Interactive analysis of high–dimensional association structures with graphical models

Iris Pigeot[1], Angelika Blauth[2], François Bry[2]

[1]Institute of Statistics, University of Munich, Ludwigstr. 33, D–80539 München
[2] Institute of Computer Science, University of Munich, Oettingenstr. 67, D–80538 München

Author for correspondence:
Iris Pigeot, Institute of Statistics, University of Munich, Ludwigstr. 33, D–80539 München
e–mail: pigeot@stat.uni-muenchen.de

Short running title:
Interactive analysis of graphical models

Interactive analysis of high–dimensional association structures with graphical models

Iris Pigeot[1*], Angelika Blauth[2*], François Bry[2]

[1]Institute of Statistics, University of Munich, Ludwigstr. 33, D–80539 München
[2] Institute of Computer Science, University of Munich, Oettingenstr. 67, D–80538 München

Abstract

Graphical chain models are a capable tool for analyzing multivariate data. However, their practical use may still be cumbersome in some respect since fitting the model requires the application of an intensive selection strategy based on the calculation of an enormous number of different regressions. In this paper, we present a computer system especially designed for the calculation of graphical chain models which is not only planned to automatically carry out the model search but also to visualize the corresponding graph at each stage of the model fit on request by the user. It additionally allows to modify the graph and the model fit interactively.

KEY WORDS: Chain graph, graphical model, object–orientation, selection strategy, statistical software package

1 Introduction

Empirical studies typically involve a huge number of different variables being collected on the subjects of interest. Although the primary research question addresses in general the explanation of one or more response variables by certain explanatories, it is often also of interest to determine the complete association structure among all variables, which cannot be modelled within most multivariate models. In addition, many models are not adequate to cope with situations where indirect influences should be investigated, i.e. situations where we allow for explanatories influencing certain other explanatories which in turn directly or indirectly have an impact on the responses. Besides the fact that the analysis of such high–dimensional data structures is of course rather complex and time–consuming another important aspect concerns the representation of the resulting model, which should be comprehensible to the data–holder, and of the results, which should be easy to communicate.

The problems related to the modelling and representating of complex association structures can be tackled within the framework of graphical models which have been established mainly by works of Cox, Lauritzen, Pearl, and Wermuth (cf. Darroch, Lauritzen, Speed, 1980; Lauritzen, Wermuth, 1989; Wermuth, Lauritzen, 1990; Cox, Wermuth, 1993; Wermuth, Cox, Pearl, 1998). Graphical models combine a statistical model with its representation as a graph where the underlying key concepts are conditional and marginal

*Financial support by the SFB 386, Deutsche Forschungsgemeinschaft is gratefully acknowledged.

independence between the variables incorporated in the analysis. For introductory books in the theoretical methodology of graphical models see for instance Whittaker (1990) and Lauritzen (1996). The book by Cox and Wermuth (1996) additionally provides approaches for model fitting strategies and hints for interpreting graphical models. An application–oriented introduction is given by the book of Edwards (1995), which also serves a manual for his software package MIM (see also below).

A major disadvantage of graphical models concerns their practical application, since the related model fit requires an exhaustive calculation of different regression models and the use of adequate model selection strategies. Here, it would be desirable to have a software which does not only enable the user to calculate the involved regressions efficiently but also draws the corresponding graph simultaneously. Of course, further features such as interactively modifying the graph obtained from the analysis may be thought of. Currently, a software package, called MIM (cf. Edwards, 1987), is available which allows the treatment of undirected graphs. Algorithms for handling directed graphs are still under development by Edwards. As further software packages DIGRAM (Kreiner, 1986, 1992), CoCo (Badsberg, 1992), and TURNER (Lauer, 1998) should be mentioned. DIGRAM calculates among others Pearson's χ^2–statistics and partial γ–coefficients for checking the hypothesis of conditional independence. It also supports Monte–Carlo algorithms for calculating exact p–values of the conditional distribution of the test statistics given the sufficient marginals in multidimensional contingency tables (Kreiner, 1987). CoCo enables the analysis of complete contingency tables and contains a large number of exact conditional tests for decomposable models as well as various methods for model search. TURNER allows the interactive modelling of discrete data via loglinear models. These are visualized by graphical hierarchical loglinear models (cf. Whittaker, 1990), whereas several enhancements are integrated. Our research project stresses on the extension of graphical models under a theoretical point of view as well as with respect to their applicability. Especially, concerning the latter we are working on a computer software which is intended to allow the interactive analysis of graphical models with special focus on graphical chain models. The system consists of different components which mainly address three different demands. First, the system is planned to carry out the data analysis by fitting an adequate statistical model. Here, the system should offer several competing selection strategies. Thus, the data–holder can compare various resulting models regarding the underlying substantive research question. Second, the fitted graphical model is to be visualized where different ways of manipulating the graph should be possible. Some of these manipulations may require a restart of the model fitting. Third, it is intended to incorporate a help system which should assist the user of the computer system at the different stages of the data analysis. The complexity of such interactive systems with graphical interfaces makes it desirable to develop them with up–to–date software development methodologies. Thus, as additional feature, the program is based on an object–oriented analysis and design which makes it easier to adapt the system to new statistical methods and to extend it with those. The representation of its modelling is based on the **u**nified

modelling **l**anguage, briefly denoted as UML. For the implementation the development environment Delphi is used. It consists of the program language object pascal together with the possibility of a visual programming of the static structure.

In the following, we would like to present the concept and basic structure of our computer system, where the paper is organized as follows. The theoretical background of graphical models and basic notations are reviewed in Section 2. Section 3 describes the basic concept of the program. Among others the modelling of the user interface is explained and the different components of the program are presented. The next section deals with the realization of the graphical representation of the underlying statistical model within the program. The different possibilities to manipulate the graph are discussed. The fifth section shows in detail the different steps of the program before the actual calculation of the model starts. In the discussion some open problems are mentioned.

2 Some basic features of graphical models and their practical use

For convenience, let us first recall the basic notions and the theoretical background of graphical models. The main idea of graphical models is the graphical representation of a multivariate association structure of a random vector, where especially marginal or conditional independencies are reflected in the graph. For this purpose, it has first to be clarified how to represent a statistical model by a graph and second how to ensure that certain statistical properties may be directly read off the graph.

Following the terminology introduced by Lauritzen and Wermuth (1989) $X_V = (Y', I')'$ denotes a random vector with Y consisting of R continuous random variables and I of Q discrete variables. The realizations of Y are given as $y \in I\!R^R$ and the set of all possible realizations i of I is given as $\mathcal{I}$. Accordingly, the index set V consists of two disjoint subsets Γ and Δ, i.e. $V = \Gamma \cup \Delta$, $\Gamma \cap \Delta = \emptyset$, with Δ denoting the index set of discrete and Γ that of continuous components. As most common and probably also most important family of distributions for such a mixed random vector consider the conditional Gaussian (CG–) distributions which assume a multivariate normal distribution of Y given $I = i$. That means the joint density function $f(x_V)$ factorizes as the discrete marginal probabilities $P(I = i) = p(i) > 0$ and the density of a multivariate distribution with mean vector $\mu(i) \in I\!R^R$ and covariance matrix $\Sigma(i) \in I\!R^{R\times R}$, where $\Sigma(i)$ is assumed to be positive definite for all $i \in \mathcal{I}$. If we further assume that $\Sigma(i)$ does not depend on i, X_V is said to have a homogeneous CG–distribution.

