1.1. Mixed Graphical Models

Graphical models have become a popular way to abstract complex systems and gain insights into relational patterns among observed variables in a large variety of disciples such as statistical mechanics (Albert and Barabasi 2002), biology (Friedman, Linial, Nachman, and Pe'er 2000), genetics (Ghazalpour, Doss, Zhang, Wang, Plaisier, Castellanos, Brozell, Schadt, Drake, Lusis, and Horvath 2006), neuroscience (Huang, Li, Sun, Ye, Fleisher, Wu, Chen, and Reiman 2010) and psychology (Borsboom and Cramer 2013).

In many of these situations the dataset of interest consists of *mixed variables* that belong to differing domains such as binary, categorical, ordinal, counts, continuous and/or skewed continuous amongst others. As an example take internet-scale marketing data, in which companies in order to optimize revenue collect information such as clicked links (categorical), time spent on websites (possibly skewed continuous), browsing history (categorical), social media postings (count), friends in social networks (count), consumption of streaming services (categorical) and many more. Another example is a medical context, in which one is for instance interested in interactions between person characteristics such as gender (binary) or age (continuous), frequencies of behaviors (count), taking place of events (categorical) and the dose of a drug (continuous).

Due to their generality and versatile applicability, graphical models have been studied intensively. The most popular graphical model is the Gaussian Graphical model (GGM) (Lauritzen 1996), because of recent advances in high-dimensional statistics where efficient algorithms have been developed for estimation: the Graphical Lasso (Banerjee, El Ghaoui, and d'Aspremont 2008; Friedman, Hastie, and Tibshirani 2008) minimizes ℓ_1-penalized Gaussian log-likelihood and is implemented in the R-packages **glasso**- (Friedman and Tibshirani 2014) and **huge** (Zhao, Li, Liu, Roeder, Lafferty, and Wasserman 2015). Alternatively, a GGM can be estimated using neighborhood selection (Meinshausen and Bühlmann 2006), which is also implemented in the **huge** package. For the graphical model underlying the binary-valued Ising model (see e.g. Wainwright and Jordan 2008), there are neighborhood-selection based methods (Ravikumar, Wainwright, and Lafferty 2010; van Borkulo, Borsboom, Epskamp, Blanken, Boschloo, Schoevers, and Waldorp 2014) implemented in the R-package **IsingFit** (Borkulo, Epskamp, and Robitzsch 2014).

Despite the ubiquity of mixed datasets there are few methods to estimate *mixed* graphical models. Available models essentially consist of transforming all variables into Gaussian variables and then using above mentioned methods to estimate the GGM: Rue, Martino, and Chopin (2009) model mixed data through a latent Gaussian MRF, Dobra and Lenkoski (2011) model binary and ordered categorical variables using a thresholded latent Gaussian variable, and Liu, Lafferty, and Wasserman (2009) gaussianize continuous variables and then fit a Gaussian MRF. For a detailed overview, we refer the reader to Haslbeck and Waldorp (2015). The problem with these approaches is that one possibly loses information due to the transformations and that they cannot incorporate unordered (nominal) categorical variables. Until recently the only *parametric* model for mixed variables was the conditional Gaussian model that consists of Gaussian and categorical nodes (Lauritzen 1996), which is only computationally feasible for a small number of categorical variables as it involves conditioning on all possible states the categorical variables. This situation was improved by Yang, Baker, Ravikumar, Allen, and Liu (2014), who introduced a novel class of mixed exponential graphical model, in which each node can be a member from a different exponential family distribution.

Haslbeck and Waldorp (2015) leveraged this result together with results on generalized covariances (Loh and Wainwright 2013) to introduce a principled approach to estimated mixed graphical models. This approach, which is implemented in **mgm**, avoids possible information loss due to transformations, handles (nominal) categorical variables and scales to the high-dimensional setting.

