Technische Universität München

Zentrum Mathematik

Eine R-Implementation normaler linearer ADG-Modelle

Bachelorarbeit

von

Lutz F. Gruber

Themensteller: Prof. Dr. Thomas Klein

Betreuer: Prof. Dr. Thomas Klein

Abgabetermin: 15. April 2010

Hiermit erkläre ich, dass ich die Bachelorarbeit selbstständig angefertigt und nur die angegebenen Quellen verwendet habe.

München, den 15. April 2010

Abstract

In a multivariate normal statistical model defined by the Markov properties determined by an acyclic digraph, the maximum likelihood estimates (MLEs) of the model's parameters can be explicitly calculated. Here the corresponding linear regression model, the normal linear ADG model, is introduced. Furthermore, the maximum likelihood estimation in this model is discussed. A user guide to the `R` implementation `nlADG` of the normal linear ADG model is presented and the package's implementation in `R` is examined.

Zusammenfassung

In einem multivariaten normalverteilten Modell, dessen Markov Eigenschaft durch einen kreisfreien gerichteten Graph gegeben ist, können die Maximum Likelihood Schätzer (MLE) der Modellparameter explizit bestimmt werden. In dieser Arbeit wird das zugehörige Regressionsmodell, das normale lineare ADG Modell, eingeführt und die Berechnung der MLEs der Modellparameter hergeleitet. Weiterhin wird ein Anwendungsbeispiel des `R` Pakets `nlADG` präsentiert und die Implementation des Pakets untersucht. Das `R` Paket `nlADG` ist eine Implementation des normalen linearen ADG Modells.

Acknowledgments

I want to thank my advisor, Prof. Dr. Thomas Klein, for bringing the topic to my attention and providing guidance throughout the project. I felt exceptionally well-supported and I am grateful for his enlightening and inspiring discussions.

Contents

Chapter 1

Introduction

Graphical models utilize graphs to describe multivariate statistical dependencies. In the so-called independence graph $D = (V, R)$, each response variable $x_{[v]}$, $v \in V$, is represented by a vertex $v \in V$ and the graph's arrows or edges between the vertices specify whether and how the corresponding response variables depend on one another. In this thesis we will only consider acyclic directed graphs (ADGs) as independence graphs. The parents $pa_D(v)$ of a vertex $v \in V$ are defined as $pa_D(v) := \{p \in V \mid (p, v) \in R\}$. Loosely speaking, a vertex $v \in V$'s descendants $de_D(v)$ are defined to be the vertices $d \in V$ that can be reached on a directed path from v; a vertex $v \in V$'s nondescendants are defined as $nd_D(v) := V \setminus (de_D(v) \cup \{v\})$. We speak of a normal ADG model if the response variables $x_{[v]}$, $v \in V$, of a statistical model follow a nonsingular multivariate normal distribution and satisfy the Markov property $x_{[v]} \perp\!\!\!\perp x_{nd_D(v) \setminus pa_D(v)} \mid x_{pa_D(v)}$. Here we write $a \perp\!\!\!\perp b \mid c$ for some random variables a, b, and c to denote that a and b are independent given c. By x_W we denote the multivariate random variable $x_W := (x_{[w]} \mid w \in W)$ for any subset $W \subseteq V$.

The linear regression model corresponding to the normal ADG model is the normal linear ADG model. The normal linear ADG model is determined by its mean-value space L, which factorizes into MANOVA subspaces according to the vertices of the independence graph D. This product structure allows for the explicit maximum likelihood estimation of a normal linear ADG model's parameters (ξ, Σ). Based on these results, I developed a package to the statistical software `R`, which implements the maximum likelihood parameter estimation in the normal linear ADG model.

A more exhaustive treatment of the topics discussed in this thesis and further reading on normal linear ADG models can be found in the paper by Andersson and Perlman [1998]. Standard references for graphical models are Whittaker [1990] and Lauritzen [1996].

The concept of conditional independence and the basics of ADGs are introduced in chapter 2. Furthermore, chapter 2 gives an introduction of the normal ADG model. In chapter 3, we study the normal linear ADG model and examine how to explicitly calculate the maximum likelihood estimates of its parameters. Chapter 4 explains how to work with the `R` package `nlADG`, and we study how the package's implementation relates to the mathematical treatment done in the preceding chapter.

Chapter 2

Normal ADG Models

2.1 Conditional Independence

The conditional independence of random variables is a very basic concept underlying the analysis of graphical models. The definitions of conditional independence as well as the results presented in this section closely follow Klenke [2008, ch. 12.3] and Hoffmann-Jørgensen [1994, ch. 6].

Definition 2.1 (Conditional independence)**.** *Let $(\Omega, \mathcal{F}, P)$ be a probability space, let $\mathcal{A} \subset \mathcal{F}$ be a sub-σ-algebra and let $(\mathcal{A}_i)_{i\in I}$ be an arbitrary family of sub-σ-algebras of $\mathcal{F}$. Assume that for any finite $J \subset I$, any choice of $A_j \in \mathcal{A}_j$ and for all $j \in J$,*

$$P\left(\bigcap_{j\in J} A_j \mid \mathcal{A}\right) = \prod_{j\in J} P\left(A_j \mid \mathcal{A}\right) \text{ almost surely.}$$

Then the family $(A_i)_{i\in I}$ is called independent given $\mathcal{A}$.

A family $(X_i)_{i\in I}$ of random variables on $(\Omega, \mathcal{F}, P)$ is called independent given $\mathcal{A}$ if the σ-algebras $(\sigma(X_i))_{i\in I}$ are independent given $\mathcal{A}$.

For some pairwise disjoint subsets A, B, C of I we write

$$X_A \perp\!\!\!\perp X_B \mid X_C.$$

to denote the conditional independence of the (multivariate) random variables X_A and X_B given $\mathcal{A} = \sigma(X_C)$. Here we write X_J for a finite subset $J \subseteq I$ to denote the (multivariate) random variable $(X_j)_{j\in J}$ and we say X_A and X_B are independent given X_C.

Definition 2.2 (Conditional density)**.** *Let $(\mathbf{X}, \mathcal{X}, \mu)$ and $(\mathbf{Y}, \mathcal{Y}, \nu)$ be σ-finite measure spaces, and let X and Y be random functions with values in $(\mathbf{X}, \mathcal{X})$ and $(\mathbf{Y}, \mathcal{Y})$ such that (X, Y) is absolutely $\mu \otimes \nu$-continuous with $\mu \otimes \nu$-density $f_{X,Y}(x, y)$. Let $f_Y(y)$ denote the ν-density of Y and set*

$$f_{X|Y}(x \mid y) = \frac{f_{X,Y}(x, y)}{f_Y(y)} \qquad \forall (x, y) \in X \times Y$$

with the convention: $\frac{a}{0} = 0$ for all $x \geq 0$.

It follows from the definition of conditional densities that

$$P(X \in A \mid Y = y) = \int_A f_{X|Y}(x \mid y)\mu(dx) \qquad \forall A \in \mathcal{X}.$$

Proposition 2.3 (Some characterizations of the conditional independence of random variables)**.** *Let X, Y and Z be random variables with values in the measurable spaces $(\mathbf{X}, \mathcal{X})$, $(\mathbf{Y}, \mathcal{Y})$ and $(\mathbf{Z}, \mathcal{Z})$, respectively, and let $\mu = P_{Y,Z}$ and $\nu = P_Z$ be the distributions laws of (Y, Z) and Z, respectively. Then the following four statements are equivalent:*

(i) *X and Y are conditionally independent given Z*

(ii) *(X, Z) and (Y, Z) are conditionally independent given Z*

(iii) *For all $A \in \mathcal{X}$ there exist a measurable function $q_A : \mathbf{Z} \to [0, 1]$ such that $P(X \in A \mid (Y, Z) = (y, z)) = q_A(x)$ μ-a.s.*

(iv) *$P(X \in A \mid (Y, Z) = (y, z)) = P(X \in A \mid Z = z)$ μ-a.s. $\forall A \in \mathcal{X}$.*

Conditional densities and these characterizations of the conditional independence of random variables are very useful tools when analyzing the dependency structure of a statistical model.

We illustrate how the conditional probability density functions of some random variables can be used to determine their (conditional) dependencies. Let $(X_1, X_2, X_3) \sim \mathcal{N}_3(0, \Sigma)$ with

$$\Sigma = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}.$$

Here $\mathcal{N}_3(0, \Sigma)$ denotes the centered 3-variate normal distribution with covariance matrix Σ. We obtain the (conditional) probability density functions

$$\begin{aligned}
f_{X_1,X_2,X_3}(x_{123}) = f_{X_1,X_2,X_3}(x_1, x_2, x_3) &= \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2} x_{123}^T \Sigma^{-1} x_{123}\right) \\
&= \frac{1}{(2\pi)^{\frac{3}{2}}} \exp\left(-\frac{1}{2}\left(3x_1^2 - 2x_1x_2 - 2x_1x_3 + x_2^2 + x_3^2\right)\right) \\
f_{X_1,X_2}(x_{12}) &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}\left(2x_1^2 - 2x_1x_2 + x_2^2\right)\right) \\
f_{X_1,X_3}(x_{13}) &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}\left(2x_1^2 - 2x_1x_3 + x_3^2\right)\right) \\
f_{X_1}(x_1) &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x_1^2\right)
\end{aligned}$$

which we use to verify that $X_2 \perp\!\!\!\perp X_3 \mid X_1$:

$$\begin{aligned} & f_{X_1,X_2}(x_1,x_2)\cdot f_{X_1,X_3}(x_1,x_3) = f_{X_1,X_2,X_3}(x_1,x_2,x_3)\cdot f_{X_1}(x_1) \\ \overset{f_{X_1}>0}{\iff} & \frac{f_{X_1,X_2}(x_1,x_2)}{f_{X_1}(x_1)}\cdot\frac{f_{X_1,X_3}(x_1,x_3)}{f_{X_1}(x_1)} = \frac{f_{X_1,X_2,X_3}(x_1,x_2,x_3)}{f_{X_1}(x_1)} \\ \iff & f_{X_2|X_1}(x_2\mid x_1)\cdot f_{X_3|X_1}(x_3\mid x_1) = f_{X_2,X_3|X_1}(x_2,x_3\mid x_1) \\ \iff & X_2 \mid X_1 \perp\!\!\!\perp X_3 \mid X_1 \\ \iff & X_2 \perp\!\!\!\perp X_3 \mid X_1 \end{aligned}$$

It is easy to show that X_2 and X_3 are not independent.

