by means of genetic engineering or by means of process engineering.

Appendix A

Vignette: How to use the gcExplorer

A.1 Overview

In this Chapter the R code for the analysis described in the Sections 5.4 and 3.3 is given. Details about the different options and arguments can be found in the corresponding sections and in the help pages of the functions (also given in Appendix B). **gcExplorer** depends on R package **flexclust** (Leisch 2006) and Bioconductor package **Rgraphviz** (Carey et al. 2005).

A.2 Exploratory Analysis

A.2.1 Interactive exploration

First the *E. coli* PS19 data is clustered using the stochastic QT–Clust algorithm implemented in function `qtclust` of package **flexclust**.

```
> library("gcExplorer")
> data("ps19")
> set.seed(1111)
> cl1 <- qtclust(ps19, radius = 2, save.data = TRUE,
+          control = list(min.size = 5))
> cl1

kccasimple object of family 'kmeans'
```

```
call:
qtclust(x = ps19, radius = 2, control = list(min.size = 5),
    save.data = TRUE)

cluster sizes:

   1    2    3    4    5    6    7    8    9   10
 302  299   41   59   52   31   30   26   14   10
  11   12   13   14 <NA>
  10    5   12   10   17
```

The resulting cluster object consisting of 14 clusters is visualized using **gcExplorer** (see Figure 5.2). A color theme can be specified using argument `theme`. Argument `filt` can be used to specify which edges should be plotted, i.e., two centroids are only connected in the graph if the similarity is above a certain threshold. Argument `layout` can be used to specify one of the non–linear layout algorithms implemented in **Rgraphviz**:

dot: hierarchical layout algorithm for directed graphs

neato and fdp: layout algorithms for large undirected graphs

twopi: radial layout

circo: circular layout

```
> gcExplorer(cl1, layout = "dot", theme = "blue", filt = 0)
```

The resulting neighborhood graph of this cluster solution of the PS19 data is displayed in Figure 5.2.

The interactive `gcExplorer` can be called using an arbitrary panel function, e.g.,

```
> gcExplorer(cl1, theme = "blue", filt = 0,
+          panel.function = gcProfile)
```

for line plots showing the corresponding gene expression profiles. An example of the interactive usage of the **gcExplorer** is given in Figure 5.3. By clicking on the nodes of the neighborhood graph new graphics devices pop up showing the corresponding cluster by using the stated panel function. In this example clusters 3, 4, 7, 13 and 14 are visualized by plotting the corresponding gene expression profiles and cluster 3 is also displayed in form of an HTML table using panel function `gcTable`.

A.2.2 Node Functions

Color coding

Further information can be added to the neighborhood graph by the use of color coding specified by argument `node.function`. Some examples of color coding are shown in Figure 5.4. The color theme can be modified using argument `theme`. In panel (a) cluster size is highlighted using function `node.size`, i.e., dark node symbols indicate large clusters and light node symbols indicate small clusters. A legend is added if the position of the legend is specified using argument `legend.pos`.

```
> gcExplorer(cl1, filt = 0, theme = "blue",
+          node.function = node.size,
+          legend.pos = "bottomright")
```

In panel (b) cluster tightness (node function `node.tight`) is used where dark nodes correspond to tight clusters which usually contain groups of genes with clearly defined gene expression profile.

```
> gcExplorer(cl1, filt = 0, theme = "red",
+          node.function = node.tight,
+          legend.pos = "bottomright")
```

In panels (c) and (d) two functional groups are investigated. In panel (c) clusters with accumulation of σ_{32}–regulated genes are highlighted which are related to heat shock response. The assignment of *E. coli* genes to Sigma factors is given in data `sigma`. In this case node function `node.go` is used where further arguments are passed using argument `node.args`. `gonr` is the name of the functional group under investigation, `source.id` and `source.group` contain gene identifiers and their assigned groups for the organism and `id` is the vector of identifiers for the clustered data set.

```
> data("sigma")
> gcExplorer(cl1, filt = 0, theme = "green",
+          node.function = node.go,
+          node.args = list(gonr = "Sigma32",
+                           source.id = sigma[,4],
+                           source.group = sigma[,1],
+                           id = bn_ps19),
+          legend.pos = "bottomright")
```

In panel (d) the GO term ”flagellar motility” which is part of the gene ontology biological process classification is shown. The assignment of *E. coli* genes to GO biological process terms is given in data set `gobp`.

```
> data("gobp")
> gcExplorer(cl1, filt = 0,
+         node.function = node.go,
+         node.args = list(gonr = "flagellar motility",
+                          id = bn_ps19,
+                          source.group = gobp[,3],
+                          source.id = gobp[,1]),
+         legend.pos = "bottomright")
```

Node symbols

Another option for adding information to the display of the neighborhood graph is to use different graphical symbols for the representation of nodes. For that purpose **gcExplorer** makes use of R package **symbols** (`http://r-forge.r-project.org/projects/symbols`).

The most natural node symbols in the case of time–course gene expression data is to use line plots showing the gene expression profiles for either the cluster centroids or the whole group of genes in a certain cluster.

First, a grid–based `node.function` has to be defined, e.g.,

```
> gmatplot <- function (object, cluster, bgdata) {
+         grid.rect()
+         data <- object@data@get("designMatrix")
+         ylimits <- c(min(data, na.rm = TRUE), max(data, na.rm = TRUE))
+         index <- (object@cluster == cluster)
+         nodedata <- data[index,]
+         symb.matplot(1:ncol(nodedata), t(nodedata), type = "l",
+                 col = "gray", ylim = ylimits, pch = 1)
+         center <- object@centers[cluster,]
+         symb.matplot(1:ncol(object@centers), center, type = "l",
+                 col = "red", ylim = ylimits, pch = 1)
+ }
```

Now this node function is used in the `gcExplorer` by setting argument `doViewPort = TRUE` which enables the use of viewports.

```
> gcExplorer(cl1, filt = 0,
+         node.function = gmatplot,
+         doViewPort = TRUE)
```

Figure 5.5 gives a very good overview of the cluster solution and the single gene clusters where similarities in gene expression profile can directly be investigated. It can be seen that clusters containing down–regulated genes are located in the bottom left part of the graph whereas up–regulated genes are located in the right part of the graph. Further, there are no edges between clusters of up- and down–regulated genes.

Another example for node symbols are pie charts. Here is a user–defined grid pie function

```
> gpie <- function (object, cluster, bgdata) {
+         clusterindex <- object@cluster
+         clusterindex[is.na(clusterindex)] <- 0
+         index <- (clusterindex == cluster)
+         A2.cl <- bgdata[index,]
+         NOgroup <- length(A2.cl[A2.cl])
+         groupA <-length(A2.cl[!A2.cl])
+         symb.pie(c(NOgroup,groupA), labels = "",
+                 radius = 1.1,
+                 col = c("white", "skyblue"))
+ }
```

For demonstration purpose the F–statistic for differential expression for each gene is used here where the amount of genes with F–statistic ≤ 20 is given in white and the amount of genes with F–statistic > 20 is given in skyblue (see Figure 5.6) left panel.

```
> f2 <- f<20
> gcExplorer(cl1, filt = 0, theme = "blue",
+         node.function = gpie,
+         bgdata = as.data.frame(cbind(as.numeric(f2))),
+         doViewPort = TRUE)
> legend("topleft", inset = 0.05,
+         legend = c("F <= 20", "F >  20"),
+         fill = c("white", "skyblue"))
```

Grid–based boxplots can be used as node symbols using the following user–defined function.

```
> gbxp <- function (object, cluster, bgdata) {
+         ylim <- c(min(bgdata), max(bgdata))
+         index <- (object@cluster == cluster)
+         nodedata <- bgdata[index,]
+         symb.bxp(boxplot(nodedata, plot = FALSE),
+                 frame.plot = TRUE, ylim = ylim)
+ }
```

In the right panel of Figure 5.6 boxplots of the log F statistic are shown.

```
> gcExplorer(cl1, filt = 0,
+          node.function = gbxp,
+          bgdata = as.data.frame(log(f)),
+          doViewPort = TRUE)
```

A.2.3 Graph Modifications

Node modifications

In order to modify an existing graph the graph structure has to be saved.

```
> graph <- gcExplorer(cl1, filt = 0,
+          node.function = gmatplot,
+          doViewPort = TRUE)
```

Now the graph structure of object `graph` can be modified using function `gcModify`. In this example argument `kpNodes` is used to keep only the stated nodes.

```
> graph1 <- gcModify(graph,
+          kpNodes = c("k5", "k7", "k9", "k10", "k13"),
+          doViewPort = FALSE)
```

The remaining subgraph is now investigated in detail using the `zoom` argument.

```
> graph2 <- gcModify(graph1, zoom = "auto")
```

In the left panel of Figure 5.7 the subgraph is shown with no node function setting argument `doViewPort=FALSE`. In the right panel the zoomed subgraph is shown.