1.2. Time-varying Mixed Graphical Models

For temporally evolving systems, time-varying graphical models provide additional information for understanding and predicting organizational processes, the diffusion of information, detecting vulnerabilities and the potential impact of interventions. An example is the developmental cycle of a biological organism, in which different genes interact at different stages of development. In a medical context, the aim could be to study the impact of a drug dose on a large number of physiological and psychological variables capturing the health of a patient. Yet another example is psychiatry, were one might be interested in the interaction of negative life events, social contacts and symptoms of psychological disorders (e.g. Depression).

While there is a large literature on modeling stationary graphical models, much less work has been done toward modeling graphical models that evolve over time. Most work belongs to two classes of time-varying models that make different assumptions about the *local stationarity* of the time-varying graph: One class of models assumes that the graph changes smoothly as a function of time and estimates the time-varying graphical model using kernel-smoothing (Song, Kolar, and Xing 2009; Zhou, Lafferty, and Wasserman 2010; Kolar, Song, Ahmed, and Xing 2010; Kolar and Xing 2009; Tao, Huang, Wang, Xi, and Li 2016; Chen and He 2015) or the fused LASSO (Monti, Hellyer, Sharp, Leech, Anagnostopoulos, and Montana 2014), which puts an extra penality on change over time. The other class of models assumes that the time-varying graphical model is piecewise constant and learns the change points (Kolar and Xing 2012; Gibberd and Nelson 2015, 2014). Most of above mentioned methods estimate a time-varying Gaussian graphical model, however, Kolar and Xing (2012) propose a method to estimate a time-varying Ising model.

In the **mgm**-package, we implement a combination of the estimation method for mixed graphical models mentioned above (Haslbeck and Waldorp 2015) with a kernel-smoothing approach as in Kolar *et al.* (2010) and thereby extend available estimation methods for smoothly time-varying graphical models to the mixed case.

1.3. Time-varying Vector Autoregressive Models

Vector autoregressive (VAR) models are a popular model for multivariate time series in many disciplines (see e.g. Hamilton 1994; Pfaff 2008). In its simplest form with a lag of order one, in this model all observations X^{t-1} at time $t-1$ are regressed on each of the observations X_i^{t-1} at time t. The VAR model is well-known in the econometric literature (see e.g. Toda and Yamamoto 1995) but has recently also been used in other disciplines, for example to analyze experience sampling data in the social and behavioral sciences (see e.g. Bringmann, Vissers, Wichers, Geschwind, Kuppens, Peeters, Borsboom, and Tuerlinckx 2013).

Important assumptions of this model are (a) that all measurements are equidistant in time and (b) that the underlying graphical model is constant across time, i.e. the mean and covariances are time-invariant. In most applications, the multivariate time series is treated as jointly Gaussian, however, VAR models can also be applied to other distributions within

the Generalized linear model framework (for a VAR model with binary responses see e.g. Kuppens, Allen, and Sheeber 2010). In **mgm**, we implement a method that estimates a ℓ_1-regularized VAR model from mixed data. In addition, we provide an implementation of a *time-varying mixed VAR model* that relaxes assumption (b) that the underlying graphical model must be time-invariant. In the context of analyzing experience sampling data for instance, this enables researches to identify varying influences between, say, psychological states and psychopathological symptoms, possibly as a function of interventions or life events.

1.4. Overview

In Section 2 we give an overview of the theory used to estimate *stationary* mixed graphical models (2.1), show how to sample from a pairwise mixed graphical model (2.2), illustrate the estimation of mixed graphical models (2.3), provide performance benchmarks of the method in realistic situations (2.4) and illustrate the method using a medical dataset (2.5).

In Section 3 we extend mixed graphical models to the *time-varying* case. After introducing the ideas behind kernel-smoothed neighborhood regression (3.1), we show how to generate data from a time-varying pairwise mixed graphical model (3.2), illustrate how to estimate time-varying mixed graphical models (3.3), give performance benchmarks for time-varying graphs in a realistic setting (3.4) and use our method to estimate a time-varying mixed graphical model from a large time series in psychopathology (3.5).