Now a graph $\mathcal{G} = (V, E)$ is given by a nonempty finite set V of vertices representing the random variables and a set $E \subset V \times V$ of edges representing the associations between pairs of variables. An edge is called undirected if $(a, b) \in E$ and $(b, a) \in E$. Correspondingly, an edge is directed if $(a, b) \in E$, but $(b, a) \notin E$. In the graphical representation, we use

dots for the **d**iscrete and **c**ircles for the **c**ontinuous variables. Undirected edges are drawn as lines, directed edges (a, b) as arrows pointing from a to b, where an undirected edge reflects a symmetric association between X_a and X_b and a directed one an asymmetric association with X_a being regarded as explanatory for X_b. If two vertices are connected by an undirected edge, they are called neighbours, where the neighbours of a are denoted as $ne(a)$. In case, a and b are connected by a directed arrow pointing from a to b, a is said to be parent $pa(b)$ of b and b is called child $ch(a)$ of a. Based on these terms the boundary $bd(A)$ of $A \subset V$ is defined as $\cup_{a \in A}\{pa(a) \cup ne(a)\} \backslash A$ and its closure $cl(A)$ is obtained as $A \cup bd(A)$. Connecting more than two vertices a sequence of different vertices $a_1, \ldots, a_n$ with $(a_{i-1}, a_i) \in E, i = 2, \ldots, n$, is called a path. If there is a path leading from a to b, but not vice versa, a is an ancestor $an(b)$ of b and b an descendant $de(a)$ of a. For three disjoint subsets, say A, B, C of V, C is said to separate A and B if each path from $a \in A$ to $b \in B$ intersects C.

The above definitions are used in the following to characterize different types of graphs. For instance, a graph is called undirected if $\mathcal{G}$ only consists of undirected edges, otherwise it is directed. It is said to be complete if all vertices of V are connected by an edge. A clique is a maximally complete subgraph, where a subgraph is induced by any subset $A \subseteq V$ and is denoted as $\mathcal{G}_A = (A, E_A)$ with $E_A = E \cap (A \times A)$. Furthermore, a decomposition of a graph $\mathcal{G}$ is given as a partition (A, B, C) of V, where C separates A and B, C is complete, and $C \subseteq \Delta$ or $B \subseteq \Gamma$. Then, $\mathcal{G}$ is decomposable if it is complete or if there exists a decomposition (A, B, C) with A and B both nonempty into decomposable subgraphs $\mathcal{G}_{A \cup C}$ and $\mathcal{G}_{B \cup C}$. This last property is of special importance when calculating the maximum–likelihood estimators on which we focus later on.

Now coming back to the related statistical models, marginal or conditional independencies can be read off the graph, if the family of distributions fulfills certain so–called Markov properties. In general, independencies are represented by missing edges, where in case of marginal dependencies those are represented in the graph by broken lines and conditional dependencies by solid lines. Graphs reflecting conditional dependencies are called concentration graphs and those reflecting marginal dependencies covariance graphs. Of course, it is now of interest how to interpret missing edges with respect to the underlying family of multivariate distributions. Here, again it is necessary to distinguish between directed and undirected graphs, where chain graphs play a special role among the directed graphs. Chain graphs are based on a partition of V in disjoint chain components $V_1, \ldots, V_T$, where edges within one component are always undirected reflecting symmetric associations between each pair of variables within this component and edges between variables belonging to different components are always directed. Here, it is convention to order the components from right to left that is directed edges are always pointing from variables further right to variables further left where all variables of one component are ordered within one box. Thus, the very right box contains pure influence variables and the very left box the pure responses. In between, we find all variables which are simultaneously responses and explanatories, the so–called intermediates. For further details see for instance Cox and Wermuth (1993). An undirected graph is a special chain graph consisting only of one single box. Regarding the definition of the Markov properties of a family of distributions for chain graphs we also have to introduce the notion of a moral graph $\mathcal{G}^m$ which can be obtained from a chain graph by replacing all directed edges with undirected ones and by marrying parents, i.e. by connecting vertices by an undirected edge if they have children in the same chain component (Frydenberg, 1990a).

Since we focus on conditional independencies, let us now briefly review the definitions of the Markov properties first for undirected concentration graphs and then for directed graphs, i.e. here for chain graphs. A family of distributions $\mathcal{P}$ of X_V fulfills (a) the pairwise Markov property with respect to an undirected concentration graph $\mathcal{G}$ if

$$X_a \perp X_b | X_{V \setminus \{a,b\}} \text{ for all } (a,b) \notin E, \tag{2.1}$$

(b) the local Markov property if

$$X_a \perp X_{V \setminus cl(a)} | X_{bd(a)} \text{ for all } a \in V, \tag{2.2}$$

(c) the global Markov property if for each C which separates A and B

$$X_A \perp X_B | X_C \text{ for all disjoint } A, B, C \subseteq V. \tag{2.3}$$

The weakest Markov property is the pairwise. It only allows to interpret missing edges in the graph as conditional independencies between pairs of random variables being not connected by an edge. It would of course be desirable to also read off the graph further conditional independencies than only the pairwise. Here, it can be shown that the above properties are all equivalent if the density of $\mathcal{P}$ is strictly positive (Lauritzen, 1996, p. 36). In that case, missing edges can thus also be interpreted in the sense of the global Markov property. Defining the Markov properties for chain graphs we have to account for their blockrecursive structure caused by the ordering of the chain components. Thus, a family of distributions $\mathcal{P}$ of X_V fulfills

(a) the blockrecursive Markov property with respect to a chain graph $\mathcal{G}$ if

$$X_a \perp X_b | X_{V_b \setminus \{a,b\}} \text{ for all } (a,b),(b,a) \notin E, a \in V_i, b \in V_j, i \leq j, V_b = \cup_{k=1}^{j} V_k, \tag{2.4}$$

(b) the pairwise Markov property if

$$X_a \perp X_b | X_{\{V \setminus de(b)\} \setminus \{a,b\}} \text{ for all } (a,b),(b,a) \notin E, a \notin de(b) \tag{2.5}$$

(c) the local Markov property if

$$X_a \perp X_{\{V \setminus de(a)\} \setminus cl(a)} | X_{bd(a)} \text{ for all } a \in V, \tag{2.6}$$

(d) the global Markov property if

$$X_A \perp X_B | X_C \text{ for all } A, B, C \subseteq V, \tag{2.7}$$

where C separates A and B in $\mathcal{G}^m_{an(A\cup B\cup C)}$ with $an(A \cup B \cup C)$ denoting the smallest set of ancestors which contains $A \cup B \cup C$. The last condition points out that chain graphs contain more information concerning the underlying distribution than their corresponding moral graphs. For chain graphs it is not as simple as for undirected concentration graphs to verify the conditions for the equivalence of the Markov properties. Of course, the global Markov property is again strongest implying the local which in turn implies the pairwise and then in turn the blockrecursive Markov property. It can, however, be shown that the equivalence of these four Markov properties holds for CG–distributions (Frydenberg, 1990a).