Edge modifications

Filtering by cluster similarity can be used to simplify the original neighborhood graph. Edges between nodes are only drawn if the similarity of a cluster to another cluster is above a certain threshold, e.g., at least 10%. This prevents the graph from being too complex.

Now the similarity matrix is modified.

```
> d1 <- clusterSim(cl1)
> d1[d1 < 0.1] <- 0
> d2 <- d1
> d2[d2 < 0.2] <- 0
> d3 <- d2
> d3[d3 < 0.3] <- 0
```

Here d1 is the original cluster similarity matrix which can be extracted from the cluster object using function clusterSim, d2 is the similarity matrix where all values smaller 0.1 are set to 0 and so on.

Again we save the original neighborhood graph to object graph. In order to modify the edges of an existing graph function gcModify is used specifying argument clsim.

```
> graph <- gcExplorer(cl1, filt = 0)

> gcModify(graph, clsim = d1)

> gcModify(graph, clsim = d2)

> gcModify(graph, clsim = d3)
```

Examples of the neighborhood graph where the different cutoff values for drawing edges are shown are given in Figure 5.8.

A.3 Inferential Analysis

A.3.1 Compare Cluster Solutions

Function comp_test is now used to test the goodness of the cluster solution obtained for the PS19 data when applied to the PS17 data where the same set of genes was investigated under different experimental conditions.

```
> data(comp19)
> ct1 <- comp_test(comp17, clusters(cl1), N = 1000)

> ct1

     size obs.avdist 5%qu.perm p.val
[1,]  302  0.5796572 0.9464934 0.000
[2,]  299  0.5542297 0.9405271 0.000
[3,]   41  0.6524593 0.8282290 0.001
[4,]   59  0.6163776 0.8537883 0.000
[5,]   52  0.7345407 0.8358843 0.003
[6,]   31  0.6134388 0.7862977 0.000
[7,]   30  0.6578180 0.7783083 0.002
[8,]   26  0.8185665 0.7699189 0.098
```

```
 [9,]   14  0.5215982 0.6766737 0.004
[10,]   10  0.3789129 0.6203714 0.001
[11,]   10  0.6966036 0.6304406 0.115
[12,]    5  0.4900756 0.4534873 0.069
[13,]   12  0.9621026 0.6575232 0.529
[14,]   10  0.6162787 0.6281170 0.038
```

The test result consists of cluster size, observed average within cluster distance, the 5% quantile of the permuted average distances and the probability of observing a lower within cluster distance ("p.val") by randomly assigning the genes to clusters. In this case 10 out of 14 clusters have a significantly smaller within cluster distance when using the cluster solution of the PS19 experiment compared to random assignment. These 10 groups of genes form tight clusters under both conditions and therefore likely to be co–regulated.

A.3.2 Functional Relevance Test

Another possibility for external validation of a cluster solution is to test the functional relevance of single edges, i.e., to test the relationship between a functional grouping and a cluster solution. In this example the *E. coli* oxygen data set Covert et al. (2004) is used and the GO term GO:0009061 (anaerobic respiration) is investigated.

The data set is loaded and clustered into 43 clusters using `qtclust`.

```
> data(oxygen)
> set.seed(1111)
> cl2 <- qtclust(oxygen, radius = 3, save.data = TRUE,
+          control = list(min.size = 5))
> cl2

kccasimple object of family 'kmeans'

call:
qtclust(x = oxygen, radius = 3, control = list(min.size = 5),
    save.data = TRUE)

3288 points in 43 clusters, 100 outliers
Distribution of cluster sizes:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00    8.00   13.00   76.47   37.50  671.00
```

Function `Group2Cluster` is used to find the cluster membership of all genes involved in anaerobic respiration and the functional relevance test is implemented in function `edgeTest`. An edge is only tested if the number of functionally related genes is above a predefined threshold given by argument `min.size`. Argument `filt` can be used to filter edges which are smaller than a predefined similarity threshold.

```
> g1 <- Group2Cluster(cl2, gonr = "GO:0009061",
+          source.group = gobp[,3], source.id = gobp[,1],
+          id = bn_oxy)
> eT <- edgeTest(cl2, group = g1, min.size = 2, filt = 0, N = 1000)
> eT$res
```

```
      Clsize1 Clsize2 Diff.in.Prop. P-value
1~2       671     526   0.015523621   1.000
1~3       671     424   0.009578073   1.000
4~6       378     209   0.018126123   1.000
2~7       526     121   0.008343022   1.000
4~7       378     121   0.022475841   1.000
6~8       209     108   0.008328903   1.000
4~12      378      16   0.114417989   0.593
1~14      671      33   0.141579732   0.509
2~14      526      33   0.157103353   0.502
1~16      671      13   0.113607704   0.593
3~16      424      13   0.123185776   0.568
1~21      671       9   0.404205994   0.000
3~21      424       9   0.413784067   0.000
14~21      33       9   0.262626263   0.046
14~22      33      12   0.484848485   0.000
21~22       9      12   0.222222222   0.129
4~25      378      10   0.189417989   0.289
6~25      209      10   0.171291866   0.335
12~25      16      10   0.075000000   0.931
2~32      526      11   0.338921535   0.008
7~32      121      11   0.330578512   0.029
12~32      16      11   0.238636364   0.049
```

```
22~32       12       11   0.303030303   0.029
3~34       424        6   0.302672956   0.030
5~34       263        6   0.325728771   0.029
21~34        9        6   0.111111111   0.769
2~35       526       17   0.092932230   0.814
21~36        9        5   0.044444444   0.999
34~36        6        5   0.066666667   0.938
22~43       12        9   0.444444444   0.000
32~43       11        9   0.141414141   0.509
36~43        5        9   0.177777778   0.329
```

The output of function `edgeTest` gives detailed information about the tested edges, i.e., the corresponding cluster sizes, the difference in proportions and the p–value. The 95% quantile of the maxima of the permuted average distances is 0.22 and can be extracted by

```
> eT$quant

      95%
0.2222222
```

The accumulation of genes involved in anaerobic respiration is displayed in Figure 5.9 left panel. Here `edge.method = "mean"` is used to draw an undirected graph. In this case a different layout algorithm is selected using `layout = "neato"`.

```
> graph <- gcExplorer(cl2, filt = 0, theme = "blue",
+         node.function = node.group,
+         node.args = list(group = g1),
+         layout = "neato",
+         edge.method = "mean",
+         legend.pos = "bottomleft")
```

The p-values are now used to form a new similarity matrix using function `newclsim`. If the p–value of an edge is smaller than 0.05 the similarity value is set to 0.

```
> clsim1 <- newclsim(eT = eT$res, object = cl2, p.filt = 0.05)
> gcModify(graph, clsim1)
```

In Figure 5.9 right panel the modified neighborhood graph is displayed. It can be seen that clusters 32, 43, 36, 34, 21 and 22 contain most of the genes involved in anaerobic respiration and form a disconnected subgraph after testing the functional relevance of the edges.

A.4 Sessioninfo

This document was produced using

- R version 2.9.1 (2009-06-26), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=en_US;LC_PAPER=en_US;LC_NAME=C;LC_ADDRESS=C;LC_TELEPHONE=C;LC_MEASUREMENT=en_US;LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, stats4, utils
- Other packages: flexclust 1.2-1.1, gcExplorer 1.0-0, graph 1.22.2, lattice 0.17-25, modeltools 0.2-16, Rgraphviz 1.22.1, symbols 0.13
- Loaded via a namespace (and not attached): tools 2.9.1

together with version 2.20.2 of graphviz.

Appendix B

Documentation of gcExplorer

`gcSim`	*Create artificial cluster data*

Description

Functionality to create artificial time course gene cluster data.

Usage

```
gcSim(sim=c("arima","norm","pattern","noise","outlier"), time=10,
      sd=0.1, sd.ri=0, size=50, n=10, ar=NULL, o=NULL, cent)

gcData(...)
```

Arguments

`sim`	simulation method used
`time`	number of time points
`sd`	standard deviation of the expression profiles
`sd.ri`	standard deviation of the random intercept or gene specific shift
`size`	cluster size, either one value for all clusters or a vector of cluster sizes of length n
`n`	number of clusters
`ar`	any value between -1 and 1

`o`	the degree of differencing
`cent`	a data matrix giving expression profiles in rows, only used if `sim="pattern"` or `sim="outlier"`
`...`	Several `"gcSim"` objects can be combined using function `gcData`.

Details

`gcSim` is a unifying function to call different data simulators.

`arima` generates expression patterns that come from an integrated AR-process with AR order 1 that can be controlled via `ar` and the degree of differencing `o`. `sim="norm"` and `sim="noise"` generate normally distributed expression patterns where `sim="noise"` is used to form a noise set of genes.

`sim="pattern"` and `sim="outlier"` can be used to generate clusters based on a set of cluster centers which are passed to the functions using the argument `cent`. `sim="outlier"` can be used to test Jackknife distance measures.