2. Stationary Mixed Graphical Models

2.1. Background

In this background section we first define basic concepts related to undirected graphical models (2.1.1) and then introduce the class of mixed graphical models that is used throughout the paper (2.1.2). We then describe our estimation method based on generalized covariance matrices (2.1.3) and illustrate the exact estimation algorithm that is implemented in **mgm** (2.1.4).

Undirected Graphical Models

Undirected graphical models (or Markov random fields) are families of probability distributions that respect a graph $G = (V, E)$ in terms of conditional independencies, which consists of a collection of nodes $V = \{1, \ldots, p\}$ and a collection of edges $\mathrm{E} \subseteq V \times V$. The neighborhood $N(s)$ of node s is the set of all nodes $t \in V$ that are connected with s by an edge $(s, t) \in E$. Note that the neighborhoods $N(s)$ of all $s \in V$ together define the graph G. The degree $deg(s)$ of a node is the number of edges it has to other nodes. We define a node cutset as a subset of nodes $S \subset V$ that breaks the graph G in two or more nonempty components. A clique is a fully connected sub-component $C \subseteq V$ such that $(s, t) \in E$ for all $s, t \in C$, while $s \neq t$.

We also require the notion of triangulation. Take a sequence of distinct nodes $\{s_1, \ldots, s_\ell\}$ such that $(s_i, s_{i+1}) \in E$ for all $1 \leq i \leq \ell - 1$, $(s_\ell, s_1) \in E$ and no other nodes in the cycle are connected by an edge. A triangulated graph is a graph in which no such cycle exists with length larger than 3.

Associate a random variable X_s to each node in V. For three subsets of nodes, A, B, and U, we write $X_A \perp\!\!\!\perp X_B | X_U$ to indicate that the random vector X_A is independent of X_B when conditioning on X_U. We can now define Markov random fields in terms of the global Markov property:

Definition 1 *(Global Markov property). If $X_A \perp\!\!\!\perp X_B | X_U$ whenever U is a vertex cutset that breaks the graph into disjoint subsets A and B, then the random vector $X := (X_1, \ldots, X_p)$ is Markov with respect to the graph G.*

Note that the neighborhood set $N(s)$ is always a node cutset for $A = \{s\}$ and $B = V \setminus \{s \cup N(s)\}$.

For strictly positive probability distributions, the global Markov property is equivalent to the Markov factorization property (Lauritzen 1996). Consider for each clique $C \in \mathcal{C}$ a clique-compatibility function $\psi_C(X_C)$ that maps configurations $x_C = \{x_s, s \in C\}$ to $\mathbb{R}^+$ such that ψ_C only depends on the variables X_C corresponding to the clique C.

Definition 2 *(Markov factorization property). The distribution of X factorizes according to G if it may be represented as a product of clique functions*

$$P(X) \propto \prod_{C \in \mathcal{C}} \psi_C(X_C). \tag{1}$$

Because we focus on strictly positive distributions, we can represent (1) in terms of an exponential family associated with the cliques C in G. We use this representation in the next section to model different conditional univariate members of the exponential family in one joint distribution.

Mixed Joint Distributions

Yang *et al.* (2014) showed that under certain conditions one can construct a multivariate distribution over variables of different domains by factoring univariate conditional members of the exponential family to a joint distribution[1]. The resulting multivariate distribution then factors to a graph defined by the node-neighborhoods of the conditional univariate distributions (for details see Yang *et al.* 2014). Here, we show only a model including pairwise interactions, but the model naturally generalizes to k-order interactions.