We already mentioned that the properties of the graph cannot only be used for reading off certain independencies but that they are also very useful concerning the statistical analysis of a multivariate data set as for instance to simplify the underlying estimation problem. Let us here reproduce only one important result which goes back to Frydenberg and Lauritzen (1989) where we focus on undirected concentration graphs with CG–distribution fulfilling the pairwise Markov property, briefly called $\mathcal{G}$–Markovian. Let $\mathcal{M}(\mathcal{G})$ denote the statistical model containing all $\mathcal{G}$–Markovian CG–distributions and $\mathcal{M}(\mathcal{G})_A$ the set of A–marginals. The latter model has to be carefully distinguished from $\mathcal{M}(\mathcal{G}_A)$ which denotes the set of all $\mathcal{G}_A$–Markovian CG–distributions, i.e. with respect to the subgraph $\mathcal{G}_A$. Furthermore, we need the set of all conditional $\mathcal{G}$–Markovian CG–distributions conditioning on X_A which we denote as $\mathcal{M}(\mathcal{G})^A$. Given a decomposition (A, B, C) of $\mathcal{G}$ the graph is collapsible onto $A \cup C$ which means that the set of $A\cup C$–marginals coincides with the set of all $\mathcal{G}_{A\cup C}$–Markovian CG–distributions, i.e. $\mathcal{M}(\mathcal{G})_{A\cup C} = \mathcal{M}(\mathcal{G}_{A\cup C})$ (Frydenberg, 1990b). Frydenberg and Lauritzen (1989) show that then the whole maximization task can be subdivided into smaller ones. Thus, it holds for the ML–estimator of the joint density f, that it can be obtained as

$$\hat{f} = \hat{f}_{[A\cup C]}\hat{f}_{[B|C]} = \frac{\hat{f}_{[A\cup C]}\hat{f}_{[B\cup C]}}{\hat{f}_{[C]}}, \tag{2.8}$$

where for instance $\hat{f}_{[A\cup C]}$ denotes the ML–estimator of f in $\mathcal{M}(\mathcal{G}_{A\cup C})$, i.e. with respect to the subgraph $\mathcal{G}_{A\cup C}$.

After giving the above brief summary of the theoretical background of graphical models, where some special results have been highlighted of course without any claim for completeness, the question arises how these models can be used for practical purposes. Here, the most challenging problem concerns the model fit. In a first step, it has to be decided which type of model is most appropriate to analyze the research question. That means, are we interested in the investigation of conditional or marginal independencies and is it more adequate to consider a directed or an indirected graph. As already addressed we restrict ourselves in this paper on concentration graphs modelling conditional independencies. Since MIM is a software for analyzing multivariate data with graphical models based on undirected graphs (for various examples concerning its application we refer to Edwards, 1995), we focus in the following on the fitting of chain graphs. Fitting chain graphs we can use the fact that the joint density factorizes into a product of several conditional densities and one marginal density with respect to $\mathcal{G}$ as follows

$$f_V = f_{V_T|V_{T-1}\dots V_1} \cdot f_{V_{T-1}|V_{T-2}\dots V_1} \cdots f_{V_2|V_1} \cdot f_{V_1}. \tag{2.9}$$

Assuming now a conditional CG–distribution for each conditional density, a so–called CG–regression, the joint distribution of X_V is called a recursive multivariate CG–regression (Lauritzen, Wermuth, 1989).

In an empirical study, this dependence chain, i.e. the partition of V into the components $V_1, \dots, V_T$, is postulated by the researcher which then results in the above factorization of the likelihood. Thus, this factorization can also be justified from subject–matter knowledge. Maximizing the joint likelihood it would be desirable if a separate maximization of the single factors is possible, which would yield an analogous result to (2.8). However, this has not been shown yet for chain graphs, although there are ideas that this presumably holds in very restrictive cases assuming a recursive multivariate CG–regression as underlying distribution of X_V. Of course, any other multivariate distribution fulfilling the equivalence of the above listed Markov properties could be assumed instead but up to now no other distribution besides the recursive multivariate CG–regression is known to fulfil the required equivalence and to cope with such a mixture of continuous and discrete random variables. But unfortunately an algorithm for fitting CG–regressions as a whole is not available at the moment, although there is one, as already mentioned, under current development by Edwards and which is intended to be included in MIM. Another possible, more heuristic approach goes back to Cox and Wermuth (1996) who suggest a data–driven strategy. Here, the problem to find an adequate graph for the given data is solved via a system of univariate regressions. For each variable in turn a regression model is adopted, which means, that all other variables in the same chain component and all those in components with lower indices serve as potential influences for this particular response.

A selection strategy which consists of several backward and forward regressions is used to find the important influences out of all potential ones. These influences are connected to the response by edges in the graph. To detect relevant interaction terms and non-linearities, which could also be possible influences, a screening is performed in advance. Of course, such a procedure is not only cumbersome to handle, but also hard to justify from a theoretical point of view since among others it does not ensure the validity of the equivalence of the Markov properties for the resulting graph. This means, that missing edges in the final graph have to be interpreted with caution, where an edge is drawn if the corresponding regression coefficient is statistically significant without adjusting for the multiplicity of the underlying statistical test problem. Nevertheless, it is a reasonable heuristic strategy, close to the data, and alternatives seem to be absent. For a discussion of the theoretical problems related to this strategy and a detailed example for its application also illustrating its practical inconveniences see for instance Caputo, Heinicke, and Pigeot (1997).

Although this selection strategy is tractable somehow, it typically requires fitting a large number of different regression models and is therefore very time–consuming and inefficient in daily use. This was one reason for our research project concerning the development of a software which should enable the user to calculate especially graphical chain models. Another inconvenient aspect related to the fitting of chain graphs concerns its graphical representation which has to be carried out by hand at the moment. Our software should also automatically draw the resulting graphs after the different steps of the model fit.

3 Overview of the program design

Let us first introduce some important terms of object–oriented software engineering for understanding the organization of the program. For more details on this subject see for example Booch (1997).

A class is an abstraction of a real world element. It describes a set of objects, which have nearly identical structure, behaviour, and relations. The main task of object–oriented design is to find proper classes to represent the real world. Additionally, different classes have to be brought into a ordering giving an overview of the detected classes. A hierarchical relationship between classes in which a class has the same structure or behaviour as another class is called inheritance. It defines a so–called "is–a"–hierarchy between classes in which a subclass inherits from more or several superclasses. Generally, a subclass specializes a superclass by adding new features. The relation between a class and its parts, which are also classes, is called aggregation. This relationship forms a "part–of"–hierarchy. In object–oriented software engineering the elements of the model are usually visualized by diagrams. We choose the UML notation for this task, which is a quasi–standard in the industry.