`gcData` can be used to combine different artificial data generators.

Value

a data matrix

Author(s)

Theresa Scharl

See Also

`pattern`

Examples

```
## generate 10 clusters with normally distributed expression patterns:
data <- gcSim(sim="norm", time=16, sd=0.1, sd.ri=0.5,
              size=50, n=10)
matplot(t(data),type="l",pch=1)

## combine expression patterns that follow an ARIMA process and a null cluster:

data <- gcData(gcSim(sim="arima", time=16, sd=0.1, sd.ri=0.5,
                        size=c(20,50,100,100), n=4),
                  gcSim(sim="noise",time=16, size=100))
matplot(t(data),type="l")
```

`clusterPlot` *Cluster solution plot*

Description

Plot the expression profiles of the smallest clusters of an object of class `"kccasimple"`.

Usage

```
## S4 method for signature 'kccasimple':
clusterPlot(object, method = c("size", "tight"), layout = c(3, 4),
   xlabels = NULL, xlab = "time", ...)
```

Arguments

`object`	An object of class `"kccasimple"`.
`method`	Which clusters should be plotted: either small clusters or tight clusters.
`layout`	A vector of the form c(nr, nc). Only a subset of nr x nc clusters will be drawn. The arrangement of nr rows and nc columns is passed to the `layout` argument of `lattice` function `xyplot`.
`xlabels`	Either a numeric vector of time points giving the positions on the x-axis or a character vector with names of the positions on the x-axis.
`xlab`	Character string or expression giving label for the x-axis
`...`	Further arguments can be passed to function `xyplot`.

Author(s)

Theresa Scharl

Examples

```
data("hsod")
cl1 <- qtclust(hsod, radius = 2, save.data = TRUE)

clusterPlot(cl1, method = "tight",layout = c(3,2))
```

`comp_test` *Compare Cluster Results*

Description

Cluster validation by testing the validity of a cluster solution under different experimental conditions.

Usage

```
comp_test(data, cll, N = 500, quant=0.05, ...)
```

Arguments

`data`	data set with the same number of rows as the clustered data set.
`cll`	Vector of cluster memberships of the clustered data set.
`N`	Number of permutations.
`quant`	The defined quantile for the permuted average distances.
`...`	Further arguments can be passed to the subfunctions.

Value

A matrix giving for each cluster the size of the cluster, the observed average within cluster distance to the computed cluster center in the new data set, the defined quantile for the permuted average distances and the p-values, i.e., the proportion of permutations where the observed within cluster distance is lower than the permuted.

Author(s)

Theresa Scharl

Examples

```
data(comp19)
set.seed(1111)
cl3 <- qtclust(comp19,radius=1.5,family=kccaFamily(dist=distEuclidean,
   cent=colMeans),save.data=TRUE,control=list(min.size=5))
cl3

ct1 <- comp_test(comp17, clusters(cl3), N=1000)
ct1
```

`edgeTest` *Functional Relevance Test*

Description

Perform a functional relevance test on the edges of a neighborhood graph

Usage

```
edgeTest(object, min.size = 1, group, N = 500, filt = 0.1,
    useNH = TRUE, quant = 0.95)
```

Arguments

`object`	An object of class `"kccasimple"`.
`min.size`	Minimum number of grouped genes in a cluster to be considered for testing
`group`	Vector of cluster memberships of functionally grouped genes (from function `Group2Cluster`).
`N`	Number of permutations.
`filt`	Threshold for edges in the neighborhood graph to be considered for testing.
`useNH`	Use the neighborhood structure or test all combination of nodes?
`quant`	The defined quantile of the maxima of the permuted average distances.

Value

A list consisting of the matrix `res` and the defined quantile `quant` of the maxima of the permuted average distances. The matrix `res` gives the cluster sizes, the difference in proportions and the corresponding p-value for each edge considered.

Author(s)

Theresa Scharl

See Also

`Group2Cluster`

Examples

```
data("hsod")
data("gobp")
set.seed(1111)
cl1 <- qtclust(hsod, radius = 2, save.data = TRUE)

g1 <- Group2Cluster(cl1, gonr = "GO:0009061",
        source.group = gobp[,3], source.id=gobp[,1],
        id = bn_hsod)
test1 = edgeTest(cl1, group=g1, min.size=2, useNH=TRUE, filt=0.1, N=1000)
```

`fitsod` *E. coli Fermentation Data*

Description

E. coli Fermentation Fit Data Object containing M-values, P-values, GeneNames and Links zu NCBI. Output of limma function `write.fit` with links to NCBI added for each gene.

Usage

```
data(fitsod)
data(hsod)
data(gobp)
```

Format

A data frame with 4368 observations on the following variables.

`A` a numeric vector giving the mean A-values

`Coef.stress3A` a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

`Coef.stress3B` a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

`Coef.stress3C` a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

`Coef.stress3F` a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

`Coef.stress4` a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

`Coef.stress4A` a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

`Coef.stress5A` a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

`Coef.stress6` a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

`t.stress3A` a numeric vector giving the t-statistic to the coefficient estimate

`t.stress3B` a numeric vector giving the t-statistic to the coefficient estimate

`t.stress3C` a numeric vector giving the t-statistic to the coefficient estimate

`t.stress3F` a numeric vector giving the t-statistic to the coefficient estimate

`t.stress4` a numeric vector giving the t-statistic to the coefficient estimate

`t.stress4A` a numeric vector giving the t-statistic to the coefficient estimate

`t.stress5A` a numeric vector giving the t-statistic to the coefficient estimate

`t.stress6` a numeric vector giving the t-statistic to the coefficient estimate

`p.value.stress3A` a numeric vector giving the corresponding p-value

`p.value.stress3B` a numeric vector giving the corresponding p-value

`p.value.stress3C` a numeric vector giving the corresponding p-value

`p.value.stress3F` a numeric vector giving the corresponding p-value

`p.value.stress4` a numeric vector giving the corresponding p-value

`p.value.stress4A` a numeric vector giving the corresponding p-value

`p.value.stress5A` a numeric vector giving the corresponding p-value `p.value.stress6` a numeric vector giving the corresponding p-value

`F` a numeric vector giving the overall F-statistik

`F.p.value` a numeric vector giving the corresponding F-p-value

`Genes.Block` first block position of the gene

`Genes.Row` first row position of the gene

`Genes.Column` first column position of the gene

`Genes.GeneName` Short genename

`Genes.ID` Gene ID

`Genes.AccessionReference` Blattner numbers

`Genes.Status` status of the gene: always gene

`links` link to NCBI

Details

The data set `hsod` is a filtered subset of the original data. It contains 527 differentially expressed genes at the 8 time points. The vector `bn_hsod` contains the corresponding identifiers which can be used to search for functional groups in the data set `gobp`. `links_hsod` contains the corresponding links to the NCBI database.

`gobp` is a data set listing functional groups to gene identifiers. The data set consists of 8726 observations, the first column gives the gene identifier, the second column gives the gene name and the third column gives the functional group.

References

Duerrschmid, K., Reischer, H., Schmidt-Heck, W., Hrebicek, T., Guthke, R., Rizzi, A., Bayer, K. (2008). Monitoring of transcriptome and proteome profiles to investigate the cellular response of *E. coli* towards recombinant protein expression under defined chemostat conditions. Journal of Biotechnology 135, 34–44.

Examples

```
data(fitsod)
```

`gcExplorer` *Graphical Cluster Explorer*

Description

Plot a neighborhood graph for `"kccasimple"` cluster solutions.

Usage

```
## S4 method for signature 'kccasimple':
gcExplorer(object, layout = c("dot", "neato", "twopi","circo","fdp"),
   theme = "grey", edge.method=c("orig","mean","min","max"),
   node.function = NULL, node.args = NULL, doViewPort = FALSE,
   filt = 0.1, interactive = !is.null(panel.function), dev=c("one","many"),
   panel.function = NULL, panel.args = NULL, bgdata = NULL,
   colscale = NULL, mfrow = c(1,1), legend.pos = "none")
```

Arguments

`object`	Object of class `"kccasimple"`.
`layout`	Layout method used: One of `"dot"`, `"neato"`, `"twopi"`, `"circo"`, and `"fdp"`.
`theme`	Color theme used.
`edge.method`	Several methods are available to draw edges: `"orig"`, `"mean"`, `"min"`, and `"max"`, see details below.
`node.function`	Optional. Additional information about the clusters can be included in the representation of nodes. Either a function calculating node colors or a grid-based function (see `doViewPort`).
`node.args`	List of arguments which should be passed to `node.function`.
`doViewPort`	Currently not used in release version of the package. Call a grid-based function specified by argument `node.function` and use it for node representation?
`filt`	Cutoff value for similarities between clusters, edges above the threshold will be displayed.
`interactive`	Should the plot be interactive?