2.2 Acyclic Directed Graphs (ADGs)

As already mentioned in the introduction, directed or undirected graphs are used in graphical modeling to describe a statistical model's dependency structure. Interestingly, one can apply graph theoretical concepts on these independence graphs to derive conclusions for the statistical analysis.

Definitions of directed graphs (digraphs) are given in Roberts and Tesman [2009, ch. 3]. Our definitions of digraphs, ADGs and notation conventions follow Andersson and Perlman [1998].

Definition 2.4 (Directed graph)**.** *A directed graph (digraph) D is a pair (V,R), where V is a finite set of vertices and $R \subseteq (V\times V)\setminus\Delta$ is a binary relation, the set of directed edges, on V such that $(u,v)\in R$ implies $(v,u)\notin R$. Here, $\Delta=\Delta(V)$ is the diagonal $\{(v,v)\mid v\in V\}$; thus loops and multiple are excluded from D.*

We use the arrows $u\to v$ in figures and $u\prec v$ in text to indicate that the edge (u,v) is in R. We denote the corresponding reflexive relation $\bar{R} := R\cup\Delta$ by $\preceq$. We write $u<v$ if $u\prec v$ or there exist $v_1,...,v_k$ in V, $k\geq 1$, such that $u\prec v_1\prec ...\prec v_k\prec v$ and we write $\leq$ to denote the corresponding reflexive relation.
Digraphs are commonly visualized by diagrams as shown in Figure 2.1.

Definition 2.5 (Acyclic directed graph)**.** *An acyclic directed graph (acyclic digraph, ADG) is a directed graph $D=(V,R)$ with the property that $v\not< v$ for all vertices $v\in V$.*

Here the relation $\leq$ is a partial ordering on V, i.e., it is reflexive, antisymmetric, and transitive. There exists a not-necessarily-unique, never-decreasing listing of any ADG D's vertices V: $V=\{v_1,...,v_r\}$ with $i<j\Rightarrow v_j\not\leq v_i$. For an ADG $D=(V,R)$ and v in V, we define $pa_D(v) := \{u\in V\mid u\prec v\}$, the parents of v; $an_D(v) := \{u\in V\mid u<v\}$, the ancestors of v; $de_D(v) := \{u\in V\mid v<u\}$, the descendants of v; and $nd_D(v) := \{u\in V\mid v\not\leq u\} = V\setminus(de_D(v)\cup\{v\})$, the nondescendants of v.
Some Examples of ADGs are shown in Figure 2.2. For the vertex B in top-left graph G in Figure 2.2, we have $pa_G(B)=\{A,D\}$, $an_G(B)=\{A,C,D\}$, $de_G(B)=\emptyset$, and $nd_G(B)=\{A,C,D\}$; for the vertex A we have $pa_G(A)=\emptyset$, $an_G(A)=\emptyset$, $de_G(A)=\{B,D\}$, and $nd_G(D)=\{C\}$.

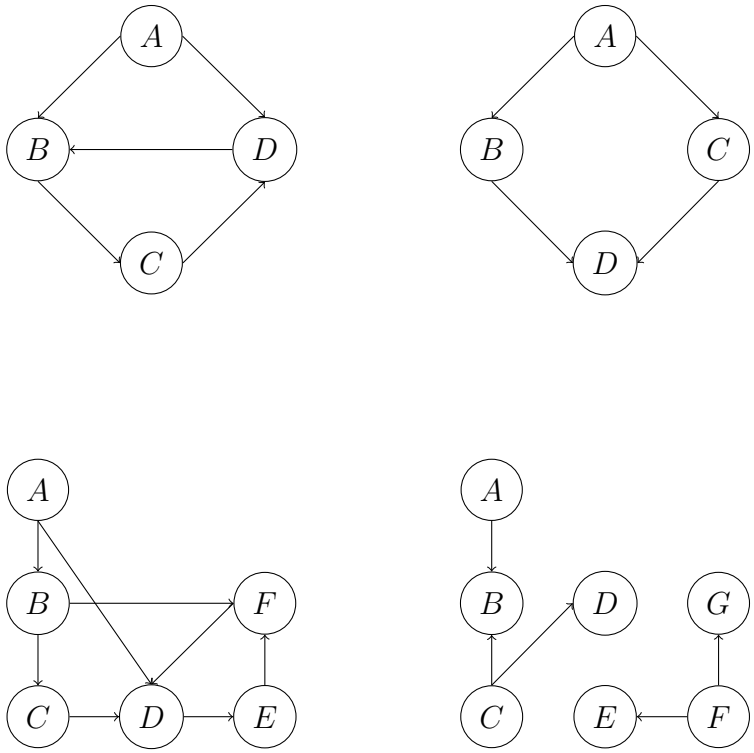

Figure 2.1: Some Digraphs

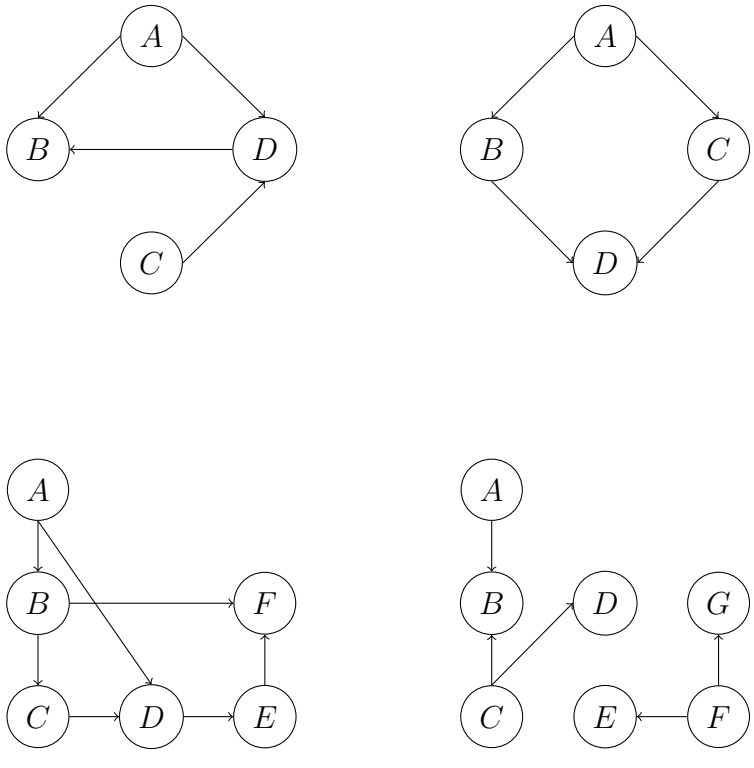

Figure 2.2: Some ADGs

2.3 Normal ADG Models

In ADG models, an ADG is used to describe a model's dependency structure. We introduce ADG models and their restrictions to normal ADG models in this section closely following Andersson and Perlman [1998].

For the remainder of this thesis, let $D = (V, R)$ be an ADG and let $\mathbf{X} := \times(\mathbf{X}_v \mid v \in V)$ be a product probability space, where each $\mathbf{X}_v$ is a measurable space sufficiently regular to ensure the existence of regular conditional probabilities. Let us represent such a multivariate probability distribution P on $\mathbf{X}$ by a random variate $x := (x_{[v]} \mid v \in V) \in \mathbf{X}$. For any subset $A \subseteq V$, define $x_A := (x_v \mid v \in A)$, and $x_\emptyset := \textit{constant}$. We abbreviate $x_{[v]}$ and x_A by v and A, respectively.

Definition 2.6. *A probability measure P on $\mathbf{X}$ is said to be (local) D-Markovian if*

$$v \perp\!\!\!\perp (nd_D(v) \setminus pa_D(v)) \mid pa_D(v) \text{ for all } v \in V.$$

For a never-decreasing listing $\{v_1, ..., v_r\}$ of V, it follows from Lauritzen et al. [1990] that P is D-Markovian if and only if

$$v_m \perp\!\!\!\perp (\{v_1, ..., v_{m-1}\} \setminus pa_D(v_m)) \mid pa_D(v_m),\ m = 2, ..., r.$$

Definition 2.7 (Normal ADG model)**.** *Let I_v, $v \in V$ be nonempty index sets with $I := \dot{\bigcup}(I_v \mid v \in V)$. The normal ADG model $\mathbf{N}_I(D)$ is the family of all D-Markovian nonsingular multivariate normal distributions on $\mathbb{R}^I$.*

A normal distribution $\mathcal{N}(\xi, \Sigma)$ is called nonsingular if its covariance matrix Σ is positive definite, i.e. if Σ is nonsingular.

Chapter 3

Normal Linear ADG Models

The normal linear ADG model is the linear regression model corresponding to normal ADG models. We will perform the parameter estimation in a set of MANOVA systems, into which the mean-value space of the normal linear ADG model factorizes. Our treatment of normal linear ADG models closely follows Andersson and Perlman [1998].

3.1 Definition

We introduce the normal linear ADG model based on the multivariate normal linear regression model (i.e. MANOVA model). We denote the set of labels of the individuals (or statistical units) by N, the set of labels of the variables (of features) observed on each individual by I and the set of labels of the predictors by T. We write $\mathbf{M}(J \times K)$ to denote the vector space of all real $J \times K$ matrices for any finite index sets J and K, $\mathbf{P}(J)$ to denote the cone of all real positive definite $J \times J$ matrices, and we define $\mathbf{M}(J) := \mathbf{M}(J \times J)$.