A main aspect of the program design is to find a basic structure, which is consistent within itself and easy to expand. In our program the organization is as follows. It is logically divided into five parts, called packages, where every part handles another aspect of the program. The scheme is shown in Figure 1.

The package graphical elements consists of the classes which are needed to visualize the graphical model and the dependence chain. This will be discussed in more detail in Section 4. The selection strategies together with the corresponding functions form the package mathematical functions. Until now, only the strategy due to Cox and Wermuth (1996), which was briefly introduced in Section 2, is conceptually acquired. The dialogs concerning the user interaction with the program can be found in the package dialogs. Classes, which are only needed for a small purpose as e.g. to read in the data, are stored in the package utility classes. The package data structures constitutes the core package of the system. It contains the structure of the graphical model together with classes which store the data. Its structure is shown in Figure 2.

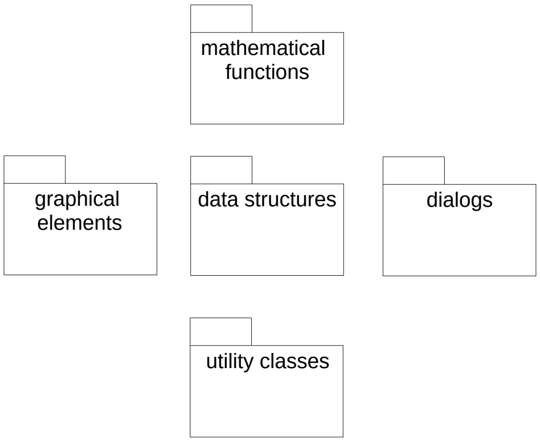

Figure 1: Basic packages

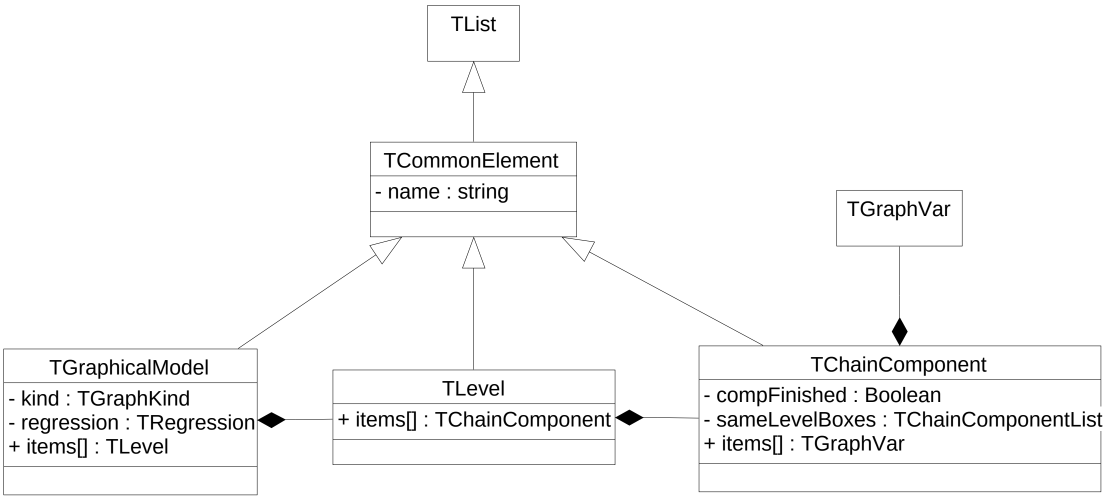

Figure 2: Modelling the graphical model

The class TGraphicalModel is the abstract representation of a graphical model. It is constructed such that all kinds of graphical models can be realized. Like all other classes which reflect the structure of the model it is a subclass of TCommonElement. This relation is shown in the diagram by a line with the triangle on the side of the superclass. TCommonElement is a dynamic list and is therefore derived from the class TList. Dynamic lists are used to collect a dynamic number of all different types of classes. TGraphicalModel consists of one or more classes of the type TLevel. In the diagram, this "part–of"–relation is shown by a line with the rhomb on the other side of the part. TLevel represents the partial ordering of the different chain components. Usually, TLevel consists of only one object of the class TChainComponent, which represents the chain components. In certain situations, two or more chain components may be ordered on the same level but clearly separated from each other since they are considered as independent because of subject–matter knowledge. Such chain components are called “stacked boxes”, because in the visualization of the graph they are drawn one upon the other. In this case a TLevel class has more than one TChainComponent objects. The main elements of chain components are the variables themselves which are ordered in the particular chain components. Thus, the class TChainComponent consists of one ore more objects of the class TGraphVar, which models the variables shown in the graph.

Besides the above aspects which mainly address the modelling of the basic structure of the program one other important criterion in its development concerns its user interface which has steadily increased in importance since the beginning of the computer era. In contrast to the first computer programs, nowadays standard is such that their handling has to be easy and comfortable for the user where already since a couple of years there has been made much effort to come to a standardization of user interfaces. Here, the German Institute for Standardization has made the running for other institutions. Thus, their guidelines how to build up a user interface already date back to 1988. These guidelines (Deutsches Institut für Normung, 1988) have been followed in the design of our program. Instead of going into details we only mention one special aspect concerning the robustness of the program with respect to possible errors. This implies that the program should be able to realize possible errors at a very early state and to react in an appropriate way as for instance by warning the user and stopping the current process while asking the user to adjust his/her input. Thus, all dialogs are designed such that they check the input regarding potential errors early and that they give warning as precise as possible for the user. Of course, it is additionally taken into account that certain types of errors cannot occur at all. For instance, dialogs are typically conducted by using list boxes or buttons to protect the user against wrong inputs. This also means that the user is informed by the program if his/her action leads to essential changes of the current process as for instance in case he/she changes the measurement scale of certain variables while the model fit is running.

The main window contains a title bar, a menu bar, a tool bar for commands frequently executed, and a status bar giving information on the currently chosen menu items. The working area of the main window remains empty and serves as frame for windows occurring during the execution of the program. These are on the one hand dialogs which enable the user to carry out certain actions and on the other hand documents which depict the graph at the different stages of the model fit. The documents depicting the graph are designed as **m**ultiple **d**ocument **i**nterface–child windows which means that more than one can be simultaneously opened and handled. A further advantage of the MDI–concept is that the structure of the main menu of the program can be dynamically changed, what means that in our case, the main menu bar is supplemented by various items when graphs are shown.

The main window consists of the following menus `File`, `View`, `Analysis`, `Descriptive analysis`, `Window`, and `Help`. Within the `File` menu files can be saved, loaded as well as printed and the application can be finished. `View` allows to show important structures of the data such as the raw data, the measurement scale, and the dependence chain. `Analysis` contains all items which may be addressed to construct a graph and to protocol the statistical results. Within the menu `Descriptive analysis` the most important aspects of a descriptive analysis of the data can be performed. Choosing the menu `Window` the different MDI–children can be ordered. `Help` offers the help function of the program and gives information on the current version as usual for windows applications. A screen shot of the main window is shown in Figure 3.