`dev`	Only used if `interactive=TRUE`. Display each cluster plot (specified by `panel.function`) in one device or open new devices for each cluster when clicking on a node.
`panel.function`	Only used if `interactive=TRUE`. The panel function which should be used to display the corresponding cluster
`panel.args`	List of arguments which should be passed to `panel.function`.
`bgdata`	Background data to be plotted by `panel.function` or `node.function`.
`colscale`	A vector of length 2 specifying the color range for edges and nodes, e.g. c(0,0.5).
`mfrow`	Only used if `interactive=TRUE`. The panel layout in which the panel plots should be displayed.
`legend.pos`	Position of the legend.

Details

A neighborhood graph is the default plot method for cluster objects of package `flexclust`. For large and highdimensional data sets like microarray data linear projection of the data into two dimensions may not scale well in the number of clusters. In this case non-linear arrangement of the nodes using layout algorithms from Graphviz can be helpful. An interface to Graphviz is provided in Bioconductor package `Rgraphviz`. One of the implemented layout algorithms can be selected using `layout`.

In a neighborhood graph each node corresponds to a cluster centroid. Two nodes are connected by an edge if there exist data points that have these two centroids as closest and second closest. The edge weights are taken from `clusterSim(object)`. The similarity between two clusters is bounded between 0 and 1 where well-separated clusters have values close to 0. The larger the similarity between clusters the stronger the edge will be drawn in the graph. The cutoff value for drawing the edge between two centroids can be chosen by argument `filt`. The larger the filt value the fewer edges will be drawn.

Originally the neighborhood graph is a directed graph. An edge will be drawn from centroid 1 to centroid 2 if there exists at least one data point that has centroid 1 as closest and centroid 2 as second closest. But there need not necessarily be a data point that has centroid 2 as closest and centroid 1 as second closest centroid. For this reason there are several methods for plotting the edges between nodes. The default `edge.method` is 'orig' where each edge is drawn separately with its corresponding weight. This method will result in a directed graph.

All other edge methods yield undirected graphs where the mean, minimum or maximum of the similarities between two clusters is used.

Additional information about the clusters can be included in the graph using `node.function` and `panel.function`. `node.function` is used for the node representation. If no `node.function` is given all nodes will be drawn in one color. The `node.function` can be used to calculate different colors for the nodes like cluster size or cluster tightness. Additionally `node.function` can be a grid–based function displaying the data in the underlying cluster, e.g. a scatterplot or a boxplot.

`gcExplorer` is implemented interactively. If `interactive=TRUE` `panel.function` is used to plot a cluster when clicking on the corresponding node. An example of a `panel.function` is given by function `gcProfile`.

Function `calcHCL` is used to calculate a HCL–based color.

Value

Object of class `"graphdata"` with the following slots: an object of class `"Ragraph"` (see package `Rgraphviz`), `object`, `bgdata`, `node.function`, `edge.method`, `theme` and `colscale`.

Author(s)

Theresa Scharl and Ingo Voglhuber

References

Theresa Scharl and Friedrich Leisch. gcExplorer: Interactive Exploration of Gene Clusters. Bioinformatics, 25(8): 1089-1090, 2009.

See Also

`node.tight`

Examples

```
data("hsod")
cl1 <- qtclust(hsod, radius = 2, save.data = TRUE)

gcExplorer(cl1, theme = "blue", node.function = node.size)
```

`gcModify` *Modify Ragraph objects and replot them*

Description

`gcModify` is a function to modify and plot an object of class `"graphdata"`:

- remove edges/nodes

- zoom

- draw custom node plots

Usage

```
## S4 method for signature 'graphdata':
gcModify(graphdata, clsim = NULL, rmNodes = NULL,
         kpNodes = NULL, edgeDep = TRUE, nodeDep = FALSE,
         zoom = c("none", "manual", "auto"),
         keepAspectRatio = TRUE, node.function = NULL,
         doViewPort = TRUE, bgdata = NULL)
```

Arguments

`graphdata`	List, containing object of class `"Ragraph"`, object of class `"kcca"` and other parameters of graph created by `gcExplorer`).
`clsim`	Matrix, new clsim to define removal or modification of edges.
`rmNodes`	Character vector, names of nodes to remove. (can not be used in combination with `kpNodes`)
`kpNodes`	Character vector, names of nodes to keep. (can not be used in combination with `rmNodes`)
`edgeDep`	Logical. If `TRUE` edges are removed, if they do not connect two nodes.
`nodeDep`	Logical. If `TRUE` nodes are removed, if they are not connected to other nodes.
`zoom`	One of: `"none"` - no zoom. `"manual"` - activate manual zoom, user interaction needed to specify area to be enlarged. `"auto"` - auto zoom, automatically enlarges graph to size of graphic device.

`keepAspectRatio`
: Logical. If `TRUE` aspect ratio is preserved.

`node.function`
: Grid based function for node plotting. To work correctly, the function will take three arguments:

  `object` is an object of class `"kcca"`.

  `cluster` is an integer giving the node (i.e., cluster) number.

  `bgdata` is a data.frame of external data.

`doViewPort`
: Logical. If `TRUE` `node.function` ist called to draw nodes.

`bgdata`
: Data.frame. external data for node drawing. (passed to `node.function`).

Details

`gcModify` is a tool to modify and plot graphs created by `gcExplorer`, zoom certain areas of the plot and use grid-based functions to draw custom node plots.

Value

Object of class `"graphdata"` with the following slots: an object of class `"Ragraph"` (see package `Rgraphviz`), `object`, `bgdata`, `node.function`, `edge.method`, `theme` and `colscale`.

Author(s)

Ingo Voglhuber

See Also

`gcExplorer`

Examples

```
data("hsod")
library(flexclust)
set.seed(1111)
cl1 <- qtclust(hsod, radius = 2,
               family = kccaFamily(dist = distEuclidean,
                    cent = colMeans), save.data = TRUE)

## create Ragraph object from kcca object with gcExplorer
```

```
graph <- gcExplorer(cl1, theme = "blue", node.function = node.size)

## extract and modify clsim
clsim <- clusterSim(cl1)
clsim[clsim < 0.5] <- 0

## use modified clsim on Ragraph object to remove edges (<0.5)
gcModify(graph, clsim)

## use nodeDep=TRUE to delete nodes without edges
gcModify(graph, clsim, nodeDep = TRUE, zoom = "none")

## use zoom="auto" to center and maximize subgraph
gcModify(graph, clsim, nodeDep = TRUE, zoom = "auto")

## Not run:
## R package symbols is available from Rforge:
## http://r-forge.r-project.org/projects/symbols/

require("symbols")

## create a grid based plotting function: plot cluster data and centers in matplot.
gmatplot <- function (object, cluster, bgdata) {
        grid.rect()
        data <- object@data@get("designMatrix")
        ylimits <- c(min(data, na.rm = TRUE), max(data, na.rm = TRUE))
        index <- (object@cluster == cluster)
        nodedata <- data[index,]
        symb.matplot(1:ncol(nodedata), t(nodedata), type = "l",
                col = "gray", ylim = ylimits, pch = 1)
        center <- object@centers[cluster,]
        symb.matplot(1:ncol(object@centers), center, type = "l",
                col = "red", ylim = ylimits, pch = 1)
}
## use grid based node function to draw nodes
gcModify(graph, clsim, nodeDep = TRUE, zoom = "auto",
         node.function = gmatplot)
```

```
## End(Not run)
```

`gcOffline` *Offline gcExplorer*

Description

Save `gcExplorer` plots or tables to a file.

Usage

```
## S4 method for signature 'kccasimple':
gcOffline(object, panel.function, panel.args=NULL,
    type=pdf, file="gcOffline", which=NULL, html=FALSE, ...)
```

Arguments

`object`	Object of class `"kccasimple"`.
`panel.function`	Only used if `interactive=TRUE`. The panel function which should be used to display the corresponding cluster
`panel.args`	List of arguments which should be passed to `panel.function`.
`type`	Create graphics of type `type`, e.g., pdf, postscript, jpeg, png.
`file`	File name prefix used for graphics files. Of the form file-which.type and file-graph.type.
`which`	A vector specifying if all cluster plots (default) or only a subset should be created.
`html`	Logical. Does the `panel.function` produce HTML tables.
`...`	Further arguments can be passed to `gcExplorer`.

Author(s)

Theresa Scharl

See Also

`gcTable`, `gcProfile`

Examples

```
data("hsod")
set.seed(1111)
cl1 <- qtclust(hsod, radius=2, save.data=TRUE)

# create three files: hsod-003.pdf, hsod-005.pdf, hsod-graph.pdf
gcOffline(cl1,panel.function=gcProfile, file="hsod", which=c(3,5))

# create two files: hsod-003.html, hsod-005.html
gcOffline(cl1, panel.function = gcTable, html = TRUE,
          panel.args = list(links = links_hsod),
          file = "hsod", which=c(3,5))

 # tidy up
unlink(list.files()[grep("hsod-",list.files())])
```

`gcProfile` *Plot for cluster results*

Description

Plot a single cluster of a 'kccasimple' object.