Let $\phi(\cdot)$ and $C(\cdot)$ be the sufficient statistic function and the base measure for the exponential family at hand, respectively. Further, let θ be a set of parameters. Then the univariate conditional distribution of X_s conditioned on all other variables $X_{\setminus s}$ has the form

$$P(X_s | X_{\setminus s}) = \exp\left\{ \theta_s \phi_s(X_s) + \sum_{t \in N(s)} \theta_{st} \phi_s(X_s) \phi_t(X_t) + C_s(X_s) - \bar{\Phi}(\theta) \right\}, \tag{2}$$

[1]In case the joint distribution includes two univariate distributions with infinite domain, it is not normalizable if *neither* both distributions are infinite only from one side *nor* the base measures are bounded with respect to the moments of the random variables (for details see Yang *et al.* 2014).

where $\bar{\Phi}(\theta)$ is the log normalizing constant.

When factoring out $\phi_s(X_s)$, one obtains the typical exponential family form, where the natural parameter consists of a threshold θ_s plus a linear combination of all variables in the neighborhood of node s. Factoring several univariate conditional distributions as in (2) gives rise to the joint distribution

$$P(X) = \exp\left\{\sum_{s\in V}\theta_s\phi_s(X_s) + \sum_{s\in V}\sum_{t\in N(s)}\theta_{st}\phi_s(X_s)\phi_t(X_t) + \sum_{s\in V}C_s(X_s) - \Phi(\theta)\right\}. \quad (3)$$

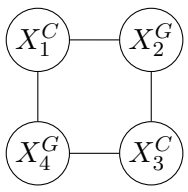
Note that $\phi(\cdot)$ and $C(\cdot)$ are defined by the particular member of the exponential family. For example, if X_s is a Gaussian variable with known variance, $\phi_s(X_s) = X_s/\sigma_s$ and $C_s(X_s) = X_s^2/2\sigma_s^2$.

In the following section we describe how we estimate the MRF underlying a model of the form (3) generalized to k-order interactions.

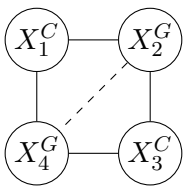
Inverses of Generalized Covariance Matrices

A classic corollary of the Hammersley-Clifford theorem (Lauritzen 1996) states that the inverse covariance matrix Γ of a multivariate Gaussian distribution is graph-structured, that is, $\Gamma_{a,b} = 0$ whenever $(a, b) \notin E$. However, this property only holds for the multivariate Gaussian and the relation between the inverse covariance matrix and MRF is generally unknown for other distributions. Loh and Wainwright (2013) showed for discrete data that when triangulating a graph G and augmenting sufficient statistics corresponding to all cliques $\widetilde{C} \in \widetilde{\mathcal{C}}$ in the triangulated graph $\widetilde{G}$ to the covariance matrix, then the inverse of this *generalized covariance matrix* reflects the graph structure of the triangulated graph $\widetilde{G}$. In Haslbeck and Waldorp (2015), we generalized the result of Loh and Wainwright (2013) to the mixed Markov random field underlying the mixed joint distribution introduced by Yang *et al.* (2014).

As an example, take an Ising-Gaussian model $X = \{X_1, X_2, X_3, X_4\}$, where X_1, X_3 are Bernoulli random variables and X_2, X_4 are Gaussian random variables, and let the underlying MRF be the graph in Figure 1 (a). Figure 1 (c) shows the empirically computed inverse Γ of the covariance matrix of the random vector X. As the entry $\Gamma_{1,3}$ is nonzero and there is an absent edge at $(3, 1)$ in G, Γ is not graph-structured.



(a)



(b)

$$\Gamma = \begin{array}{c} \\ X_1 \\ X_2 \\ X_3 \\ X_4 \end{array} \begin{array}{c} \begin{array}{cccc} X_1 & X_2 & X_3 & X_4 \end{array} \\ \begin{pmatrix} 5.03 & 0.50 & -0.01 & 0.50 \\ 0.50 & 1.00 & 0.50 & 0 \\ -0.01 & 0.50 & 5.03 & 0.50 \\ 0.50 & 0 & 0.50 & 1.00 \end{pmatrix} \end{array}$$

(c)

Figure 1: (a) Ising-Gaussian Markov random field, (b) A triangulation of of (a), (c) Inverse covariance matrix.