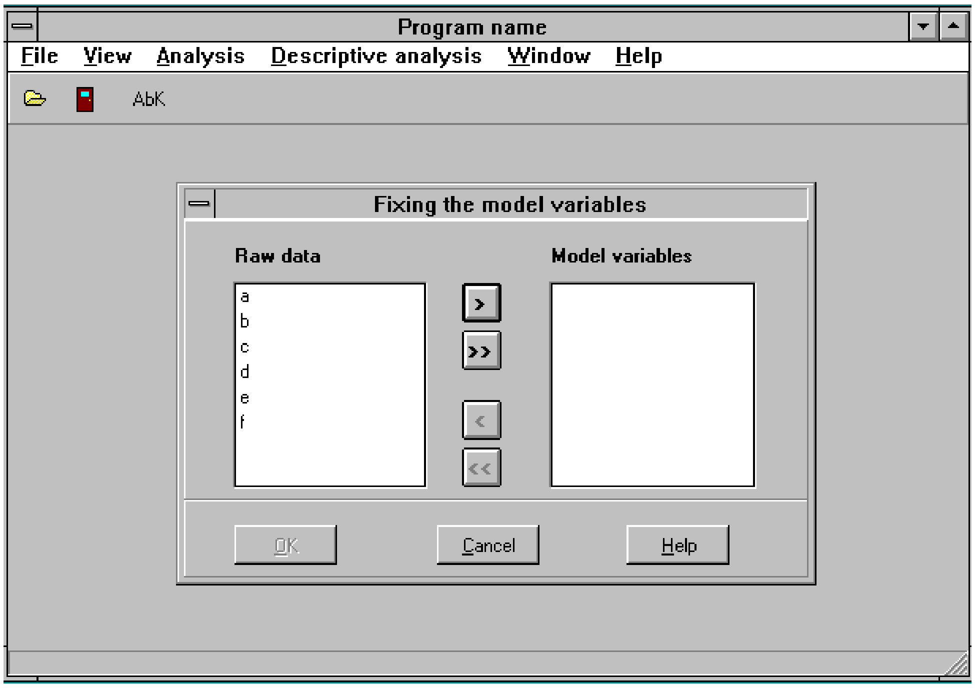

Figure 3: Main window of the program

The next section focuses on one of the most important aspects mentioned above, namely the modelling of the graph and its graphical representation.

4 The representation of the graph

The main component of the program concerns the visualization and interactive fitting of a graphical model, since here the advantages of using a program become obvious. To make entirely use of these advantages, the user should additionally get the chance to modify or to extend the process of model search by hand, where the handling has to be easy. Modelling the graph within the program, it has to be taken into account, that the elements of the graph have to be designed flexibly to come up to various applications. This approach does not only go in line with the object–oriented paradigm but it is also important within the design of our program because there are three different kinds of representations of graphs in the program, namely the dependence chain, the graph at each step of the model fitting process, and the resulting graphical model, whereas the latter two split up further into concentration and covariance graphs. One aim of modelling the visualization of the graphs is to find the common elements of these types for building up a class hierarchy. Another important aspect is to realize the visualization independently of the mathematical representation of the graphical model to make extensions concerning only one part easier, as for instance regarding the inclusion of a new selection strategy. In this section, we briefly describe the structure of the visualization of the graphical model and its interaction with the mathematical representation.

The representation of a chain graph consists of several boxes, arrows, and at the model fitting stage, of one double–box. As already described in Section 2, the boxes represent the chain components, the arrows the directed associations, and the double–box contains all variables the association structure of which is not of interest at the moment and therefore not further investigated. Each box contains all variables of the corresponding chain component and their undirected association structure. Regarding its representation this means that a box contains nodes and lines. The classes for the visualization of the graphs rebuild this structure. A simplified representation of the visualization of the graphical model which is at the same time used for representing the resulting graph at each step of the model fit is shown in Figure 4. For ease of understanding some classes which are not important for explaining the structure are omitted.

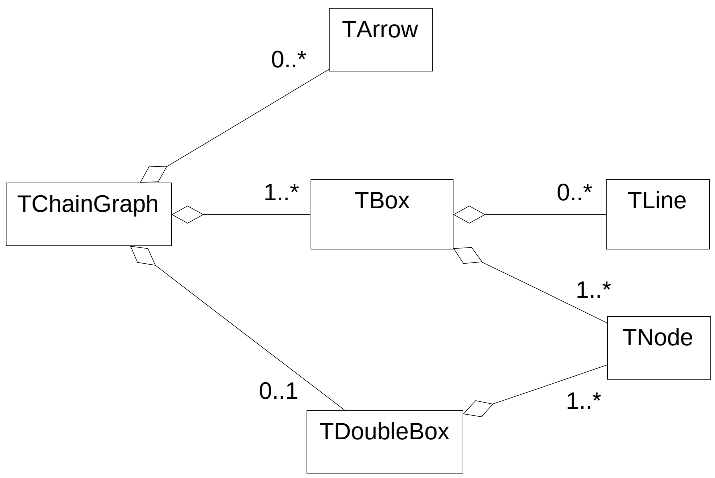

Figure 4: Visualization of the graphical model

The classes have the same names as the elements they represent but with an initial letter 'T' which is a commonly used notation for classes in the programming environments of Borland. The numbers on the one side of the aggregations specify the allowed quantity of classes which can be part of the class on the side of the rhomb, where $*$ denotes an arbitrary number of classes. For example, the class TChainGraph can have zero or one quantities of the class TDoubleBox depending on if the graph is shown in its final or in its model fitting stage.

The translation from the elements of the representation to the elements of the mathematical view of graphical models is performed by the class TTranslator. Table 1 shows the most important connections, which are handled by TTranslator. We will give some explanations of interesting entries.

Graphical model	**Visualization**
TGraphicalModel	TChainGraph
TGraphicalModel.kind	TChainGraph.lineStyle
TChainComponent	TChainBox
TGraphVar	TNode
TGraphVar.influenceOf	TArrow
TGraphVar.directAssoc	TLine
TGraphVar.scale	TNode.style

Table 1: Connections between a graphical model and its visualization

With the attribute TGraphicalModel.kind the kind of a graphical model, i.e. a covariance or a concentration graph, is realized. In its graphical representation this is indicated by broken lines in the case of a covariance graph and by solid lines in the case of a concentration graph (cf. Section 2). This is modelled with the attribute TChainGraph.lineStyle. The class TGraphVar has two attributes TGraphVar.influencesOf and TGraphVar.directAssoc including all those variables which are influenced in an undirected or directed way by the selected variable. According to the graphical representation of symmetric associations as lines and of asymmetric associations as arrows we have two classes TLine and TArrow, respectively. The scale of a variable gives information on how to draw the corresponding node. In case it is continuous it is drawn as circle and for discrete variables we use a dot (cf. Section 2) which connects the attribute TGraphVar.scale with the attribute TNode.style.

As an important part of the model fit the program offers the possibility to represent the graph at each stage of this process. As already mentioned in the previous section, this is realized by MDI–child windows. When the graph is depicted on the screen, the menu `File` is extended by items for saving and printing the depicted graph. In addition, we find new menus in the main menu `Calculation` and `Modification`. `Calculation` consists of two items. With one of those the calculation can be continued and the other enables the user to undo the last step of the calculation. The menu `Modification` contains all items which allow to change the representation of the graph and to interrupt its automatic calculation. Here, two different modes have to be distinguished, named `information mode` and `revision mode`. Both will now be described in detail.