Definition 3.1 (MANOVA subspace)**.** *A linear subspace $L \subseteq \mathbf{M}(N \times I)$ is called a MANOVA subspace if $L\mathbf{M}(I) \subseteq L$ or, equivalently, if $L\mathbf{M}(I) = L$.*

Every MANOVA subspace L can be represented in the form

$$L = \{ZB \mid B \in \mathbf{M}(T \times I)\},$$

where $Z \in \mathbf{M}(N \times T)$ is a design matrix and $B \in \mathbf{M}(T \times I)$ is a parameter matrix. Conversely, any linear subspace of that form is a MANOVA subspace. Furthermore, each MANOVA subspace $L \subseteq \mathbf{M}(N \times I)$ uniquely determines a normal MANOVA model $\mathbf{N}_{N \times I}(L)$, which is defined as

$$\mathbf{N}_{N \times I}(L) := (\mathcal{N}_{N \times I}(\xi, \Sigma \otimes 1_N) \mid (\xi, \Sigma) \in L \times \mathbf{P}(I)),$$

where $\mathcal{N}_{N \times I}(\xi, \Sigma \otimes 1_N)$ denotes the multivariate normal distribution with mean ξ and covariance $\Sigma \otimes 1_N$. This model consists of N independent normal random vectors $x_j \sim \mathcal{N}_{1 \times I}(\xi_j, \Sigma)$ for all individuals $j \in N$. Here we have $\Sigma \in \mathbf{P}(I)$, $\xi_j \in \mathbb{R}^{1 \times I}$, and $\xi := (\xi_j \mid j \in N) \in L$.

Definition 3.2 (D-linear subspace)**.** *A linear subspace $L \subseteq \mathbf{M}(N \times I)$ is called a D-linear subspace, or a D-subspace, if it satisfies the following three conditions:*

(i) $L = \times(L_{[v]} \mid v \in V)$*;*

(ii) $\forall v \in V, L_{[v]}$ *is a MANOVA subspace of* $\mathbf{M}(N \times [v])$*;*

(iii) $\forall v \in V, L_{\prec v \succ}\mathbf{M}(\prec v \succ \times [v]) \subseteq L_{[v]}$.

Here $L_J := \{\xi_J \mid \xi \in L\} \subseteq \mathbf{M}(N \times J)$, where ξ_J denotes the $N \times J$ submatrix of $\xi \in \mathbf{M}(N \times I)$ for any $J \subseteq I$.

Definition 3.3 (Normal linear ADG model)**.** *For any D-subspace L and covariance matrix $\Sigma \in \mathbf{P}(D; I)$, the normal linear ADG model $\mathbf{N}_{N\times I}(L, D)$ is defined as*

$$\mathbf{N}_{N\times I}(L, D) := (\mathcal{N}_{N\times I}(\xi, \Sigma \otimes 1_N) \mid (\xi, \Sigma) \in L \times \mathbf{P}(D; I)).$$

Normal linear ADG models mainly differ from MANOVA models in two ways: firstly, the set of covariance matrices Σ is restricted from $\mathbf{P}(I)$ to $\mathbf{P}(D; I)$ and, secondly, the mean-value space L is expanded from the class of MANOVA subspaces to the class of D-linear subspaces.

3.2 D-Parameters

For $\Sigma \in \mathbf{P}(I)$, let Σ_{JK} denote the $J \times K$ submatrix of Σ, let $\Sigma_J := \Sigma_{JJ} \in \mathbf{P}(J)$, and let Σ_J^{-1} denote $(\Sigma_J)^{-1}$. If $J = \emptyset$ or $K = \emptyset$, then $\mathbf{M}(J \times K) := \{0\}$, the zero vector space. If $J = \emptyset$ and $K = \emptyset$, let the product of a $J \times \emptyset$ matrix and a $\emptyset \times K$ matrix be the $J \times K$ zero matrix. We define

$$\Sigma_{J\bullet K} := \Sigma_J - \Sigma_{JK}\Sigma_J^{-1}\Sigma_{KJ} \in \mathbf{P}(J)$$

for disjoint subsets $J, K \subseteq I$. For every $v \in V$, we define the abbreviations

$$\not\prec v \not\succ := I_{nd_D(v)\setminus pa_D(v)},$$

$$\prec v \succ := I_{pa_D(v)},$$

$$[v] := I_v,$$

and we define $\preceq v \succeq := \prec v \succ \dot\cup [v]$. We can now partition the submatrices $\Sigma_{\not\prec v \not\succ \dot\cup \prec v \succ \dot\cup [v]}$, $v \in V$, of the covariance matrix $\Sigma \in \mathbf{P}(I)$ according to the normal linear ADG model's Markov structure:

$$\Sigma_{\not\prec v \not\succ \dot\cup \prec v \succ \dot\cup [v]} = \begin{pmatrix} \Sigma_{\not\prec v \not\succ} & \Sigma_{\not\prec v \succ} & \Sigma_{\not\prec v]} \\ \Sigma_{\prec v \not\succ} & \Sigma_{\prec v \succ} & \Sigma_{\prec v]} \\ \Sigma_{[v \not\succ} & \Sigma_{[v \succ} & \Sigma_{[v]} \end{pmatrix}, \text{ where}$$

$$\Sigma_{\not\prec v \not\succ} \in \mathbf{P}(\not\prec v \not\succ),$$

$\Sigma_{\prec v\succ} \in \mathbf{P}(\prec v\succ)$,

$\Sigma_{[v]} \in \mathbf{P}([v])$,

$\Sigma_{\prec v\not\succ} := \Sigma_{\prec v\succ \not\prec v\not\succ} \in \mathbf{M}(\prec v\succ \times \not\prec v\not\succ)$,

$\Sigma_{[v\not\succ} := \Sigma_{[v]\not\prec v\not\succ} \in \mathbf{M}([v] \times \not\prec v\not\succ)$,

$\Sigma_{[v\succ} := \Sigma_{[v]\prec v\succ} \in \mathbf{M}([v] \times \prec v\succ)$,

$\Sigma_{\not\prec v\succ} = (\Sigma_{\prec v\not\succ})^T$,

$\Sigma_{\not\prec v]} = (\Sigma_{[v\not\succ})^T$,

$\Sigma_{\prec v]} = (\Sigma_{[v\succ})^T$.

Definition 3.4 (D-parameters of (ξ, Σ))**.** *For $(\xi, \Sigma) \in \mathbf{M}(N \times I) \times \mathbf{P}(I)$, the family of matrices*

$$\pi_D(\xi, \Sigma) := ((\xi_{[v]} - \xi_{\prec v\succ}\Sigma^{-1}_{\prec v\succ}\Sigma_{\prec v]}, \Sigma^{-1}_{\prec v\succ}\Sigma_{\prec v]}, \Sigma_{[v]\bullet}) \mid v \in V)$$
$$\in (\times\mathbf{M}(N \times [v]) \times \mathbf{M}(\prec v\succ \times [v]) \times \mathbf{P}([v]) \mid v \in V)$$

is called the family of D-parameters of (ξ, Σ).

Proposition 3.5. *If L is a D-subspace, the mapping*

$$\pi_D : L \times P(D; I) \to \times(L_{[v]} \times \mathbf{M}(\prec v\succ \times [v]) \times \mathbf{P}([v]) \mid v \in V) =: \Pi(L, D; I)$$

is bijective. Thus, every $(\xi, \Sigma) \in L \times \mathbf{P}(D; I)$ is uniquely determined by its D-parameters.

In addition to the D-parameters' characterizing the pair (ξ, Σ), the inverse mapping π_D^{-1} is presented in Andersson and Perlman [1998] in the form of a reconstruction algorithm.

Algorithm 3.6 (Reconstruction algorithm)**.** Let $v_1, ..., v_r$ be a never-decreasing listing of the elements in V. Abbreviate v_m by m, $[v_m]$ by $[m]$, $\prec v_m\succ$ by $\prec m\succ$, and $[v_m\succ$ by $[m\succ$. Partition Σ according to the decomposition $I = [1]\dot\cup...\dot\cup[r]$ and list the D-parameters $\pi_D(\xi, \Sigma)$ in the corresponding order:

$$\pi_D(\xi, \Sigma) =: ((\mu_m, \beta_m, \Lambda_m) \mid m = 1, ...r)$$
$$\in \times(\mathbf{M}(N \times [m]) \times \mathbf{M}(\prec m\succ \times [m]) \times \mathbf{P}([m]) \mid m = 1, ...r).$$

For $m = 1, ..., r$ do:

$\Sigma_{[m\succ} = \beta_m^T\Sigma_{\prec m\succ}$

$\Sigma_{[m]} = \Lambda_m + \beta_m^T\Sigma_{\prec m]}$

$\Sigma_{[m>} = \Sigma_{[m\succ}\Sigma^{-1}_{\prec m\succ}\Sigma_{\prec m>} = \beta_m^T\Sigma_{\prec m>}$

$\xi_{[m]} = \mu_m + \xi_{\prec m\succ}\beta_m$

Here $\Sigma_{[m>}$ denotes $[m] \times (([1]\dot\cup...\dot\cup[m-1]) \setminus \prec m\succ)$ submatrix of $\Sigma_{[1]\dot\cup...\dot\cup[m-1]}$ and $\Sigma_{\prec m>}$ denotes the $\prec m\succ \times (([1]\dot\cup...\dot\cup[m-1]) \setminus \prec m\succ)$ submatrix of $\Sigma_{[1]\dot\cup...\dot\cup[m-1]}$. The reconstruction of the pair (ξ, Σ) is complete after r steps.