Usage

```
## S4 method for signature 'kccasimple':
gcProfile(object, which, data = NULL, cexl = 0.8, xlab = "",
   ylab = "M", ylim=c(-6,6), cex.axis=1, xlabels=NULL,
   opar = par(las=1, mar=c(5, 4, 2, 0.5) + 0.1),
   data.type=c("time", "other"), legend=TRUE, main=NULL, ...)
```

Arguments

`object`	an object of class `"kccasimple"`
`data`	Plot either the data stored in `object` or external data.
`which`	Number of the cluster.
`cexl`	Point size of the legend.

`xlab`	Label for the x-axis.
`ylab`	Label for the y-axis.
`ylim`	Range of the y-axis.
`cex.axis`	Point size of x-axis.
`xlabels`	Positions on the x-axis. Default is `1:ncol(data)`.
`opar`	Graphical parameters.
`data.type`	If the data come from arbitrary source (default) colnames of the data are used as xlabels if not stated otherwise using `xlabels`. If the data comes from a time course experiment x-values start at 0 and different time intervals are supported.
`legend`	Logical. Should a legend be drawn?.
`main`	Main title of the plot. If null ”Cluster i” is used.
`...`	Further arguments can be passed to `matplot`.

Author(s)

Theresa Scharl

Examples

```
data("hsod")
cl1 <- qtclust(hsod, radius=2, save.data=TRUE)

gcProfile(cl1, which=5)
gcProfile(cl1, which=5, xlabels=c(0,8,15,22,45,68,90,150,180),
         xlab="time after induction [min]",data.type="time")
```

`gcTable` *HTML table for cluster results*

Description

Create HTML table for a single cluster of a `"kccasimple"` object.

Usage

```
## S4 method for signature 'kccasimple':
gcTable(object, which, links, file="gcTable", ...)
```

Arguments

`object`	an object of class "`kccasimple`"
`which`	Number of the cluster.
`links`	Vector of the same length as rows in the data with links to a database.
`file`	File name prefix used for HTML tables. Of the form file-which.html.
`...`	Further arguments can be passed to `write.htmltable`.

Author(s)

Theresa Scharl

See Also

`write.htmltable`

Examples

```
data("hsod")
cl1 <- qtclust(hsod, radius=2, save.data=TRUE)

gcTable(cl1, which=5, links = links_hsod, file = "hsod")
## Not run:
gcExplorer(cl1, theme = "blue", panel.function = gcTable,
           panel.args = list(links = links_hsod, file="hsod"),
           node.function = node.size)
## End(Not run)
```

`go.details` *Functional Group Methods*

Description

Plot or extract size, members or data of a functional group

Usage

```
## S4 method for signature 'data.frame':
go.details(object, mvalues, gn, id, stats, links, gonr,
    source.id, source.group, details = c("size", "names", "id", "data"),
    table = TRUE, file = "go.details", plot = TRUE, cexl = 0.8,
    xlab = "", xlabels = NULL, ylab = "M", ylim = c(-6,6), cex.axis = 1,
    main = NULL, data.type = c("time", "other"), legend = TRUE, ...)
```

Arguments

`object`	An object of class `"data.frame"`.
`mvalues`	Vector giving the columns in `object` which correspond to the gene expression values.
`gn`	Column of `object` which corresponds to the gene names used for representation.
`id`	Column of `object` which corresponds to the unique IDs of the same type as given in `source.id`.
`links`	Column of `object` which corresponds to links to database.
`stats`	Column(s) of `object` which correspond to statistics.
`gonr`	Unique identifier from `source.group` giving the group of genes to be extracted.
`source.id`	Vector of gene IDs assigned to functional groups given in `source.group`.
`source.group`	Vector of the same length as `source.id`.
`details`	The type of details to be extracted.
`table`	Logical. Should an HTML table be created.
`file`	The file where the output of 'gotable' will be written.

`plot`	Logical. Should the genes be plotted.
`cexl`	Point size of the legend.
`xlab`	Label for the x-axis.
`xlabels`	Either a numeric vector of time points giving the positions on the x-axis or a character vector with names of the positions on the x-axis.
`ylab`	Label for the y-axis.
`ylim`	Range of the y-axis.
`cex.axis`	Point size of the axis.
`main`	The main title of the plot or HTML table. If null the name of the functional group is used.
`data.type`	The data type is either on a time scale (default) or something else ("other").
`legend`	Draw a legend?
`...`	Further arguments can be passed to `matplot` or `write.htmltable`.

Author(s)

Theresa Scharl

See Also

`fitsod`

Examples

```
data(fitsod)
data(gobp)

## Plot the functional group
go.details(fitsod, mvalues = 2:9, gn = 31, id = 33, links = 35, stats = 26,
        gonr = "flagellar", source.group = gobp[,3], source.id = gobp[,1],
        plot = TRUE)

## A file named "go.details.html" will be created in the current
## working directory.
go.details(fitsod, mvalues = 2:9, gn = 31, id = 33, links = 35, stats = 26,
        gonr = "flagellar", source.group = gobp[,3], source.id = gobp[,1],
        table = TRUE)

## Names of the genes in functional group "flagellar"
go.details(fitsod, mvalues = 2:9, gn = 31, id = 33, links = 35, stats = 26,
        gonr = "flagellar", source.group = gobp[,3], source.id = gobp[,1],
        details = "names")

## Gene expression values of the functional group
d1 <- go.details(fitsod, mvalues = 2:9, gn = 31, id = 33, links = 35, stats = 26,
        gonr = "flagellar", source.group = gobp[,3], source.id = gobp[,1],
        details = "data")
dim(d1)
```

`Group2Cluster` *Find clusters to a group*

Description

Find the cluster memberships for a group and create the vector of all cluster memberships where the grouped elements are assigned to.

Usage

```
Group2Cluster(object, gonr, source.group, source.id, id)
Random2Cluster(object, perc)
DefinedCluster(object, filt=0, numEdges=6, perc=1, noise=0)
```

Arguments

`object`	An object of class `kccasimple`
`gonr`	Unique identifier from `source.group` giving the group of genes to be extracted
`source.group`	Vector of functional groups where `source.id` are assigned to
`source.id`	Corresponding vector of identifiers to source.group
`id`	Vector of identifiers of the same length as rows in the clustered data of the same type as given in `source.id`

`perc`	For artificial assignment: the percentage of elements in a cluster that should be assigned to the group
`filt`	Edges above this threshold are taken into account
`numEdges`	Number of edges chosen where clusters are assigned similar amount of affected elements
`noise`	The percentage of noise that should be added (i.e., further assigned elements in different clusters)

Value

A vector of cluster memberships.

Author(s)

Theresa Scharl

See Also

`edge.test`

Examples

```
data("hsod")
data("gobp")
set.seed(1111)
cl1 <- qtclust(hsod, radius = 2, save.data = TRUE)

g1 <- Group2Cluster(cl1, gonr = "GO:0009061",
        source.group = gobp[,3], source.id=gobp[,1],
        id = bn_hsod)
table(g1)
```

`jkdist` *Further Distance and Centroid Computations*

Description

Helper functions to create 'kccaFamily' objects.

Usage

```
distJackCor(x, centers)
distJackEuc(x, centers)
distJackMan(x, centers)
distJackMax(x, centers)

centSpline(d)
```

Arguments

`x`	A data matrix
`d`	A data matrix
`centers`	A matrix of centroids

Details

A possible problem using classical distance measures for clustering time–course gene expression data is that single outlier variables can completely change the expression pattern of certain genes. Outliers at special time points are very common in microarray experiments as technical problems like dust or a scratch on the slide can easily distort the data. In such a case these outlier variables can lead to unwanted correlations between genes and to incorrect assignment to clusters. There is a need for distance measures which are robust against outlier variables. The idea of Jackknife (Efron, 1982) distance measures is not to exclude the whole observation for such a gene but rather one or several variables. We want to introduce so–called "Jackknife" distance measures which can handle one outlier time point. The so-called Jackknife correlation was first used by Heyer et al. (1999) to cluster gene expression data. It is defined as

$$d_{xy} = 1 - \min(\rho_{xy}^{(1)}, \rho_{xy}^{(2)}, \ldots, \rho_{xy}^{(T)})$$

where $\rho_{xy}^{(t)}$ is the correlation of pair x,y computed with the t-th time point deleted.

This concept can be extended for the three geometric distance measures Euclidean, Manhattan and Maximum distance. Jackknife Euclidean distance is defined as

$$d_{xy} = \min(d_{xy}^{(1)}, d_{xy}^{(2)}, \ldots, d_{xy}^{(T)})$$

where $d_{xy}^{(t)}$ is the Euclidean distance of pair x,y computed with the t-th time point deleted. Jackknife Manhattan distance and Jackknife Maximum distance can be defined in the same way.