The information mode enables the user to modify the graph to come to a clearer representation and to get additional information on the corresponding statistical model. For this purpose, the information mode consists of several items which can be chosen by the user from the menu or the tool bar. These include the following features.

Standard can be used to enlarge or to reduce the visible part of the graph centered on the chosen element where chain components or variables can be marked by clicking the mouse button. Furthermore, the name of the marked element can be changed. In addition, chain components as well as variables can be moved by holding down the left mouse button, where all edges, i.e. arrows or lines, being connected with the elements to be rearranged are simultaneously moved. Such actions are of course restricted on moves which do not change the model being already calculated. This means, variables can only be rearranged within one component, and chain components cannot be moved such that their ordering within the chain will be different afterwards.

Hide allows to hide the corresponding element of the graph by clicking the mouse button. This is visualized in the graph by a triangle. If chain components or variables are hidden all directly connected edges will not be shown any longer either. Clicking on a hidden element makes it visible again.

Label gives additional information on a marked element.

Colour shows a dialog in which a colour can be selected to draw a marked element.

Details gives more detailed information on a marked element within an extra window. Regarding variables for instance minimum, maximum, and the arithmetic mean may be displayed and regarding edges this may be the direction of the association.

Highlighting elements enables the user to emphasize parts of the graph such as certain paths or vertices by colours, where the parts to be coloured have first to be fixed in a dialog.

Within the revision mode the user can interrupt the calculation of the model by changing the structure of the graph, i.e. he/she has the possibility to add or to delete edges. In both cases, a warning is given that parts of the model fit have to be carried out again when the calculation of the model is to be continued. Note that this mode is not provided for moving variables from one chain component to another since this could imply such fundamental changes that for instance the data–screening has to be restarted. If the user wants to modify the ordering of the variables he/she has to build up a new dependence chain and to restart the complete calculation from the beginning. Here, we have the following possibilities.

Add an edge makes it possible to draw an edge between two variables after having selected one variable after the other. This edge will then be retained in the model in all following steps of the model fit. To distinguish this edge from those obtained by the applied selection strategy it is coloured in red.

Remove an edge allows to delete an edge from the graph after having it marked. To distinguish it from those edges which have been removed in course of the model fit it is represented as a red dotted line.

Restore original state goes back to the former state of the model fit before changes in the graph have been made.

5 Preliminary steps for constructing the graph

Before the actual calculation of the graph can take place, the user has to carry out some preliminary steps from reading in the data, via fixing the measurement scale and choosing the model variables, to finally postulating the dependence chain. The screen shots shown in this section are based on a fictitious example just for illustrating how to proceed.

After having read in the data file, the variables have to be labelled and a symbol for missing values has to be chosen. Then, the appropriate measurement scale has to be assigned to each variable as shown in Figure 5.

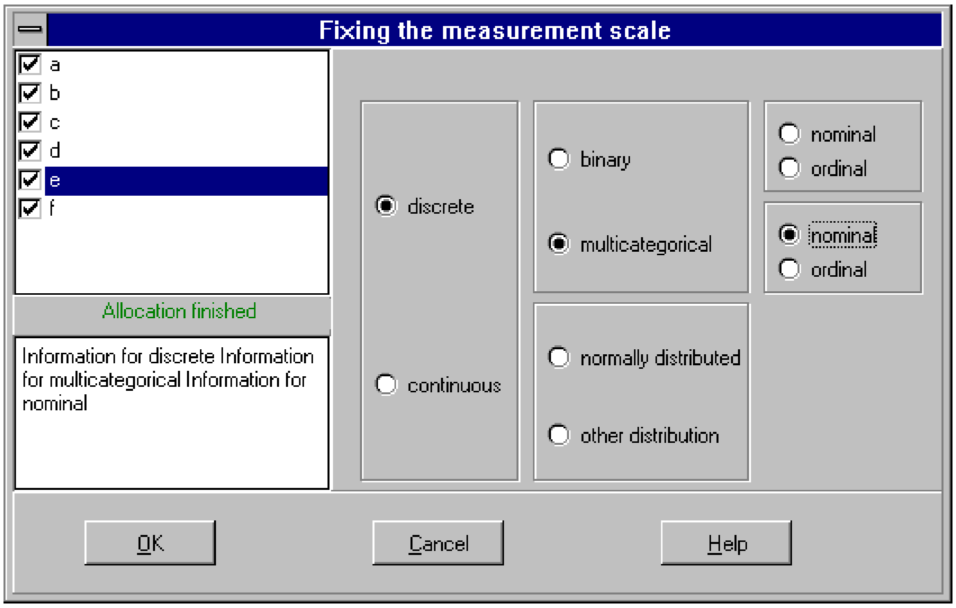

Figure 5: Fixing the measurement scale

For this purpose, a variable has to be selected out of the list by a mouse click. For this variable, the measurement scale has to be chosen by running through different levels which are ordered as a tree. That is, first it has to be distinguished between discrete or continuous, where then for the former it has to be decided between binary and more than two categories and for the latter between normally distributed and distributed according to any other distribution. At the moment, the item `other distribution` is not further specified but our current research project also aims at investigating other continuous distributions being possibly appropriate as marginal distributions in the context of graphical models insofar as for instance an adequate multivariate distribution fulfilling the equivalence of the Markov properties may be constructed based on these marginals (cf. Caputo, 1998). At the third level, the discrete variables have to be finally fixed as nominal or ordinal, respectively. It is sufficient to choose only the radio buttons in the very right position of a tree to set the scale measurement for a variable. The other radio buttons are included in this dialog since more detailed information on the different scales can be obtained on request. A tick in front of the variable indicates that the measurement scale for this variable is completely specified. Only in case all variables are specified with respect to their measurement scale the dialog can be closed by the button `OK`.

As next step, it has to be decided which variables are to be included in the actual analysis. This dialog is served as follows. One or more variables have to be selected and are then added to or removed from the list of model variables by clicking on `>` or `<`, respectively. In case all variables should be added or removed it suffices to click on `>>` or `<<`. In our example, all variables are to be incorporated in the model. Thus, they are all listed as model variables as shown in Figure 6.