Let $\widetilde{\mathcal{C}}$ be the set of all cliques in the triangulated graph $\widetilde{G}$ in Figure 1 (b) and define the random vector $\Psi(\phi(X); \mathcal{A}) := \{\phi_A(X_A), A \in \mathcal{A}\}$. We can now show empirically, that the

11×11 generalized covariance matrix $\text{cov}(\Psi(X;\widetilde{\mathcal{C}}))^{-1}$ is graph-structured with respect to the triangulated graph $\widetilde{G}$ (for the formal theorem see Theorem 1 in Haslbeck and Waldorp 2015). While this result allows to estimate the graph structure of the triangulated graph $\widetilde{G}$ via estimating Γ, the direct application of the result does not yet provide a way to estimate the *original graph* G.

This problem is solved by constructing a smaller inverse covariance matrix Γ^s that includes all nodes and all candidate neighborhoods of node s with size up to d, where d is the degree of node s in the true graph G (Haslbeck and Waldorp 2015). This inverse Γ^s is graph structured with respect to the neighborhood $N(s)$ *in the original graph* G, that is, the entries $\Gamma^s_{s,t} = 0$ whenever $(s,t) \notin E$. As an example, take a graph with four nodes and let $d(s) = 2$. Then for each node s we see that its possible neighborhoods are always triangulated.

Note that the set of all neighborhoods $N(s)$ defines the graph G. We can therefore obtain the original graph G by estimating the row s in each of these smaller inverse covariance matrices. Because row s in Γ^s is a scalar multiple of the regression vector of X_s upon $X_{\setminus s}$ (e.g. Lauritzen 1996), we can use linear regression to estimate the row s of the inverse covariance matrix. In the following section we use this relationship in a nodewise graph-estimation algorithm.

Nodewise estimation algorithm

We recover $N(s)$ by solving the linear regression problem $E(X_s|X_{\setminus s}) = \theta_0 + X_{\setminus s}^T\theta$. In order to obtain a sparse solution, squared loss is minimized together with the ℓ_1-norm of the parameter vector

$$\min_{(\theta_0,\theta)\in\mathbb{R}^p} \left[\frac{1}{N} \sum_{i=1}^{N} (y_i - \theta_0 - X_{\setminus s;i}^T \theta)^2 + \lambda_n ||\theta||_1 \right]. \tag{4}$$

For non-Gaussian variables a (link) function is used to define a linear relation between the dependent variable X_s and its predictors (for details see Friedman, Hastie, and Tibshirani 2010). The regularization parameter λ_n and the smallest detectable effect size τ_n are assigned to the scaling

$$\lambda_n \gtrsim \sqrt{d}||\theta||_2 \sqrt{\frac{\log p}{n}}, \qquad \tau_n \asymp \sqrt{d}||\theta||_2 \sqrt{\frac{\log p}{n}}, \tag{5}$$

where d is the degree in the true graph G, $||\theta||_2$ is the ℓ_2-norm of the population parameter vector and $\gtrsim$ and $\asymp$ stands for asymptotically larger and equivalent, respectively. This scaling reflects the sparsity assumption under which Algorithm 1 is consistent (for details see Loh and Wainwright 2013). Note that the threshold τ_n is computed using the ℓ_2-norm of the *estimated* parameter vector $\widehat{\theta}$. In Algorithm 1, we select λ_n such that it minimizes the cross validated deviance or alternatively the extended Bayesian Information Criterion (EBIC) (Foygel and Drton 2011)

$$EBIC = -2LL(\hat{\theta}) + J\log(n) + 2\gamma J \log(p-1), \tag{6}$$

where $LL(\hat{\theta})$ is the log likelihood of the model, J is the number of parameters, p is the number of variables and γ is the hyperparameter weighting the extra penality $2J\log(p-1)$.