Author(s)

Theresa Scharl

References

Theresa Scharl and Friedrich Leisch: Jackknife distances for clustering time–course gene expression data, in JSM Proceedings 2006

`node.tight` *Node Methods for Neighborhood Graphs*

Description

Several methods how to color nodes of a neighborhood graph.

Usage

```
## S4 method for signature 'kccasimple':
node.tight(object, theme, colscale)
## S4 method for signature 'kccasimple':
node.size(object, theme, colscale)
## S4 method for signature 'kccasimple':
node.go(object, theme, colscale, gonr, source.group, source.id, id)
## S4 method for signature 'kccasimple':
node.group(object, theme, colscale, group)
## S4 method for signature 'kccasimple':
legend.size(object, theme, colscale=NULL, pos="bottomleft")
## S4 method for signature 'kccasimple':
legend.tight(object, theme, colscale=NULL, pos="bottomleft")
```

Arguments

`object`	An object of class `"kccasimple"`
`theme`	A color theme, eg. `theme="blue"`.
`colscale`	Range of luminescence lum of hcl colors, default is min to max.
`gonr`	Unique identifier from `source.group` giving the group of genes to be extracted.

`source.id`	Vector of gene IDs assigned to functional groups given in `source.group`.
`source.group`	Vector of the same length as `source.id`.
`id`	Vector of identifiers of the same length as rows in the clustered data of the same type as given in `source.id`.
`group`	Vector of integers giving the cluster membership of grouped genes.
`pos`	Position where the legend should be placed.

Details

Function `node.size` is used to highlight large clusters where the largest cluster will be assigned the darkest color.

Function `node.tight` is used to highlight tight clusters where the tightest cluster will be assigned the darkest color.

Function `node.go` is used to highlight clusters with accumulation of the functional group given by `gonr` where the highest proportion will be assigned the darkest color.

Function `node.group` is used to highlight clusters with accumulation of a functional group where the class membership are passed by argument `group`. Again the highest proportion will be assigned the darkest color.

Author(s)

Theresa Scharl and Ingo Voglhuber

See Also

`gcExplorer`

Examples

```
data("hsod")
set.seed(1111)
cl1 <- qtclust(hsod, radius = 2, save.data = TRUE)

gcExplorer(cl1, theme = "blue", node.function = node.size,
           legend.pos= "topleft")

gcExplorer(cl1, theme = "red", node.function = node.tight,
           legend.pos= "topleft")
```

```
data("gobp")
gcExplorer(cl1, theme = "green", node.function = node.go,
           node.args = list(gonr = "transport", source.group = gobp[,3],
                            source.id = gobp[,1], id = bn_hsod),
           legend.pos= "topleft")
```

`oxygen` *Preprocessed microarray oxygen deprivation data*

Description

Normalized gene expression microarray data from *E. coli*.

Usage

```
data(oxygen)
```

Format

`oxygen` is a data matrix containing n=43 experiments of various mutants under oxygen deprivation (Covert et al., 2004). The mutants were designed to monitor the response from E. coli during an oxygen shift in order to target the a priori most relevant part of the transcriptional network by using six strains with knockouts of five key transcriptional regulators in the oxygen response (*arcA*, *appY*, *fnr*, *oxyR* and *soxS*). The data was obtained by downloading the corresponding CEL files from the Gene Expression Omnibus (`http://www.ncbi.nlm.nih.gov/geo`) under accession `GDS680` and then normalized using the `rma()` function from the `affy` package. Following the steps described in (Castelo and Roverato, 2008) probesets were mapped to Entrez Gene Identifiers and filtered such that the `ExpressionSet` in the `qpgraph` package names `EcoliOxygen` contains a total of p=4205 genes. Here a subset of the EcoliOxygen data was used containing all genes where Blattner numbers were available.

Note

This data set was taken from Bioconductor package `qpgraph` and modified

Source

Covert, M.W., Knight, E.M., Reed, J.L., Herrgard, M.J., and Palsson, B.O. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92-96, 2004.

Gama-Castro, S., Jimenez-Jacinto, V., Peralta-Gil, M., Santos-Zavaleta, A., Penaloza-Spinola, M.I., Contreras-Moreira, B., Segura-Salazar, J., Muniz-Rascado, L., Martinez-Flores, I., Salgado, H., Bonavides-Martinez, C., Abreu-Goodger, C., Rodriguez-Penagos, C., Miranda-Rios, J., Morett, E., Merino, E., Huerta, A.M., Trevino-Quintanilla, L., and Collado-Vides, J. RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. *Nucleic Acids Res.*, 36(Database issue):D120-124, 2008.

Castelo, R. and Roverato, A. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *J. Comp. Biol.*, 16(2):213-227, 2009.

Examples

```
data(oxygen)
```

`pattern` *Expression pattern*

Description

Expression patterns that can be used to generate artificial gene expression data.

Usage

```
pattern(time = 8, v = 5)
```

Arguments

`time`	number of time points
`v`	absolute value of maximum gene expression

Value

a data matrix

Author(s)

Theresa Scharl

See Also

`gcSim`

Examples

```
cent <- pattern(time=15)
data <- gcSim(sim="pattern", cent=cent)
matplot(t(data),type="l")
```

`ps19` *E. coli Fermentation Data*

Description

E. coli Fermentation Data - Transcription profiling of *E. coli* HMS174(DE3)(pET30aNproGFPmut3.1) - cellular response to limited induction with IPTG

Usage

```
data(ps19)
```

Format

A data frame with 918 observations on the following 10 variables.

`10` Estimated coefficient for a particular gene for the contrast of the sample 10 hours past induction to the sample before induction

`12` Estimated coefficient for a particular gene for the contrast of the sample 12 hours past induction to the sample before induction

`14` Estimated coefficient for a particular gene for the contrast of the sample 14 hours past induction to the sample before induction

`16` Estimated coefficient for a particular gene for the contrast of the sample 16 hours past induction to the sample before induction

`18` Estimated coefficient for a particular gene for the contrast of the sample 18 hours past induction to the sample before induction

`20` Estimated coefficient for a particular gene for the contrast of the sample 20 hours past induction to the sample before induction

`22` Estimated coefficient for a particular gene for the contrast of the sample 22 hours past induction to the sample before induction

`24` Estimated coefficient for a particular gene for the contrast of the sample 24 hours past induction to the sample before induction

`26` Estimated coefficient for a particular gene for the contrast of the sample 26 hours past induction to the sample before induction

`28` Estimated coefficient for a particular gene for the contrast of the sample 28 hours past induction to the sample before induction

Source

Two experiments (including all processing protocols) have been loaded into ArrayExpress (http://www.ebi.ac.uk/microarray-as/ae/). The ArrayExpress accession number of the array design is A-MARS-10. The experiment with fully induced E. coli expression system (ps19) has accession number E-MARS-16 and the experiment with partially induced system (ps17) has accession number E-MARS-17.

References

T. Scharl, G. Striedner, F. Poetschacher, F. Leisch and K. Bayer: Interactive visualization of clusters in microarray data: an efficient tool for improved metabolic analysis of E. coli. *Microbial Cell Factories*, 8:37, 2009.

Examples

```
data(ps19)
```

`sigma` *E. coli Sigma Factors and Global Regulators*

Description

The E. coli sigma factors and the genes they regulate.

Usage

```
data(sigma)
```

Format

A data frame with 1851 observations on the following 6 variables.

`SigmaFactor` a factor with levels `Sigma19` `Sigma24` `Sigma28` `Sigma32` `Sigma38` `Sigma54` `Sigma70`.

`SigmaGene` a factor with levels `fecI` `fliA` `rpoD` `rpoDS` `rpoE` `rpoH` `rpoN` `rpoS` and more.

`RegulatedGeneName` The genename of the regulated genes.

`RegulatedGenebnumber` The Blattner numbers of the regulated genes.

`function` a factor with levels `+`, `-` and `+-`.

`GeneType` a factor with levels `Phantom Gene` `Pseudo Gene`.

Source

http://regulondb.ccg.unam.mx/LicenseRegulonDBd.jsp

References

Salgado H, Gama-Castro S, Peralta-Gil M, Diaz-Peredo E, Sanchez-Solano F, Santos-Zavaleta A, Martinez-Flores I, Jimenez-Jacinto V, Bonavides-Martinez C, Segura-Salazar J, Martinez-Antonio A, Collado-Vides J. RegulonDB (version 5.0): Escherichia coli K-12 transcriptional regulatory network, operon organization, and growth conditions Nucleic Acids Res. 2006 Jan 1;34(Database issue):D394-7

Examples

```
data(sigma)
data(reg)
```

`write.htmltable` *Write a data frame into an HTML table within a HTML page*

Description

Write a `"data.frame"` into an HTML table within a HTML page

Usage

```
write.htmltable(x, filename, title="", sortby=NULL, decreasing=TRUE,
    open="wt", formatNumeric=function(x) paste(signif(x, 3)))
```

Arguments

`x`	`data.frame`.
`filename`	character. File name.
`title`	character. Title of HTML page.
`sortby`	character. Name of column by which to sort the table rows.
`decreasing`	logical. Should the sort order be increasing or decreasing?
`open`	character. This argument is passed on to `file`.
`formatNumeric`	function that takes a numeric and returns a character. This function is called for all numeric values in the table.