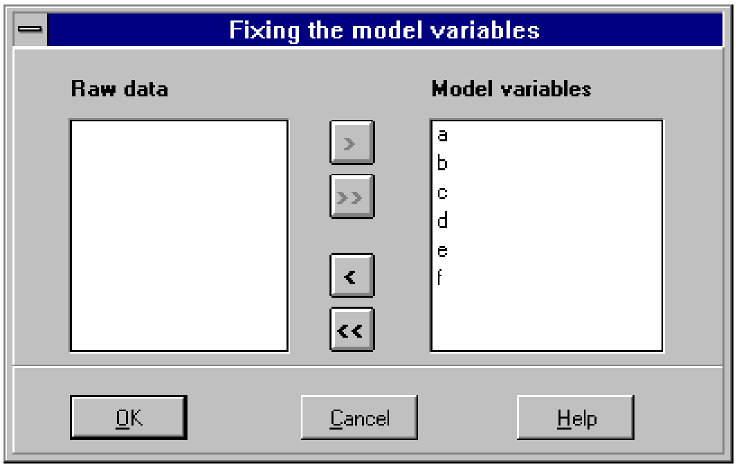

Figure 6: Fixing the model variables

The next step addresses the postulation of the dependence chain which is realized in the program by the item `Construct the dependence chain` of the menu `Analysis`. Using the button `Add` an arbitrary number of chain components can be generated. The variables to be investigated which can be found in a list box on the left side of the window can then be positioned by "drag–and–drop" in the appropriate chain component. The variables can also be exchanged between the different chain components based on the same principle. A chain component can be marked by a mouse click being then depicted with a red border. The marked component can then again be removed from the chain by clicking on the button `Remove`. This implies that all variables contained in this component are given back to the original list and that the remaining components are reordered. A marked chain component can also be labelled by writing this information in the edit box `Name`. There is the possibility to create stacked boxes by splitting the marked component in two or more boxes. After having distributed all variables over the chain components the dialog will be closed by the button `OK`. This state is depicted in Figure 7.

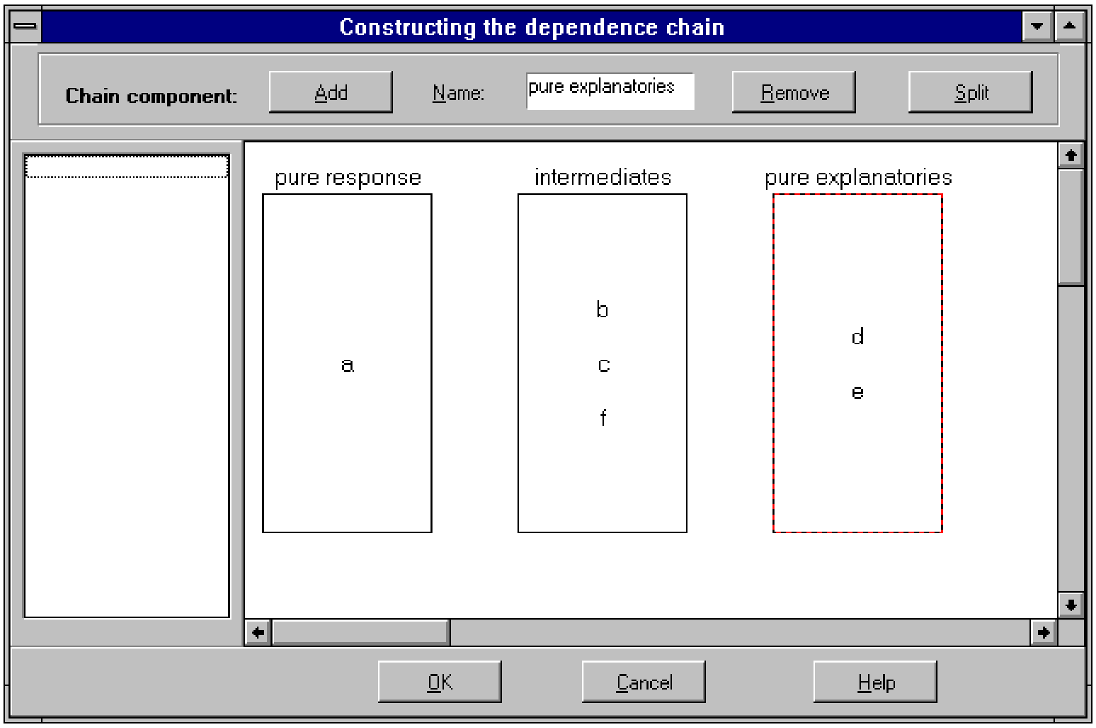

Figure 7: Constructing the dependence chain – finished

As last step the actual chain graph can be calculated by selecting the item `Analysis -- Construct graphical model`. This step may be preceeded by a data–screening to check if interactions or nonlinearities should be included in the model. In the dialog depicted in Figure 8 the user has at first to address some general aspects such as choice of the type of graph and of the type of regression to be calculated where the latter not only depends on the measurement scale, but also on the type of graph. Thus, the dropdown list for the type of regression changes depending on the graph which has been selected in advance, since not all regressions are reasonable for both, concentration and covariance graphs.

The lower part of the dialog addresses the number of steps which should be automatically carried out by the computer before the user gets the chance to interrupt and to modify

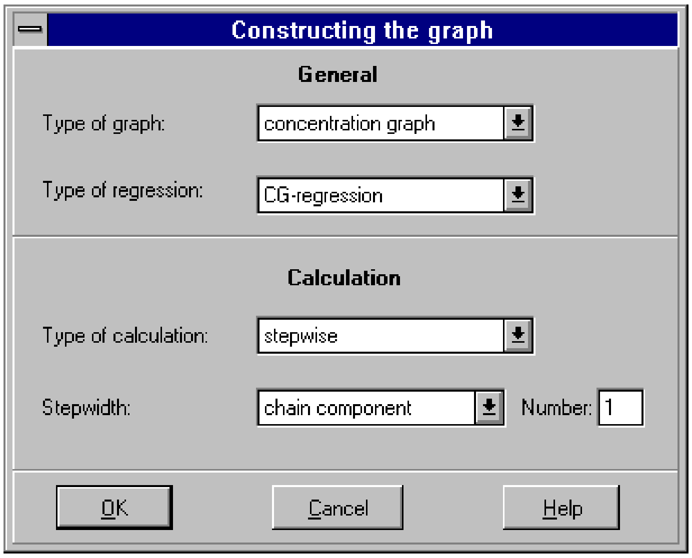

Figure 8: Dialog for calculating the graph

the model fit. In the dropdown list `Type of calculation` it is distinguished whether the complete graph should be fitted automatically or whether the user wants to stop the process of model fit after a certain number of steps to modify the calculated model. Choosing the item `stepwise`, new elements of the dialog are shown on the screen where the number of steps to be conducted automatically has to be fixed. The dropdown list `Stepwidth` enables a rough division and the item `Number` fixes the desired number of steps. The following stepwidths can be chosen. A complete calculation of one chain component is run if `chain component` is selected. Choosing `variable` a certain number of variables are automatically investigated and the corresponding models are calculated. If the user is interested to follow the model fit variable by variable he/she has to select `single step` of `selection` as stepwidth. Here, the program stops after each step of the selection strategy and shows the result. The item `Number` is then of course dropped. The screen shot depicted in Figure 8 shows the calculation of a concentration graph via a CG–regression where the graph should be displayed on the screen after having calculated all regressions belonging to one chain component.

Notice that the calculation of the regressions also requires fixing critical values which can be done in an extra dialog where some defaults are given as support especially for users being not familiar with such model searchs.