3.3 Maximum Likelihood Estimation in a Normal Linear ADG Model

Given the reconstruction algorithm and the characterization of the pair (ξ, Σ) by its D-parameters $((\mu_v, \beta_v, \Lambda_v) \mid v \in V)$, the maximum likelihood estimates $(\hat{\xi}, \hat{\Sigma})$ of (ξ, Σ) can be obtained by calculating their respective D-parameters $((\hat{\mu}_v, \hat{\beta}_v, \hat{\Lambda}_v) \mid v \in V) := \pi_D(\hat{\xi}, \hat{\Sigma})$. In the proof of Proposition 4.1 in Andersson and Perlman [1998] show that

$$x_{[v]} \mid x_{\prec v \succ} \sim \mathcal{N}_{1\times[v]}(x_{\prec v\succ}\beta_v, \Lambda_v) \tag{3.1}$$

when $x \sim \mathcal{N}_{1\times I}(0, \Sigma)$ and $\Sigma \in \mathbf{P}(D; I)$. The assumption of a normal linear ADG model $\mathbf{N}_{N\times I}(L, D) = (\mathcal{N}_{N\times I}(\xi, \Sigma \otimes 1_N) \mid (\xi, \Sigma) \in L \times \mathbf{P}(D; I))$ satisfies $\Sigma \in \mathbf{P}(D; I)$. As explained before, this model consists of N independent normal random vectors $x_j \sim \mathcal{N}_{1\times I}(\xi_j, \Sigma)$ for every individual $j \in N$. It follows that $(x_j - \xi_j) \sim \mathcal{N}_{1\times I}(0, \Sigma)$. Using (3.1), we obtain for all $v \in V$:

$$\begin{aligned}
(x_j - \xi_j)_{[v]} \mid (x_j - \xi_j)_{\prec v\succ} &\sim \mathcal{N}_{1\times[v]}((x_j - \xi_j)_{\prec v\succ}\beta_v, \Lambda_v) && \forall j \in N \\
\iff x_{j,[v]} \mid x_{j,\prec v\succ} &\sim \mathcal{N}_{1\times[v]}(\xi_{j,[v]} + (x_j - \xi_j)_{\prec v\succ}\beta_v, \Lambda_v) && \forall j \in N \\
\iff x_{[v]} \mid x_{\prec v\succ} &\sim \mathcal{N}_{N\times[v]}(\xi_{[v]} + (x - \xi)_{\prec v\succ}\beta_v, \Lambda_v \otimes 1_N)
\end{aligned}$$

Reformulating

$$\begin{aligned}
\xi_{[v]} + (x - \xi)_{\prec v\succ}\beta_v &= \xi_{[v]} + x_{\prec v\succ}\beta_v - \xi_{\prec v\succ}\beta_v \\
&= \xi_{[v]} + x_{\prec v\succ}\Sigma^{-1}_{\prec v\succ}\Sigma_{\prec v]} - \xi_{\prec v\succ}\Sigma^{-1}_{\prec v\succ}\Sigma_{\prec v]} \\
&= \xi_{[v]} - \xi_{\prec v\succ}\Sigma^{-1}_{\prec v\succ}\Sigma_{\prec v]} + x_{\prec v\succ}\Sigma^{-1}_{\prec v\succ}\Sigma_{\prec v]} \\
&= \mu_v + x_{\prec v\succ}\beta_v,
\end{aligned}$$

we obtain

$$x_{[v]} \mid x_{\prec v\succ} \sim \mathcal{N}_{N\times[v]}(\mu_v + x_{\prec v\succ}\beta_v, \Lambda_v \otimes 1_N) \qquad \forall v \in V$$

as the MANOVA models $\mathbf{N}_{N\times[v]}(L_{[v]})$, $v \in V$. The corresponding MANOVA subspaces $L_{[v]}$ can be constructed from the design matrices $Z_v := \begin{pmatrix} \tilde{Z}_v & x_{\prec v\succ} \end{pmatrix} \in \mathbf{M}(N \times (\tilde{T}_v + \prec v\succ))$ and we obtain

$$\begin{aligned}
L_{[v]} &= \{Z_v B_v \mid B_v \in \mathbf{M}((\tilde{T}_v + \prec v\succ) \times [v])\} \\
&:= \left\{ \begin{pmatrix} \tilde{Z}_v & x_{\prec v\succ} \end{pmatrix} \begin{pmatrix} \tilde{B}_v \\ \beta_v \end{pmatrix} \middle| \tilde{B}_v \in \mathbf{M}(\tilde{T}_v \times [v]), \beta_v \in \mathbf{M}(\prec v\succ \times [v]) \right\}.
\end{aligned}$$

The predictors of the response variables $x_{[v]}$, $v \in V$, generate the design submatrices $\tilde{Z}_v \in \mathbf{M}(N \times \tilde{T}_v)$, $v \in V$. It can be shown that—if they exist—the maximum likelihood estimates for μ_v, β_v, and Λ_v in the MANOVA models $\mathbf{N}_{N\times[v]}(L_{[v]})$, $v \in V$, equal the D-parameters $(\hat{\mu}_v, \hat{\beta}_v, \hat{\Lambda}_v \mid v \in V) = \pi_D(\hat{\xi}, \hat{\Sigma})$ (cf. [Andersson and Perlman, 1998, ch.

7]). Given the observation $y \in \mathbf{M}(N \times I)$ of x, the well-known regression formulas in the MANOVA model yield:

$$\begin{aligned} \hat{B}_v &= (Z_v^T Z_v)^- Z_v^T y_{[v]} \\ n\hat{\Lambda}_v &= (y_{[v]} - (\hat{\mu}_v + y_{\prec v \succ}\hat{\beta}_v))^T (y_{[v]} - (\hat{\mu}_v + y_{\prec v \succ}\hat{\beta}_v)) \end{aligned}$$

We can simplify the results stated above to fit into our D-parameter scheme:

$$\begin{aligned} \hat{\beta}_v &= (y_{\prec v \succ}^T \tilde{Q}_v y_{\prec v \succ})^{-1} y_{\prec v \succ}^T \tilde{Q}_v y_{[v]} \\ \hat{\mu}_v &= \tilde{P}_v (y_{[v]} - y_{\prec v \succ}\hat{\beta}_v) \\ n\hat{\Lambda}_v &= y_{[v]}^T \tilde{Q}_v (y_{[v]} - y_{\prec v \succ}\hat{\beta}_v) \end{aligned}$$

Here $\tilde{P}_v := \tilde{Z}_v(\tilde{Z}_v^T \tilde{Z}_v)^- \tilde{Z}_v^T \in \mathbf{M}(N)$, $\tilde{Q}_v := 1_N - \tilde{P}_v \in \mathbf{M}(N)$ and A^- denotes a generalized inverse of a matrix $A \in \mathbf{M}(N)$. The maximum likelihood estimation of the D-parameters $(\hat{\mu}_v, \hat{\beta}_v, \hat{\Lambda}_v \mid v \in V)$ can be performed if and only if the number of observations $n := |N|$ satisfies

$$n \geq \max\{\dim(\mathrm{row}(\tilde{P}_v)) + |\preceq v \succeq| \mid v \in V\}.$$

Chapter 4

The R Package "nlADG"

The `R` package `nlADG` provides a convenient interface to the maximum likelihood estimation in a normal linear ADG model.

4.1 User Guide

The `R` package requires two input parameters, `formulas` and `D`, which together specify the normal linear ADG model $\mathbf{N}_{N\times I}(L)$. The input parameter `formulas` requires a `list` of regression formulas, each specifying the MANOVA subspace $\tilde{L}_{[v]} := \{\tilde{Z}_v\tilde{B}_v \mid \tilde{B}_v \in \mathbf{M}(\tilde{T}_v\times I_v)\} \subseteq \mathbf{M}(N\times I_v)$ for a response variable $x_{[v]}$, $v \in V$. Here $\tilde{Z}_v \in \mathbf{M}(N\times\tilde{T}_v)$ denotes a design matrix corresponding to the MANOVA subspace $\tilde{L}_{[v]}$. We denote the observed values of the response variable $x := (x_{[v]} \mid v \in V)$ by $y \in \mathbf{M}(N \times I)$. For any entry in `formulas`, the name of the entry has to match the name of the corresponding response variable. The input parameter `D` requires the adjacency matrix of the ADG $D = (V, R)$, which specifies the dependency structure of the normal linear ADG model. The normal linear ADG model in which the `R` package `nlADG` performs the maximum likelihood estimation is

$$\mathbf{N}_{N\times I}(L) := \mathbf{N}_{N\times I}(\times(L_{[v]} \mid v \in V)),$$

with

$$L_{[v]} := \left\{ \begin{pmatrix} \tilde{Z}_v & y_{\prec v\succ} \end{pmatrix} \begin{pmatrix} \tilde{B}_v \\ \beta_v \end{pmatrix} \middle| \tilde{B}_v \in \mathbf{M}(\tilde{T}_v \times [v]), \beta_v \in \mathbf{M}(\prec v\succ \times [v]) \right\}$$
$$\subseteq \mathbf{M}(N \times I_v) \qquad \forall v \in V.$$

The syntax to create an `nlADG` model is `model <- nlADG(formulas, D, ...)`. The output of the function can be accessed under `model$coefficients`, `model$sigma`, `model$xi`, `model$fitted.values`, `model$residuals`, `model$MANOVAmodels`, `model$ADG`, and `model$I`. The MLEs of the coefficient matrices B_v, $v \in V$, are stored in `model$coefficients`, the MLE $\hat{\Sigma}$ of the covariance matrix Σ is stored in `model$sigma`, the MLE $\hat{\xi}$ of the mean-value matrix ξ is stored in `model$xi` as well as in `model$fitted.values`,

the residuals $r := y - \hat{\xi}$ are stored in `model$residuals`, the `lm` objects corresponding to the MANOVA subspaces $L_{[v]}$, $v \in V$, are stored in `model$MANOVAmodels`, the normal linear ADG model's ADG with its column names listed in a never-decreasing order is stored in `model$ADG`, and the partition $I = \dot{\bigcup}(I_v \mid v \in V)$ is stored in `model$I`.

The `nlADG` model `model` can be used for prediction as well. Given some predictive data `newdata` in the `data.frame` format, the syntax for prediction is `prediction <- predict(model, newdata, ...)`.