Details

This function is taken from package `arrayMagic`.

Value

The function is called for its side effect: writing a file.

Author(s)

Wolfgang Huber `http://www.dkfz.de/mga/whuber`

Examples

```
out = tempfile()

n  = 10
ex = data.frame(genename=paste("Gene", 1:n, sep=""), score=
       signif(16*runif(n)), database=paste("http://super.data.base/?id",
       round(1e9*runif(n)), sep=""))

write.htmltable(ex, out, "Hi there", sortby="score")

cat("Now have a look at ", out, ".html\n", sep="")
```

List of Figures

List of Tables

Bibliography

Abraham, C., Cornillon, P.-A., Matzner-Lober, E., and Molinari, N. (2003). Unsupervised curve clustering using B-splines. *Scandinavian Journal of Statistics*, 30(3):581–595.

Achmüller, C., Kaar, W., Ahrer, K., Wechner, P., Hahn, R., Werther, F., Schmidinger, H., Cserjan-Puschmann, M., Clementschitsch, F., Striedner, G., Bayer, K., Jungbauer, A., and Auer, B. (2007). N^{pro} fusion technology to produce proteins with authentic n termini in *E. coli*. *Nature Methods*, 4:1037–1043.

Androulakis, I., Yang, E., and Almon, R. (2007). Analysis of time-series gene expression data: Methods, challenges, and opportunities. *Annual Review of Biomedical Engineering*, 9:205–228.

Bar-Joseph, Z., Gerber, G., Jaakkola, T. S., Gifford, D. K., and Simon, I. (2003). Continuous representations of time series gene expression data. *Journal of Computational Biology*, 3–4:341–356.

Barrett, T., Troup, D., Wilhite, S., Ledoux, P., Rudnev, D., Evangelista, C., Kim, I., Soboleva, A., Tomashevsky, M., and Edgar, R. (2007). NCBI GEO: mining tens of millions of expression profiles – database and tools update. *Nucleic Acids Res.*, 35:D760–5.

Ben-Dor, A., Shamir, R., and Yakhini, Z. (1999). Clustering gene expression patterns. *Journal of Computational Biology*, 6(3–4):281–297.

Bickel, D. R. (2003). Robust cluster analysis of microarray gene expression data with the number of clusters determined biologically. *Bioinformatics*, 19(7):818–824.

Biernacki, C., Celeux, G., and Govaert, G. (2003). Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. *Computational Statistics & Data Analysis*, 41:561–575.

Caldas, T., Malki, A., Kern, R., Abdallah, J., and Richarme, G. (2006). The Escherichia coli thioredoxin homolog YbbN/Trxsc is a chaperone and a weak protein oxidoreductase. *Biochemical and Biophysical Research Communications*, 343:780–786.

Carey, V. J., Gentleman, R., Huber, W., and Gentry, J. (2005). Bioconductor software for graphs. In Gentleman, R., Carey, V. J., Huber, W., Irizarry, R. A., and Dudoit, S., editors, *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, Statistics for Biology and Health. Springer-Verlag, New York. ISBN 978-0-387-25146-2.

Castelo, R. and Roverato, A. (2009). Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *Journal of Computational Biology*, 16(2):213–227.

Celeux, G. and Govaert, G. (1992). A classification EM algorithm and two stochastic versions. *Computational Statistics & Data Analysis*, 14:315–332.

Chambers, J. and Hastie, T. (1992). *Programming with data: A guide to the S language.* Springer Verlag, Berlin, Germany.

Chipman, H., Hastie, T. J., and Tibshirani, R. (2003). Clustering microarray data. In Speed, T. P., editor, *Statistical Analysis of Gene Expression Microarray Data*, chapter 4, pages 159–200. Chapman & Hall/CRC Press.

Cho, R. J., Campbell, M. J., Winzeler, E. A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T. G., Gabrielian, A. E., Landsman, D., Lockhart, D. J., and Davis, R. W. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2(1):65–73.

Covert, M., Knight, E., Reed, J., Herrgard, M., and Palsson, B. (2004). Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92–96.

Cuesta-Albertos, J., Gordaliza, A., and Matran, C. (1997). Trimmed k-Means: An attempt to robustify quantizers. *The Annals of Statistics*, 25:553–576.

de Hoon, M. J. L., Imoto, S., and Miyano, S. (2002). Statistical analysis of a small set of time-ordered gene expression data using linear splines. *Bioinformatics*, 18(11):1477–1485.

De Smet, F., Mathys, J., Marchal, K., Thijs, G., Moor, B. D., and Moreau, Y. (2002). Adaptive quality-based clustering of gene expression profiles. *Bioinformatics*, 18(5):735–746.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM-algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38.

Diebolt, J. and Ip, E. (1996). Stochastic EM: method and application. In Gilks, W., Richardson, S., and Speigelhalter, D., editors, *Markov Chain Monte Carlo in Practice*. London: Chapman & Hall.

Dürrschmid, K., Reischer, H., Schmidt-Heck, W., Hrebicek, T., Guthke, R., Rizzi, A., and Bayer, K. (2008). Monitoring of transcriptome and proteome profiles to investigate the cellular response of *E. coli* towards recombinant protein expression under defined chemostat conditions. *J. Biotechnol.*, 135(1):34–44.

Efron, B. (1982). *The Jackknife, the Bootstrap, and Other Resampling Plans*. Society for Industrial & Applied Mathematics, Philadelphia.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868.

Fraley, C. and Raftery, A. (1998). How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588.

Fraley, C. and Raftery, A. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631.

Fraley, C., Raftery, A. E., and Wehrens, R. (2005). Incremental model-based clustering for large datasets with small clusters. *Journal of Computational and Graphical Statistics*, 14(3):529–546.

Gentleman, R., Carey, V. J., Huber, W., Irizarry, R. A., and Dudoit, S., editors (2005). *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Statistics for Biology and Health. Springer-Verlag, New York. ISBN 978-0-387-25146-2.

Ghosh, D. and Chinnaiyan, A. M. (2002). Mixture modelling of gene expression data from microarray experiments. *Bioinformatics*, 18(2):275–286.

Grün, B. and Leisch, F. (2008). Flexmix version 2: Finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, 28(4):1–35.

Hakamada, K., Okamoto, M., and Hanai, T. (2006). Novel technique for preprocessing high dimensional time-course data from DNA microarray: mathematical model-based clustering. *Bioinformatics*, 22(7):843–848.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as136: A k–means clustering algorithm. *Applied Statistics*, 128:100–108.

Heyer, L. J., Kruglyak, S., and Yooseph, S. (1999). Exploring expression data: Identification and analysis of coexpressed genes. *Genome Research*, 9:1106–1115.

Hirashima, A., Wang, S., and Inouye, M. (1974). Cell–free synthesis of a specific lipoprotein of the Escherichia coli outer membrane directed by purified messenger RNA. *Proc. Natl. Acad. Sci.*, 71(10):4149–53.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2:193–218.

Jiang, D., Tang, C., and Zhang, A. (2004). Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386.

Karatzoglou, A., Smola, A., Hornik, K., and Zeileis, A. (2004). kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20.

Kaufman, L. and Rousseeuw, P. (1990). *Finding Groups in Data.* John Wiley and Sons, Inc., New York.

Kerr, G., Ruskin, H. J., Crane, M., and Doolan, P. (2008). Techniques for clustering gene expression data. *Comput. Biol. Med.*, 38(3):283–293.

Kohonen, T. (1989). *Self–organization and Associative Memory.* Springer Verlag, New York.

Kthiri, F., Le, H., Tagourti, J., Kern, R., Malki, A., Caldas, T., Abdallah, J., Landoulsi, A., and G.A.A, R. (2008). The thioredoxin homolog YbbN functions as a chaperone rather than as an oxidoreductase. *Biochem Biophys Res Commun.*, 374(4):668–72.

Leisch, F. (2004). FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8):1–18.

Leisch, F. (2006). A toolbox for k-centroids cluster analysis. *Computational Statistics & Data Analysis*, 51(2):526–544.

Leisch, F. (2008). Visualizing cluster analysis and finite mixture models. In Chen, C., Härdle, W., and Unwin, A., editors, *Handbook of Data Visualization*, Springer Handbooks of Computational Statistics. Springer Verlag.

Luan, Y. and Li, H. (2003). Clustering of time-course gene expression data using a mixed-effects model with B-splines. *Bioinformatics*, 19(4):474–482.