6 Discussion

Fitting a graphical chain model and visualizing it on request by the user as well as modifying it interactively can be regarded as the main features of our program. It should, however, be noted that the implemented selection strategy is only one approach to search for a model being consistent with the data, where of course it cannot be expected that there is one true model identified by the applied strategy. Typically, there will be different models consistent with the data and it will be the task of the data–holder to cautiously decide in favour of that particular model which best reflects his/her subject–matter knowledge or which is just the simplest. In this respect, our current work also addresses the development of other model search strategies leading presumably to differing models being all consistent with the data. This should make it easier for the data–holder to weigh up the different possibilities and to come to a decision. This problem of model search is also discussed in detail by Edwards (1995, Chapter 6). He describes two different selection procedures for undirected graphs. First, he introduces a stepwise strategy where similarly to the procedure we use two successive models are compared in turn. The two models are successive in the sense that they differ in only one edge. In the second strategy, which is proposed by Edwards and Havránek (1985, 1987), a set of models is divided into so–called accepted and rejected models by an overall goodness–of–fit test.

We are especially interested in an algorithm being on the one hand data–driven but on the other hand also model–based where we also would like to account for the concept of decomposability for different reasons. As also pointed out by Edwards (1995, p. 143) the restriction on decomposable models which was originally proposed by Wermuth (1976) does not only lead to a gain in interpretation but also in efficiency when calculating the ML–estimates since explicit estimates are possible. In addition, because of the resulting collapsibility unnecessary fitting can be avoided. But, as already mentioned in Section 2, the decomposability of graphical chain models with CG–regressions is still an open problem.

Our current research also concerns alternative families of multivariate distributions as for instance the proposal by Koehler and Symanowski (1995) which has been extensively investigated by Caputo (1998) regarding its capability as multivariate distribution in the framework of graphical models. As one essential property she has for instance shown that these distributions fulfil the equivalence of the Markov properties. Looking for other multivariate distributions than the CG–distribution seems to be necessary when thinking of data situations such as event history analysis where the assumption of normality is typically not justified. To cope with that problem Edwards (1995, p. 151ff) suggests to embed the CG–distributions in a broader family of distributions by using Box–Cox transformations (cf. Box, Cox, 1964).

Now, let us come back to the realization of our computer system. Besides all aspects concerning the design which are already fixed, the framework of the program has also been implemented. The most important classes are modelled, but the implementation of them still has to be done, where the main tasks are to find efficient algorithms for the selection strategy as well as for the visualization of the graph. As discussed above it is important to realize a well thought out interface to the package mathematical functions to easily adopt the program to the integration of other selection strategies. In addition to the development of other perhaps more appropriate selection strategies and their implementation, there are further desirable capabilities of the program which have not been yet realized.

For example, the data gained in empirical studies are usually stored in databases to make them easily available for statistical analyses. Thus, it would be reasonable to create an additional interface to read in data stored in databases. This can be quite easily realized with Delphi since its connection to databases is one of its most important skills.

A possible disadvantage of our program results from its present restriction to run on computers with Microsoft–Windows as operating system. It would be therefore desirable to have at least a connection to computers with other operating systems.

A third aspect relates to that the dialogs of our program have to be currently carried out via menus. It would be reasonable to have the additional possibility of using a commando driven interface to make it faster for the trained user.

Finally, it should be mentioned that it would be desirable to integrate interfaces to other programs for the analysis of graphical models and to commonly used statistical software packages in our computer system.

References

Badsberg, JH (1992) Model search in contingency table by CoCo. In: Dodge Y, Whittaker J (eds) Computational Statistics, CompStat 1992 Neuchâtel. Physica–Verlag, Heidelberg, pp. 251-256

Booch, G (1997) Object–oriented analysis and design. With applications. 2. ed. Addison–Wesley, New York

Box, GEP and Cox, DR (1964) An analysis of transformations (with discussion). J. Roy. Statist. Soc., Ser. B 26: 211-250

Caputo, A (1998) Eine alternative Familie von Modellverteilungen für Kovarianz– und Konzentrationsgraphen. Dissertation, University of Munich

Caputo, A, Heinicke, A, and Pigeot, I (1997) A graphical chain model derived from a selection strategy for the sociologists graduates study. SFB 386 – Discussion Paper 74, University of Munich

Cox, DR and Wermuth, N (1993) Linear dependencies represented by chain graphs (with discussion). Statist. Sci. 8: 204-283

Cox, DR and Wermuth, N (1996) Multivariate dependencies – models, analysis and interpretation. Chapman and Hall, London

Darroch, JN, Lauritzen, SL, and Speed, TP (1980) Markov fields and log–linear interaction models for contingency tables. Ann. Statist. 8: 522-539

Deutsches Institut für Normung (1988) DIN 66234 Teil 8: Bildschirmarbeitsplätze; Grundsätze ergonomischer Dialoggestaltung. Deutsches Institut für Normung

Edwards, D (1987) A guide to MIM. Research Report 87/1. Statistical Research Unit, University of Copenhagen, Denmark

Edwards, D (1995) Introduction to graphical modelling. Springer–Verlag, New York

Edwards, D and Havránek T (1985) A fast procedure for model search in multidimensional contingency tables. Biometrika 72: 339-351

Edwards, D and Havránek T (1987) A fast model selection procedure for large families of models. J. Amer. Statist. Assoc. 82: 205-213

Frydenberg, M (1990a) The chain graph Markov property. Scand. J. Statist. 17: 333-353

Frydenberg, M (1990b) Marginalization and collapsibility in graphical interaction models. Ann. Statist. 18: 790-805

Frydenberg, M and Lauritzen, SL (1989) Decomposition of maximum likelihood in mixed graphical interaction models. Biometrika 76: 539-555

Koehler, KJ and Symanowski, JT (1995) Constructing multivariate distributions with specific marginal distributions. J. Multivar. Anal. 55: 261-282

Kreiner, S (1986) Computerized exploratory screening of large–dimensional contingency tables. Compstat. 7: 43-48

Kreiner, S (1987) Analysis of multidimensional contingency tables by exact conditional tests: techniques and strategies. Scand. J. Statist. 14: 97-112

Kreiner, S (1992) Notes on DIGRAM, Version 2.10. The Danish Institute of Educational Research, Copenhagen, Denmark

Lauer, S (1998) Interactive modelling of categorical data. In: Marx B, Friedl H (eds) Proceedings of the 13th International Workshop on Statistical Modeling, New Orleans, July 27-31, 1998: 443-446

Lauritzen, SL (1996) Graphical models. Clarendon Press, Oxford

Lauritzen, SL and Wermuth, N (1989) Graphical models for associations between variables, some of which are qualitative and some quantitative. Ann. Statist. 17: 31-57

Wermuth, N (1976) Model search among multiplicative models. Biometrics 32: 253-263

Wermuth, N and Lauritzen, SL (1990) On substantive research hypotheses, conditional independence graphs and graphical chain models (with discussion). J. Roy. Statist. Soc., Ser. B 52: 21-72

Wermuth, N, Cox, DR, and Pearl, J (1998) Explanations for multivariate structures derived from univariate recursive structures. Technical Report, Johannes Gutenberg University, Mainz

Whittaker, J (1990) Graphical models in applied multivariate statistics. Wiley, New York