Consider the following example that illustrates how to work with `nlADG`. Suppose we know the values of the response variables height, weight, and girth radius and the predictive data shoe size, gender, and age of ten individuals. We use a normal linear ADG model to explain the three response variables weight $x_{[w]}$, shoe size $x_{[s]}$, and girth radius $x_{[r]}$. We assume that an individual's shoe size and girth radius are independent given his/her weight. The ADG corresponding to our assumptions of the model's dependency structure is shown in Figure 4.1. We use an individual's squared

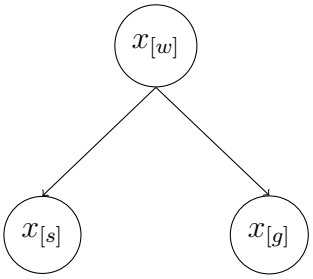

Figure 4.1: The ADG to the regression model

height as a predictor variable for his/her weight. In addition to his/her squared height, we use his/her gender as a categorical predictor variable for his/her shoe size. Lastly, we use an individual's squared height and the square root of his/her age as predictors for his/her girth radius. The dataset used for the regression is presented in Table 4.1. The `R` commands to set up the above-described model are:

```
> library(nlADG)
Loading required package: ggm

> weight <- c(58, 80, 65, 68, 73, 63, 84, 76, 70, 72)
> shoesize <- c(31, 45, 35, 36, 41, 34, 47, 42, 37, 40)
> girthradius <- c(0.87, 1.00, 0.88, 0.83, 0.75, 0.76, 0.90, 0.95, 0.76, 0.89)

> height <- c(1.59, 1.87, 1.68, 1.73, 1.78, 1.66, 1.92, 1.81, 1.75, 1.77)
> height2 <- height^2
> gender <- as.factor(c("f", "m", "f", "f", "m", "f", "m", "m", "f", "m"))
> age <- c(19, 27, 23, 32, 24, 27, 20, 21, 22, 33)
> sqrtage <- sqrt(age)

> formulas <- list( weight = weight ~ height2 - 1,
```

	Weight	Shoe Size	Girth Radius	Height	Gender	Age
1	58	31	0.87	1.59	f	19
2	80	45	1.00	1.87	m	27
3	65	35	0.88	1.68	f	23
4	68	36	0.83	1.73	f	32
5	73	41	0.75	1.78	m	24
6	63	34	0.76	1.66	f	27
7	84	47	0.90	1.92	m	20
8	76	42	0.95	1.81	m	21
9	70	37	0.76	1.75	f	22
10	72	40	0.89	1.77	m	33

Table 4.1: The dataset for our case study

```
      shoesize = shoesize ~ height2 + gender - 1,
      girthradius = girthradius ~ height2 + sqrtage - 1 )
> D <- DAG(shoesize ~ weight, girthradius ~ weight)

> model <- nlADG(formulas, D)
> model
Call:
nlADG.default(formulas = formulas, D = D)

MLE xi:
     weight girthradius shoesize
1  57.96153   0.7183511 30.98455
2  80.17313   0.9441046 44.70071
3  64.70892   0.7977848 34.51439
4  68.61796   0.8799416 36.55938
5  72.64163   0.8667670 40.76067
6  63.17740   0.8094488 33.71319
7  84.51778   0.9354859 46.97358
8  75.11086   0.8675210 42.05243
9  70.21367   0.8351318 37.39416
10 71.82773   0.9105864 40.33488

MLE sigma:
                weight girthradius    shoesize
weight      0.17919430 0.012209309 0.043692412
```

```
girthradius 0.01220931 0.006558551 0.002976960
shoesize    0.04369241 0.002976960 0.104902136
```

We can read from the output that

$$\hat{\xi} = \begin{pmatrix} \hat{\xi}_{[w]} & \hat{\xi}_{[r]} & \hat{\xi}_{[s]} \\ \hline 57.96153 & 0.7183511 & 30.98455 \\ 80.17313 & 0.9441046 & 44.70071 \\ 64.70892 & 0.7977848 & 34.51439 \\ 68.61796 & 0.8799416 & 36.55938 \\ 72.64163 & 0.8667670 & 40.76067 \\ 63.17740 & 0.8094488 & 33.71319 \\ 84.51778 & 0.9354859 & 46.97358 \\ 75.11086 & 0.8675210 & 42.05243 \\ 70.21367 & 0.8351318 & 37.39416 \\ 71.82773 & 0.9105864 & 40.33488 \end{pmatrix}, \text{ and}$$

$$\hat{\Sigma} = \left(\begin{array}{c|ccc} & x_{[w]} & x_{[r]} & x_{[s]} \\ \hline x_{[w]} & 0.17919430 & 0.012209309 & 0.043692412 \\ x_{[r]} & 0.01220931 & 0.006558551 & 0.002976960 \\ x_{[s]} & 0.04369241 & 0.002976960 & 0.104902136 \end{array}\right).$$

To predict the weight $\hat{\xi}_{[w]}$, shoe size $\hat{\xi}_{[s]}$, and girth radius $\hat{\xi}_{[r]}$ for a 20-year-old male individual who is 1.79 m tall, we can use the `predict` function.

```
> newdata <- data.frame(sqrtage = sqrt(20), gender = "m", height2 = 1.79^2)
> predict(model, newdata)
    weight girthradius shoesize
1 73.46012    0.847877 41.18886
```

We can read from the output that this individual's weight, girth radius and shoe size are expected to be approx. 73 [kg], 0.85 [m] and 41, respectively.

4.2 Implementation

The big picture of `nlADG`'s structure is best visualized by Figure 4.2.

The `nlADG.default(formulas, D, ...)` function, which is invoked by the `nlADG (formulas, D, ...)` command, verifies the input and coordinates the parameter estimation. It calls the function `assertDsubspace` to ensure that the data satisfy the

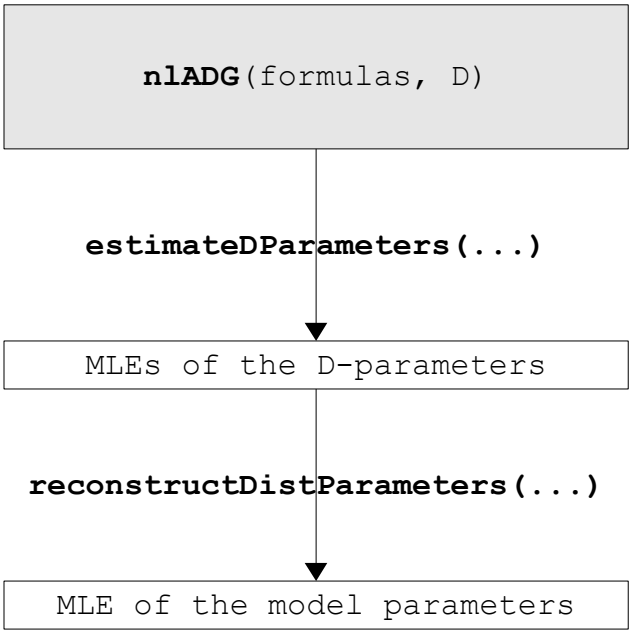

Figure 4.2: Program structure of nlADG

D-subspace characteristics and the function `assertNoObservations` to ensure that enough observations are given to produce a unique maximum likelihood estimate $(\hat{\xi}, \hat{\Sigma})$ of (ξ, Σ). As described above, the parameter estimation is conducted in two steps. Firstly, `estimateDParameters` produces the maximum likelihood estimates of the D-parameters $(\hat{\mu}_v, \hat{\beta}_v, \hat{\Lambda}_v \mid v \in V)$. Then, `reconstructDistParameters` performs the reconstruction algorithm on the MLEs of the D-parameters to construct the pair $(\hat{\xi}, \hat{\Sigma})$.

Both the `assertDsubspace` function and the `assertNoObservations` function examine certain design matrices to verify the required properties in the data.

```
assertDsubspace = function(formulas, D, ...) {
  for (v_ in colnames(D)) {
    for (p_ in pa(v_, D)) {
      designmatrix_v = as.matrix(model.matrix(formulas[[v_]]))
      expanded_designmatrix_v = cbind(designmatrix_v,
          as.matrix(model.matrix(formulas[[p_]])))

      if (qr(designmatrix_v, ...)$rank <
          qr(expanded_designmatrix_v, ...)$rank) {
        stop("The regression system does not fulfill the D-linear
            subspace characteristics.")
      }
    }
  }
```

```
}
```

Given that the program defines the linear subspace L as the product space of the MANOVA subspaces $L := \times(L_{[v]} \mid v \in V)$, it is sufficient to verify

$$\forall v \in V, L_{\prec v \succ}\mathbf{M}(\prec v \succ \times [v]) \subseteq L_{[v]},$$

which is equivalent to (cf. Andersson and Perlman [1998])

$$\forall u, v \in V \text{ with } u \prec v, \text{row}(L_{[u]}) \subseteq \text{row}(L_{[v]}),$$

to confirm the D-subspace characteristics of L. For every $v \in V$ and $p \in pa_D(v)$, the `assertDsubspace` function combines the design matrices $\tilde{Z}_v \in \mathbf{M}(N \times [v])$ and $\tilde{Z}_p \in \mathbf{M}(N \times [p])$ to $\tilde{Z}_{v,p} := \begin{pmatrix} \tilde{Z}_v & \tilde{Z}_p \end{pmatrix} \in \mathbf{M}(N \times ([v] + [p]))$. If for any combination $v \in V$ and $p \in pa_D(v)$ the rank of $\tilde{Z}_{v,p}$ is greater than the rank of $\tilde{Z}_v$, the property stated above does not hold and L cannot be a D-subspace.

```
assertNoObservations = function(formulas, D, ...) {
  n_ = dim(model.frame(formulas[[1]]))[1]

  for (v_ in colnames(D)) {
    rankZ = qr(as.matrix(model.matrix(formulas[[v_]])), ...)$rank
    dimvandparents = dim(as.matrix(lm(formulas[[v_]])$coefficients))[2]
    for (p_ in pa(v_, D)) {
      dimvandparents = dimvandparents +
          dim(as.matrix(lm(formulas[[p_]])$coefficients))[2]
    }

    if (n_ < rankZ + dimvandparents) {
      stop("Not enough observations are given for the maximum likelihood
          estimation to exist and be unique.")
    }
  }
}
```

The function `assertNoObservations` uses the relationship

$$\text{rank}(\tilde{Z}_v) = \dim(\text{row}(\tilde{P}_v))$$

to determine whether the inequality

$$n \geq \max\{\dim(\text{row}(\tilde{P}_v)) + |\preceq v \succeq| \mid v \in V\}$$

holds.