Ma, P., Castillo-Davis, C. I., Zhong, W., and Liu, J. S. (2006). A data-driven clustering method for time course gene expression data. *Nucleic Acids Research*, 34(4):1261–1269.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Cam, L. M. L. and Neyman, J., editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, Berkeley.

Martinetz, T. and Schulten, K. (1994). Topology representing networks. *Neural Networks*, 7(3):507–522.

McLachlan, G., Bean, R. W., and Peel, D. (2002). A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics*, 18(3):413–422.

Murrell, P. (2006). *R Graphics*. Tayler & Francis Group.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: analysis and an algorithm. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press.

Parmigiani, G., Garrett, E. S., Irizarry, R. A., and Zeger, S. L. (2003). *The Analysis of Gene Expression Data: Methods and Software*. Springer Verlag, New York.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Ramirez Santos, J., Contreras Ferrat, G., and Gomez Eichelmann, M. (2005). Stationary phase in Escherichia coli. *Rev Latinoam Microbiol.*, 47(3–4):92–101.

Ramoni, M. F., Sebastiani, P., and Kohane, I. S. (2002). Cluster analysis of gene expression dynamics. *Proc. Natl. Acad. Sci. USA*, 99(14):9121–9126.

Rousseeuw, P. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Salgado, H., Gama-Castro, S., Peralta-Gil, M., Diaz-Peredo, E., Sanchez-Solano, F., Santos-Zavaleta, A., Martinez-Flores, I., Jimenez-Jacinto, V., Bonavides-Martinez, C., Segura-Salazar, J., Martinez-Antonio, A., and Collado-Vides, J. (2006). RegulonDB (version 5.0): Escherichia coli K-12 transcriptional regulatory network, operon organization, and growth conditions. *Nucleic Acids Res.*, 34(Database issue):D394–7.

Scharl, T., Grün, B., and Leisch, F. (2009a). Model–based clustering of time–course gene expression data: Evaluation of initialization and random effects. Submitted.

Scharl, T. and Leisch, F. (2006a). Jackknife distances for clustering time–course gene expression data. In *2006 JSM Proceedings*, pages 346–353. American Statistical Association, Alexandria, USA.

Scharl, T. and Leisch, F. (2006b). The stochastic qt-clust algorithm: evaluation of stability and variance on time-course microarray data. In Rizzi, A. and Vichi, M., editors, *Compstat 2006—Proceedings in Computational Statistics*, pages 1015–1022. Physica Verlag, Heidelberg, Germany.

Scharl, T. and Leisch, F. (2008a). Using neighborhood graphs for the investigation of E. coli gene clusters. In Ahdesmäki, M., Strimmer, K., Radde, N., Rahnenführer, J., Klemm, K., Lähdesmäki, H., and Yli-Harja, O., editors, *Proceedings of the 5th International Workshop on Computational Systems Biology, WCSB 2008 (June 11-13, 2008, Leipzig, Germany)*, pages 157–160. Tampere University of Technology, Tampere, Finland.

Scharl, T. and Leisch, F. (2008b). Visualizing gene clusters using neighborhood graphs in r. In Brito, P., editor, *Proceedings of COMPSTAT'2008, International Conference on Computational Statistics, Porto - Portugal, August 24th-29th 2008*, pages 51 –58. Physica–Verlag.

Scharl, T. and Leisch, F. (2009a). gcExplorer: Interactive exploration of gene clusters. *Bioinformatics*, 25(8):1089–1090. doi: 10.1093/bioinformatics/btp099.

Scharl, T. and Leisch, F. (2009b). Quality–based clustering of functional data: Applications to time course microarray data. In Fink, A., Lausen, B., Seidel, W., and Ultsch, A., editors, *Advances in Data Analysis, Data Handling and Business Intelligence, Proceedings of the 32nd Annual Conference of the Gesellschaft für Klassifikation e.V., Joint Conference with the British Classification Society (BCS) and the Dutch/Flemish Classification Society (VOC), Helmut–Schmidt–University, Hamburg, July 16–18, 2008*, Studies in Classification, Data Analysis, and Knowledge Organization. Springer Verlag. Accepted for publication.

Scharl, T., Striedner, G., Pötschacher, F., Leisch, F., and Bayer, K. (2009b). Interactive visualization of clusters in microarray data: an efficient tool for improved metabolic analysis of E. coli. *Microbial Cell Factories*, 8:37.

Scharl, T., Voglhuber, I., and Leisch, F. (2009c). Exploratory and inferential analysis of gene cluster neighborhood graphs. *BMC Bioinformatics*. Accepted for publication.

Scharl, T., Voglhuber, I., and Leisch, F. (2009d). How to use the gcExplorer: Online appendix to the paper "Exploratory and inferential analysis of gene cluster neighborhood graphs". Package vignette.

Serban, N. and Wasserman, L. (2005). Cats: Clustering after transformation and smoothing. *Journal of the American Statistical Association*, 100(471):990–999.

Serres, M., Goswami, S., and Riley, M. (2004). GenProtEC: an updated and improved analysis of functions of Escherichia coli K-12 proteins. *Nucleic Acids Res.*, 32(1):D300–2.

Sheng, Q., Moreau, Y., Smet, F. D., Marchal, K., and Moor, B. D. (2005). Advances in cluster analysis of microarray data. In Azuaje, F. and Dopazo, J., editors, *Data Analysis and Visualization in Genomics and Proteomics*. John Wiley & Sons, Ltd. ISBN 0-470-09439-7.

Smyth, G. K. (2005). Limma: linear models for microarray data. In Gentleman, R., Carey, V. J., Huber, W., Irizarry, R. A., and Dudoit, S., editors, *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, Statistics for Biology and Health. Springer-Verlag, New York. ISBN 978-0-387-25146-2.

Soni, K., Jesudhasan, P., Cepeda, M., Williams, B., Hume, M., Russel, W., Jayaraman, A., and Pillai, S. (2007). Proteomic analysis to identify the role of LuxS/AI-2 mediated protein expression in Escherichia coli O157:H7. *Foodborne Pathog Dis.*, 4(4):463–71.

Speed, T. P., editor (2003). *Statistical Analysis of Gene Expression Microarray Data*. Chapman & Hall/CRC Press.

Striedner, G., Cserjan-Puschmann, M., Pötschacher, F., and Bayer, K. (2003). Tuning the transcription rate of recombinant protein in strong escherichia coli expression systems through repressor titration. *Biotechnol Prog.*, 19(5):1427–32.

Sugai, R., Shimizu, H., K.Nishiyama, and Tokuda, H. (2001). Overexpression of yccL (gnsA) and ydfY (gnsB) increases levels of unsaturated fatty acids and suppresses both the temperature–sensitive fabA6 mutation and cold–sensitive secG null mutation of Escherichia coli. *J Bacteriol.*, 183(19):5523–8.

Tarpey, T. (2003). Clustering functional data. *Journal of Classification*, 20:93–114.

Tarpey, T. (2007). Linear transformations and the k-means clustering algorithm: Applications to clustering curves. *The American Statistician*, 61:34–40.

Thalamuthu, A., Mukhopadhyay, I., Zheng, X., and Tseng, G. C. (2006). Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics*, 22(19):2405–2412.

The Gene Ontology Consortium (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29.

Voglhuber, I. (2008). *Visualization of Centroid–Based Cluster Solutions*. Vienna University of Technology, Austria. Diploma Thesis.

Wehrens, R., Buydens, L. M., Fraley, C., and Raftery, A. E. (2004). Model-based clustering for image segmentation and large datasets via sampling. *Journal of Classification*, 21:231–253.

Yeung, K. Y., Fraley, C., Murua, A., Raftery, A. E., and Ruzzo, W. L. (2001). Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987.

Zeileis, A., Meyer, D., and Hornik, K. (2007). Residual-based shadings for visualizing (conditional) independence. *Journal of Computational and Graphical Statistics*, 16(3):507–525.

Curriculum Vitae

Personal Data

Born:	November 11, 1979 in Kirchdorf a. d. Krems, Austria
Nationality:	Austria

Education

1990–1998	High School, BG Steyr Werndlpark, Austria
1998–2004	University studies in Applied Mathematics ("Technische Mathematik"), Vienna University of Technology
since 2004	Ph.D. Studies in Applied Mathematics, Vienna University of Technology

Career History & Work Experience

07/2002–12/2002	AXA Versicherungen AG, 1010 Vienna, Area "Privatkunden Mathematik"
07/2003–09/2004	UNIQA Versicherungen AG, 1021 Vienna, Area "Mathematik LV/UV"
01/2004–09/2004	Diploma Thesis at the Department of Bioresources and Molecular Diagnostics, ARC Seibersdorf Research GmbH.
since 11/2004	Research Assistant, Department of Statistics and Probability Theory, Vienna University of Technology and Department of Biotechnology, University of Natural Resources and Applied Life Sciences, Vienna