After minimizing (4), all entries in the parameter vector $\widehat{\theta}$ below the threshold τ_n are set to zero. Finally parameters comprising interactions involving categorical variables are combined in one edge-weight by taking the sum of the absolute values of all parameters.

These steps give rise to the following nodewise estimation algorithm:

Algorithm 1 *(Nodewise estimation algorithm for stationary graphs)*

for *all* $s \in V$ **do**

1. *Construct the prediction vector* X *including interactions corresponding to all required candidate neighborhoods*
2. *Select a regularization parameter* λ_n *using cross-validation or EBIC*
3. *Solve the* ℓ_1 *regularized regression problem in (4) and denote the solution by* $\hat{\theta}$.
4. *Threshold the entries of* $\hat{\theta}$ *at* τ_n

end

5. *Combine parameters into a weighted adjacency matrix using the AND- or OR-rule*

The neighborhood estimates for all $s \in V$ combined give rise to an asymmetric weighted-adjacency matrix, where the values above and below the diagonal stem from two different regressions. In the next step these pairs of edge-weights are combined into one edge parameter using the AND-rule (both parameters have to be nonzero to have a present edge) or OR-rule (at least one parameter hast to be nonzero to have a present edge). This procedure returns a weighted adjacency matrix, whose zero-pattern comprises the estimated underlying Markov random field.

Note that the computational complexity of Algorithm 1 is $\mathcal{O}(p2^d \log p)$. Therefore, the algorithm is only applicable to graphs with a degree that is bounded by d. This means that we have to make an assumption regarding d in the true graph. However, simulation results show that Algorithm 1 remains consistent, even when the sparsity assumption reflected by τ_n is largely violated (Haslbeck and Waldorp 2015). This suggests that in practice the choice of d should be motivated by the assumed highest degree of interactions in the true graph G. Importantly, this is a much less restrictive assumption. For instance in random graphs with constant sparsity, d grows with the number of nodes, while the maximal degree of interactions can remain constant. Note that assumptions about the maximal degree of interactions are common practice, for instance the multivariate Gaussian distribution or the Ising model include only pairwise interactions.

2.2. Sampling from a Mixed Graphical Model

In this section we use an example to show how to use the function `mgmsampler` to sample cases from a pairwise mixed joint distribution including Gaussian, Exponential, Poisson and categorical variables. In Section 2.3 we recover the mixed Markov random field underlying this mixed joint distribution from data using the `mgmfit` function.

We specify a mixed graphical model with p variables by defining a vector `type` of length p indicating the type of random variable, a vector `lev` of length p indicating the number of

levels of categorical variables, a $p \times p$ symmetric weighted adjacency matrix `graph` and a list `thresh` containing thresholds for each variable. In the following example, we specify a pairwise mixed distribution as in (3) over a Gaussian, a Poisson, an Exponential, and two categorical random variable with three and two categories:

```
type <- c("g", "p", "e", "c", "c")
```

where `g`, `p`, `e` and `c` stand for Gaussian, Poisson, Exponential and Categorical. Next, we specify the number of levels for the categorical variable

```
lev <- c(1, 1, 1, 3, 2),
```

where continuous random variables have number of levels one and the categorical variables have three and two categories. We can now specify the weighted adjacency matrix

```
graph <- matrix(0,5,5)
graph[2,4] <- graph [4,2] <- -.5 # edge between P and C1 with weight -.5
graph[1,5] <- graph[5,1] <- .9 # edge between G and C2 with weight .9
graph[1,4] <- graph [4,1] <- .8 # edge between G and C2 with weight .8
graph[4,5] <- graph [5,4] <- 1.5 # edge between C1 and C2 with weight 1.5
```

which gives us the following symmetric $p \times p$ weighted adjacency matrix:

$$\begin{array}{c} \\ G \\ P \\ E \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{ccccc} G & P & E & C_1 & C_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0.8 & 0.9 \\ 0 & 0 & 0 & -0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ .8 & -0.5 & 0 & 0 & 1.5 \\ 0.9 & 0 & 0 & 1.5 & 0 \end{pmatrix} \end{array} \tag{7}$$

Finally, we specify thresholds

```
thresh <- list(1, .7, .1, c(.1, .2, .3), c(0, .1)),
```

where we specify one threshold for each continuous variable and one threshold for each category of each categorical variable.