All the regression conducted by `nlADG` takes place in the `estimateDParameters` function.

```
estimateDParameters = function(formulas, D, ...) {
  n_ = dim(model.frame(formulas[[1]]))[1]

  ...

  for(v_ in colnames(D)) {
    model_v = lm(formulas[[v_]])

    if (length(pa(v_, D)) >= 1) {
      no_predictors = dim(model.matrix(formulas[[v_]]))[2]

      model_frame_expanded_v = expand.model.frame(formulas[[v_]], pa(v_, D))
      model_v = lm(model_frame_expanded_v)

      no_predictors_expanded = dim(as.matrix(model_v$coef))[1]

      beta_hat[[v_]] = matrix(
          as.matrix(model_v$coef)[(no_predictors+1):no_predictors_expanded,],
          nrow=no_predictors_expanded-no_predictors )

      clearedDataframe = model_frame_expanded_v
      for(p_ in pa(v_, D)) {
        if (is.matrix(clearedDataframe[[p_]])) {
          clearedDataframe[[p_]] = matrix(0,
          ncol=dim(as.matrix(model_frame_expanded_v[[p_]]))[2],
          nrow=dim(as.matrix(model_frame_expanded_v[[p_]]))[1])
        } else {
          clearedDataframe[[p_]] =
              rep(0, dim(as.matrix(model_frame_expanded_v[[p_]]))[1])
        }
      }
      mu_hat[[v_]] = as.matrix(predict(model_v, clearedDataframe))

      fittingDataframe = model_frame_expanded_v
      for (p_ in pa(v_, D)) {
        if (is.matrix(fittingDataframe[[p_]])) {
          fittingDataframe[[p_]] = fittedvalues[[p_]]
        } else {
          fittingDataframe[[p_]] = as.vector(fittedvalues[[p_]])
        }
      }
      fittedvalues[[v_]] = as.matrix(predict(model_v, fittingDataframe))
    } else {
      mu_hat[[v_]] = as.matrix(model_v$fitted.values)
```

```
      beta_hat[[v_]] = matrix(0, ncol=dim(mu_hat[[v_]])[2], nrow=0)

      fittedvalues[[v_]] = as.matrix(model_v$fitted.values)
    }

    original_values_v = as.matrix(model_v$residuals) +
        as.matrix(model_v$fitted.values)

    residuals[[v_]] = original_values_v - fittedvalues[[v_]]

    lambda_hat[[v_]] = t(as.matrix(model_v$residuals)) %*%
        original_values_v / dim(mu_hat[[v_]])[1]

    coefficients[[v_]] = as.matrix(model_v$coefficients)

    if (dim(mu_hat[[v_]])[2] >= 2) {
      rv_names = paste(v_, 1:dim(mu_hat[[v_]])[2], sep=".")
    } else {
      rv_names = v_
    }

    ...

    models[[v_]] = model_v
  }

  ...

  list(beta_hat=beta_hat, mu_hat=mu_hat, lambda_hat=lambda_hat,
      residuals=residuals, coefficients=coefficients, I=I, models=models,
      fittedvalues=fittedvalues)
}
```

As already described in Section 4.1, `estimateDParameters` expands the MANOVA subspaces $\tilde{L}_{[v]} := \{\tilde{Z}_v \tilde{B}_v \mid \tilde{B}_v \in \mathbf{M}(\tilde{T}_v \times I_v)\} \subseteq \mathbf{M}(N \times I_v)$, $v \in V$, by incorporating the dependency structure given by `D` to

$$L_{[v]} := \left\{ \begin{pmatrix} \tilde{Z}_v & y_{\prec v \succ} \end{pmatrix} \begin{pmatrix} \tilde{B}_v \\ \beta_v \end{pmatrix} \middle| \tilde{B}_v \in \mathbf{M}(\tilde{T}_v \times [v]), \beta_v \in \mathbf{M}(\prec v \succ \times [v]) \right\}$$
$$\subseteq \mathbf{M}(N \times I_v) \qquad \forall v \in V.$$

It has the `R` command `expand.model.frame` combine the design matrix $\tilde{Z}_v$ and the observation submatrix $y_{\prec v \succ}$. Then, the function assigns the parameter estimation and fitting to `lm`. It retrieves $\hat{\beta}_v$ from the `coefficients` matrix produced by `lm`, and uses

the `predict` function on the `lm` object with $y_{\prec v\succ} = 0$ to determine $\hat{\mu}_v$. The formula used to calculate $n\hat{\Lambda}_v$ is

$$\begin{aligned} y_{[v]}^T r_{[v]} &= y_{[v]}^T \left(y_{[v]} - (\hat{\mu}_v + y_{\prec v\succ}\hat{\beta}_v) \right) \\ &= y_{[v]}^T \left((y_{[v]} - y_{\prec v\succ}\hat{\beta}_v) - \hat{\mu}_v \right) \\ &= y_{[v]}^T (1 - \tilde{P}_v)(y_{[v]} - y_{\prec v\succ}\hat{\beta}_v) \\ &= y_{[v]}^T \tilde{Q}_v (y_{[v]} - y_{\prec v\succ}\hat{\beta}_v) \\ &= n\hat{\Lambda}_v. \end{aligned}$$

The construction of the pair $(\hat{\xi}, \hat{\Sigma})$ given the MLEs of the D-parameters $(\hat{\mu}_v, \hat{\beta}_v, \hat{\Lambda}_v \mid v \in V)$ in `reconstructDistParameters` is simply a translation of the Reconstruction Algorithm 3.6 into `R`.

```
reconstructDistParameters = function(beta, mu, lambda, D, I, ...) {
  V = colnames(D)

  n_ = dim(mu[[1]])[1]

  dimI = length(selectInd(I, V))

  sigma_hat = matrix(0, nrow=dimI, ncol=dimI)
  xi_hat = data.frame( dummy=1:n_ )

  cumIndices = c()

  columnnames = vector(mode="character", length=dimI)

  for(v_ in V) {
    sigma_hat[selectInd(I, v_), selectInd(I, pa(v_, D))] = t(beta[[v_]]) %*%
        sigma_hat[selectInd(I, pa(v_, D)), selectInd(I, pa(v_, D))]
    sigma_hat[selectInd(I, pa(v_, D)), selectInd(I, v_)] =
        t(sigma_hat[selectInd(I, v_), selectInd(I, pa(v_, D))])
    sigma_hat[selectInd(I, v_), selectInd(I, v_)] = lambda[[v_]] +
        t(beta[[v_]]) %*% sigma_hat[selectInd(I, pa(v_, D)), selectInd(I, v_)]
    sigma_hat[selectInd(I, v_), selectInd(I, setdiff(cumIndices, pa(v_, D)))] =
        t(beta[[v_]]) %*% sigma_hat[selectInd(I, pa(v_, D)),
        selectInd(I, setdiff(cumIndices, pa(v_, D)))]
    sigma_hat[selectInd(I, setdiff(cumIndices, pa(v_, D))), selectInd(I, v_)] =
        t(sigma_hat[selectInd(I, v_),
        selectInd(I, setdiff(cumIndices, pa(v_, D)))])

    cumIndices = c(cumIndices, v_)

    xi_hat[[v_]] = mu[[v_]]
```

```
    if (length(pa(v_, D)) >= 1) {
      xi_hat[[v_]] = mu[[v_]] + as.matrix(xi_hat[pa(v_, D)]) %*% beta[[v_]]
    }

    colnames(xi_hat[[v_]]) = NULL

    columnnames[selectInd(I,v_)] = colnames(mu[[v_]])
  }

  dimnames(sigma_hat) = list(columnnames, columnnames)

  xi_hat[["dummy"]] = NULL

  list(xi_hat=xi_hat, sigma_hat=sigma_hat)
}
```

Chapter 5

Conclusions

The mean-value space of a normal linear ADG model factorizes into MANOVA subspaces according to the model's independence graph. In the MANOVA model, we can easily perform an explicit maximum likelihood estimation of its parameters, and derive from these the normal linear ADG model's D-parameters. Given the set of the D-parameters and the reconstruction algorithm, we can construct the MLEs of the normal linear ADG model's parameters from the MLEs of the D-parameters.

The `R` package `nlADG` uses `lm` to perform the parameter estimation in the MANOVA submodels and implements the reconstruction algorithm to determine the MLEs of the normal linear ADG model's parameters.

In this thesis, we focused on the maximum likelihood estimation in a normal linear ADG model. As we worked with the results of Andersson and Perlman [1998] without actually studying their mathematical proofs, it was not necessary to state a very important property of normal linear ADG models: The likelihood function and the joint probability density function (pdf) of a normal linear ADG model factorize according to the model's Markov structure.

Given the MLEs of a normal linear ADG model's parameters, a follow-up topic of interest might be the testing of one normal linear ADG model against another. For further reading, the reader may be referred to Andersson and Perlman [1998].

Appendix A

Source Code

\Users\Public\nlADG\R\nlADG.R **Page 1**

```
selectInd = function(indexList, selectedIndices) {
  ind = c()

  for (v in selectedIndices) {
    ind = c(ind, indexList[[v]])
  }

  ind
}

assertDsubspace = function(formulas, D, ...) {
  for (v_ in colnames(D)) {
    for (p_ in pa(v_, D)) {
      designmatrix_v = as.matrix(model.matrix(formulas[[v_]]))
      expanded_designmatrix_v = cbind(designmatrix_v, as.matrix(model.matrix(formulas[
[p_]])))

      if (qr(designmatrix_v, ...)$rank < qr(expanded_designmatrix_v, ...)$rank) {
        stop("The regression system does not fulfill the D-linear subspace characteris
tics.")
      }
    }
  }
}

assertNoObservations = function(formulas, D, ...) {
  n_ = dim(model.frame(formulas[[1]]))[1]

  for (v_ in colnames(D)) {
    rankZ = qr(as.matrix(model.matrix(formulas[[v_]])), ...)$rank
    dimvandparents = dim(as.matrix(lm(formulas[[v_]])$coefficients))[2]
    for (p_ in pa(v_, D)) {
      dimvandparents = dimvandparents + dim(as.matrix(lm(formulas[[p_]])$coefficients)
)[2]
    }

    if (n_ < rankZ + dimvandparents) {
      stop("Not enough observations are given for the maximum likelihood estimation to
 exist and be unique.")
    }
  }
}

assertPDI = function(M, D, I) {
  D_ = diag(1, ncol=dim(D)[2], nrow=dim(D)[1])
  dimnames(D_) = dimnames(D)

  nd_ = list()
  for(v_ in colnames(D)) {
    nd_[[v_]] = setdiff(colnames(D), c(v_, pa(v_, D)))
  }

  for(v_ in colnames(D)) {
    D_ = D_ %*% D

    for(v__ in colnames(D)) {
      nd_[[v__]] = setdiff(nd_[[v__]], colnames(D_)[round(D_[v__,])==1])
    }
  }

  for(v_ in colnames(D)) {
    nd_v = nd_[[v_]]

    if(length(pa(v_, D)) >= 1 && length(nd_v) >= 1) {
      lM = M[selectInd(I, v_), selectInd(I, nd_v)]
```

```
      rM = M[selectInd(I, v_), selectInd(I, pa(v_, D))] %*% solve(M[selectInd(I, pa(v_
, D)),selectInd(I, pa(v_, D))]) %*% M[selectInd(I, pa(v_, D)), selectInd(I, nd_v)]

      diff = abs(lM - rM)
      diff = diff * (diff > sqrt(.Machine$double.eps))

      if(max(diff) > 0) {
        stop("The matrix is not in P(D,I)")
      }
    }
  }
}


nlADG = function(formulas, ...) UseMethod("nlADG")

nlADG.default = function(formulas, D, ...) {
  formulas = as.list(formulas)

  if(!hasArg(D) || !is.matrix(D)) {
    D = matrix(0, ncol=length(formulas), nrow=length(formulas), dimnames=list(names(fo
rmulas), names(formulas)))
  } else {
    D = as.matrix(D)
  }

  D = topSort(D)

  assertDsubspace(formulas, D, ...)
  assertNoObservations(formulas, D, ...)

  DParameterEst = estimateDParameters(formulas, D, ...)

  modelParameterEst = reconstructDistParameters(DParameterEst$beta_hat, DParameterEst$
mu_hat, DParameterEst$lambda_hat, D, DParameterEst$I, ...)

  model = list()

  model$coefficients = DParameterEst$coefficients
  model$sigma = modelParameterEst$sigma_hat
  model$xi = modelParameterEst$xi_hat
  model$fitted.values = modelParameterEst$xi_hat
  model$residuals = DParameterEst$residuals
  model$MANOVAmodels = DParameterEst$models
  model$ADG = D
  model$I = DParameterEst$I
  model$call = match.call()

  class(model) = "nlADG"

  model
}

estimateDParameters = function(formulas, D, ...) {
  n_ = dim(model.frame(formulas[[1]]))[1]

  mu_hat = list()
  beta_hat = list()
  lambda_hat = list()
  fittedvalues = data.frame( dummy=1:n_ )
  coefficients = list()
  residuals = data.frame( dummy=1:n_ )
  models = list()

  I = list()
```

```
  Icounter = 0

  for(v_ in colnames(D)) {
    model_v = lm(formulas[[v_]])

    if (length(pa(v_, D)) >= 1) {
      no_predictors = dim(model.matrix(formulas[[v_]]))[2]

      model_frame_expanded_v = expand.model.frame(formulas[[v_]], pa(v_, D))
      model_v = lm(model_frame_expanded_v)

      no_predictors_expanded = dim(as.matrix(model_v$coef))[1]

      beta_hat[[v_]] = matrix( as.matrix(model_v$coef)[(no_predictors+1):no_predictors
_expanded,] , nrow=no_predictors_expanded-no_predictors)

      clearedDataframe = model_frame_expanded_v
      for(p_ in pa(v_, D)) {
        if (is.matrix(clearedDataframe[[p_]])) {
          clearedDataframe[[p_]] = matrix(0, ncol=dim(as.matrix(model_frame_expanded_v
[[p_]]))[2], nrow=dim(as.matrix(model_frame_expanded_v[[p_]]))[1])
        } else {
          clearedDataframe[[p_]] = rep(0, dim(as.matrix(model_frame_expanded_v[[p_]]))
[1])
        }
      }
      mu_hat[[v_]] = as.matrix(predict(model_v, clearedDataframe))

      fittingDataframe = model_frame_expanded_v
      for (p_ in pa(v_, D)) {
        if (is.matrix(fittingDataframe[[p_]])) {
          fittingDataframe[[p_]] = fittedvalues[[p_]]
        } else {
          fittingDataframe[[p_]] = as.vector(fittedvalues[[p_]])
        }
      }
      fittedvalues[[v_]] = as.matrix(predict(model_v, fittingDataframe))
    } else {
      mu_hat[[v_]] = as.matrix(model_v$fitted.values)

      beta_hat[[v_]] = matrix(0, ncol=dim(mu_hat[[v_]])[2], nrow=0)

      fittedvalues[[v_]] = as.matrix(model_v$fitted.values)
    }

    original_values_v = as.matrix(model_v$residuals) + as.matrix(model_v$fitted.values
)

    residuals[[v_]] = original_values_v - fittedvalues[[v_]]

    lambda_hat[[v_]] = t(as.matrix(model_v$residuals)) %*% original_values_v / dim(mu_
hat[[v_]])[1]

    coefficients[[v_]] = as.matrix(model_v$coefficients)

    if (dim(mu_hat[[v_]])[2] >= 2) {
      rv_names = paste(v_, 1:dim(mu_hat[[v_]])[2], sep=".")
    } else {
      rv_names = v_
    }

    dimnames(beta_hat[[v_]]) = list(c(), rv_names)
    dimnames(mu_hat[[v_]]) = list(c(), rv_names)
    dimnames(lambda_hat[[v_]]) = list(rv_names, rv_names)
    dimnames(coefficients[[v_]]) = list(rownames(coefficients[[v_]]), rv_names)
```

\Users\Public\nlADG\R\nlADG.R **Page 4**

```
    models[[v_]] = model_v

    I[[v_]] = Icounter + 1:dim(mu_hat[[v_]])[2]
    Icounter = Icounter + dim(mu_hat[[v_]])[2]
  }

  fittedvalues[["dummy"]] = NULL
  residuals[["dummy"]] = NULL


  list(beta_hat=beta_hat, mu_hat=mu_hat, lambda_hat=lambda_hat, residuals=residuals, c
oefficients=coefficients, I=I, models=models, fittedvalues=fittedvalues)
}

reconstructDistParameters = function(beta, mu, lambda, D, I, ...) {
  V = colnames(D)

  n_ = dim(mu[[1]])[1]

  dimI = length(selectInd(I, V))

  sigma_hat = matrix(0, nrow=dimI, ncol=dimI)
  xi_hat = data.frame( dummy=1:n_ )

  cumIndices = c()

  columnnames = vector(mode="character", length=dimI)

  for(v_ in V) {
    sigma_hat[selectInd(I, v_), selectInd(I, pa(v_, D))] =  t(beta[[v_]]) %*% sigma_ha
t[selectInd(I, pa(v_, D)), selectInd(I, pa(v_, D))]
    sigma_hat[selectInd(I, pa(v_, D)), selectInd(I, v_)] = t(sigma_hat[selectInd(I, v_
), selectInd(I, pa(v_, D))])
    sigma_hat[selectInd(I, v_), selectInd(I, v_)] = lambda[[v_]] + t(beta[[v_]]) %*% s
igma_hat[selectInd(I, pa(v_, D)), selectInd(I, v_)]
    sigma_hat[selectInd(I, v_), selectInd(I, setdiff(cumIndices, pa(v_, D)))] = t(beta
[[v_]]) %*% sigma_hat[selectInd(I, pa(v_, D)),selectInd(I, setdiff(cumIndices, pa(v_,
D)))]
    sigma_hat[selectInd(I, setdiff(cumIndices, pa(v_, D))), selectInd(I, v_)] = t(sigm
a_hat[selectInd(I, v_), selectInd(I, setdiff(cumIndices, pa(v_, D)))])

    cumIndices = c(cumIndices, v_)

    xi_hat[[v_]] = mu[[v_]]

    if (length(pa(v_, D)) >= 1) {
      xi_hat[[v_]] = mu[[v_]] + as.matrix(xi_hat[pa(v_, D)]) %*% beta[[v_]]
    }

    colnames(xi_hat[[v_]]) = NULL

    columnnames[selectInd(I,v_)] = colnames(mu[[v_]])
  }

  dimnames(sigma_hat) = list(columnnames, columnnames)

  xi_hat[["dummy"]] = NULL

  list(xi_hat=xi_hat, sigma_hat=sigma_hat)
}

predict.nlADG = function(object, newdata, ...) {
  newdata = as.data.frame(newdata)
  models = object$MANOVAmodels
```

\Users\Public\nlADG\R\nlADG.R **Page 5**

```
  D = object$ADG
  V = colnames(D)
  n_ = dim(newdata)[1]


  prediction = data.frame( dummy=1:n_ )

  for (v_ in V) {
    data_v = newdata
    for (p_ in pa(v_, D)) {
      if (is.matrix(model.frame(models[[v_]])[[p_]])) {
        data_v[[p_]] = as.matrix(prediction[[p_]])
      } else {
        data_v[[p_]] = as.vector(prediction[[p_]])
      }
    }

    prediction[[v_]] = as.matrix(predict(models[[v_]], data_v))
  }

  prediction[["dummy"]] = NULL

  prediction
}

print.nlADG = function(x, ...) {
  cat("Call:\n")
  print(x$call)
  cat("\nMLE xi:\n")
  print(x$xi)
  cat("\nMLE sigma:\n")
  print(x$sigma)
}

summary.nlADG = function(object, ...) {
  print(object, ...)
}

print.summary.nlADG = function(x, ...) {
  print(x, ...)
}
```

Appendix B

Manual

Package 'nlADG'

April 15, 2010

Version 0.1-0

Title Regression in the Normal Linear ADG Model

Author Gruber, Lutz F.

Maintainer Gruber, Lutz F. <lgruber@mytum.de>

Depends R (>= 2.10.0), ggm

Description Regression in the Normal Linear ADG Model

License GPL-3

R topics documented:

`assertDsubspace` *Assert that the D-Linear Subspace Properties Hold*

Description

`assertDsubspace` compares the design matrices of the given formulas to hint for a possible violation of the `D`-linear subspace characteristics.

Usage

```
assertDsubspace(formulas, D, ...)
```

Arguments

formulas	A list of regression formulas. There must be a regression formula for each response variable.
D	The adjacency matrix of the normal linear ADG model's independence graph.
...	Additional parameters.

Details

The function assertDsubspace produces an error if the regression system does not admit the D-linear subspace characteristics.

Author(s)

Gruber, Lutz F.

References

Andersson, S. A. and Perlman, M. D. (1998) Normal Linear Regression Models With Recursive Graphical Markov Structure. *Journal of Multivariate Analysis* **66**, 133–187

See Also

nlADG, DAG

Examples

```
## The predictor variable
p <- 1:10

## Two response variables
x1 <- 2*p + rnorm(10, 0, 0.1)
x2 <- cbind(50 - 2*p + rnorm(10, 0, 0.1), 0.5*p + rnorm(10, 0, 0.1))

## The regression formulas of the response variables
formulas <- list(x1 = (x1 ~ p - 1), x2 = (x2 ~ p))

## Check for the D-linear subspace characteristics
assertDsubspace(formulas, D = DAG(x2 ~ x1)) #will go through
## Not run: assertDsubspace(formulas, D = DAG(x1 ~ x2)) #will produce an error
```

---

assertNoObservations

Assert that Enough Observations Are Given

Description

assertNoObservations checks if enough observations are given for the maximum likelihood estimates to exist and be unique. If not, the function produces an error.

Usage

```
assertNoObservations(formulas, D, ...)
```

Arguments

formulas	A list of regression formulas. There must be a regression formula for each response variable.
D	The adjacency matrix of the normal linear ADG model's independence graph sorted according to its topological order.
...	Additional parameters.

Author(s)

Gruber, Lutz F.

References

Andersson, S. A. and Perlman, M. D. (1998) Normal Linear Regression Models With Recursive Graphical Markov Structure. *Journal of Multivariate Analysis* **66**, 133–187

See Also

nlADG, DAG, topSort

Examples

```
## The predictor variable
v <- 1:5

## The response variable
x <- 25 - 2*v + rnorm(5)

## The regression formula of the response variable
formulas <- list(x=(x~v))

## The model's independence graph
D <- DAG(x~x)

## Check if enough observations are given
assertNoObservations(formulas, D) #will work

## Reduce the number of observations
x <- x[1:2]
v <- v[1:2]
## Not run: assertNoObservations(formulas, D) #will produce an error message
```

---

assertPDI *Assert that a Matrix is an Element of P(D, I)*

Description

assertPDI checks if the matrix M is an element of P(D, I). If not, the function produces an error.

Usage

```
assertPDI(M, D, I)
```

Arguments

M	The matrix for which to check if it is an element of P(D, I).
D	The adjacency matrix of the normal linear ADG model's independence graph sorted according to its topological order.
I	The index list I as returned from the nlADG class.

Author(s)

Gruber, Lutz F.

References

Andersson, S. A. and Perlman, M. D. (1998) Normal Linear Regression Models With Recursive Graphical Markov Structure. *Journal of Multivariate Analysis* **66**, 133–187

See Also

nlADG, DAG, topSort

Examples

```
## The predictor variable
p <- 1:10

## Two response variables
x1 <- 2*p + rnorm(10, 0, 0.1)
x2 <- cbind(50 - 2*p + rnorm(10, 0, 0.1), 0.5*p + rnorm(10, 0, 0.1))
x3 <- -p + rnorm(10, 0, 0.1)

## The regression formulas of the response variables
formulas <- list(x1 = (x1 ~ p - 1), x2 = (x2 ~ p), x3 = (x3 ~ p))

## The independence graph of the normal linear ADG model
D <- DAG(x2 ~ x1, x3 ~ x2)

## Fit the model
model <- nlADG(formulas, D)

## Check for the D-linear subspace characteristics
assertPDI(model$sigma, model$ADG, model$I) #will work
## Not run: assertPDI(matrix(rnorm(16), ncol=4), model$ADG, model$I) #will most likely nc
```

---

estimateDParameters

Estimate the D-Parameters of a Normal Linear ADG Model

Description

estimateDParameters calculates the MLEs of the D-parameters of a normal linear ADG model.

Usage

```
estimateDParameters(formulas, D, ...)
```

Arguments

formulas	A list of regression formulas. There must be a regression formula for each response variable.
D	The adjacency matrix of the normal linear ADG model's independence graph sorted according to its topological order.
...	Additional parameters.

Value

beta_hat	The estimate of the D-parameter beta.
mu_hat	The estimate of the D-parameter mu.
lambda_hat	The estimate of the D-parameter lambda.
residuals	The residuals.
coefficients	The estimates of the regression coefficient matrices.
I	A list which specifies the position of the response variables in the covariance matrix.
models	A list of MANOVA models corresponding to the response variables.
fittedvalues	The fitted values.

Note

The function is called by nlADG. It is not intended to be directly used by the end user.

Author(s)

Gruber, Lutz F.

References

Andersson, S. A. and Perlman, M. D. (1998) Normal Linear Regression Models With Recursive Graphical Markov Structure. *Journal of Multivariate Analysis* **66**, 133–187

See Also

nlADG, topSort

nlADG *Regression in the Normal Linear ADG Model*

Description

nlADG calculates the MLEs of the normal linear ADG model's parameters.

Usage

```
nlADG(formulas, ...)

## Default S3 method:
nlADG(formulas, D, ...)
## S3 method for class 'nlADG':
print(x, ...)
## S3 method for class 'nlADG':
summary(object, ...)
## S3 method for class 'nlADG':
print.summary(x, ...)
## S3 method for class 'nlADG':
predict(object, newdata, ...)
```

Arguments

formulas	A list of regression formulas. There must be a regression formula for each response variable.
D	The adjacency matrix of the normal linear ADG model's independence graph.
x	An object of the nlADG class.
object	An object of the nlADG class.
newdata	The data for which to predict the outcome.
...	Additional parameters, which may be used by functions deeper in the structure of this program.

Details

For any entry in formulas, the name of the entry has to match the name of the corresponding response variable.

Value

coefficients	The estimates of the regression coefficients.
sigma	The estimate of the covariance matrix.
xi	The estimate of the mean of the distribution.
fitted.values	The fitted values given the predictors.
residuals	The residuals.
MANOVAmodels	A list of MANOVA models corresponding to the response variables.
ADG	The sorted adjacency matrix of the model's independence graph D.
I	A list which specifies the position of the response variables in the covariance matrix.

Author(s)

Gruber, Lutz F.

References

Andersson, S. A. and Perlman, M. D. (1998) Normal Linear Regression Models With Recursive Graphical Markov Structure. *Journal of Multivariate Analysis* **66**, 133–187

See Also

DAG

Examples

```
## The predictor variable
p <- 1:10

## Two response variables
x1 <- 2*p + rnorm(10, 0, 0.1)
x2 <- cbind(50 - 2*p + rnorm(10, 0, 0.1), 0.5*p + rnorm(10, 0, 0.1))

## The regression formulas of the response variables
formulas <- list(x1 = (x1 ~ p - 1), x2 = (x2 ~ p))

## The normal linear ADG model's independence graph
D <- DAG(x2 ~ x1)

## Fit the model
model <- nlADG(formulas, D)


model$xi # Access the MLE of xi
model$sigma # Access the MLE of sigma

## Predict values
newdata <- data.frame(p=100:105)
predict(model, newdata)
```

`reconstructDistParameters`
Reconstruct the Normal Linear ADG Model's Parameters from their D-Parameters

---

Description

`reconstructDistParameters` reconstructs the normal linear ADG model's parameters from their `D`-parameters.

Usage

```
reconstructDistParameters(beta, mu, lambda, D, I, ...)
```

Arguments

`beta`	The `D`-parameter `beta`.
`mu`	The `D`-parameter `mu`.
`lambda`	The `D`-parameter `lambda`.
`D`	The adjacency matrix of the normal linear ADG model's independence graph sorted according to its topological order.
`I`	A list which specifies the position of the response variables in the covariance matrix.
`...`	Additional parameters.

Value

`xi_hat`	The reconstructed distribution parameter `xi`.
`sigma_hat`	The reconstructed distribution parameter `sigma`.

Note

The function is called by `nlADG`. It is not intended to be directly used by the end user.

Author(s)

Gruber, Lutz F.

References

Andersson, S. A. and Perlman, M. D. (1998) Normal Linear Regression Models With Recursive Graphical Markov Structure. *Journal of Multivariate Analysis* **66**, 133–187

See Also

`nlADG`, `topSort`

`selectInd` *Select Entries of the MLE sigma Matrix by their Names*

Description

Given the index list `I`, the `selectInd` function allows for the convenient selection of entries in the `sigma` matrix by the names of the respective response variables.

Usage

```
selectInd(indexList, selectedIndices)
```

Arguments

`indexList`	A list that specifies the rows and columns in which the values to the response variables can be found.
`selectedIndices`	A vector with the names of the response variables whose entries to select.

Value

The row and column indices of the selected response variables in the `sigma` matrix.

Author(s)

Gruber, Lutz F.

References

Andersson, S. A. and Perlman, M. D. (1998) Normal Linear Regression Models With Recursive Graphical Markov Structure. *Journal of Multivariate Analysis* **66**, 133–187

See Also

`nlADG`, `DAG`, `pa`

Examples

```
## Set up a normal linear ADG model
v <- 1:100
x <- 2*v + rnorm(100, 0, 5)
y <- x + 50 + rnorm(100, 0, 1)

D <- DAG(y~x)
formulas <- list(x=(x~v-1), y=(y~v))

model <- nlADG(formulas, D)

## The MLE of sigma
sigma <- model$sigma

## The indices of the response variable x
xInd <- selectInd(model$I, "x")
sigma[xInd, xInd]

## The indices to the response variables pa("y", model$ADG)
ypInd <- selectInd(model$I, pa("y", model$ADG))
sigma[ypInd, ypInd]

## The indices to the response variables c("x", "y")
xypInd <- selectInd(model$I, c("x", "y"))
sigma[xypInd, xypInd]
```

Index

Bibliography

Steen A. Andersson and Michael D. Perlman. Normal linear regression models with recursive graphical Markov structure. *Journal of Multivariate Analysis*, 66:133–187, 1998.

Jørgen Hoffmann-Jørgensen. *Probability with a view toward statistics.* Chapman & Hall, 1994.

Achim Klenke. *Probability Theory.* Springer-Verlag London Limited, 2008.

S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H.-G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.

Steffen Lilholt Lauritzen. *Graphical Models.* Oxford Statistical Science Series. Clarendon Press, Oxford, 1996.

Fred S. Roberts and Berry Tesman. *Applied Combinatorics.* Chapman & Hall / CRC, 2009.

Joe Whittaker. *Graphical Models in Applied Multivariate Statistics.* Wiley Series in Probability and Statistics. John Wiley & Sons, New York, 1990.