Note that interactions involving categorical variables involve more than one parameter. In the over-complete representation we use here for sampling, an interaction between two categorical variables has $m \cdot k$ parameters, where m and k are the number of categories of the two categorical variables. In case of an interaction between a continuous random variable and a categorical variable, there are m parameters, where m is the number of categories of the categorical variable.

These interactions can be specified explicitly in the form of the matrix (8) via `parmatrix` argument. The diagonal elements of the parameter matrix are not specified, as they would refer to an interaction of a variable with itself. For interactions between continuous variables there is one parameter. The parameters of an interaction involving a continuous and categorical variable are interpreted as follows: for example, the entry 0.9 indicates that positive values of variable G are positively related to the probability of observing category 1 of C_2. Specifically, $\theta_{st} = 0.9$ in (3) if $X_s = \text{G}$ and $X_t = \mathbb{I}_{\{C_2=1\}}$, the indicator variable for the event $C_2 = 1$. In the case of interactions between two categorical variables the parameter corresponds to the

interaction between the two corresponding indicator variables.

$$\begin{array}{c} \\ G \\ P \\ E \\ C_{11} \\ C_{12} \\ C_{13} \\ C_{21} \\ C_{22} \end{array} \begin{array}{c} \begin{array}{cccccccc} G & P & E & C_{11} & C_{12} & C_{13} & C_{21} & C_{22} \end{array} \\ \begin{pmatrix} NA & 0 & 0 & 0.8 & 0 & 0 & 0.9 & 0 \\ 0 & NA & 0 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & NA & 0 & 0 & 0 & 0 & 0 \\ 0.8 & -0.5 & 0 & NA & NA & NA & 1.5 & 0 \\ 0 & 0 & 0 & NA & NA & NA & 0 & 1.5 \\ 0 & 0 & 0 & NA & NA & NA & 0 & 1.5 \\ 0.9 & 0 & 0 & 1.5 & 0 & 0 & NA & NA \\ 0 & 0 & 0 & 0 & 1.5 & 1.5 & NA & NA \end{pmatrix} \end{array} \tag{8}$$

As an alternative to specifying all interaction parameters by hand one can specify only the weighted adjacency matrix (7) via the `graph` argument and the sampling function `mgmsampler` computes a parameter matrix (8). We then use the following (arbitrary) mapping: for interactions between categorical variables we use a generalization of the Potts parameterization (see e.g. Wainwright and Jordan 2008) that extends to the case where $m \neq k$. If we have m, k categories, X_s, X_t are two arbitrary nodes and $j \in [1, \ldots, m]$ and $i \in [1, \ldots, k]$, then $\theta_{st,ji} = 1$ when $j = i$ and $\theta_{st,ji} = 0$, when $j \neq k$. In case $m > k$, we have $\theta_{st,ji} = 1$ for $j > k$ and $i = k$ and $\theta_{st,ji} = 0$ otherwise. Interactions between categorical variables X_s with m categories and continuous variables X_t are specified as follows: if $X_s = j'$ where $j' \leq m/2$, then $\theta_{sj,t} = 1$ and otherwise $\theta_{sj,t} = 0$. Applying these rules to (7) results in (8). Note that this mapping is arbitrary and only implemented to provide a quick way of generating samples from a mixed graphical model.

We now sample from the model specified in (8) by loading the **mgm**-package and calling the sampling function `mgmsampler`

```
library(mgm)
set.seed(23)
data <- mgmsampler(n = 500, type, lev, graph, thresh, nIter = 250)
```

where `n` indicates the number of cases sampled and `nIter` specifies the number of iterations in the Gibbs sampler. The columns in the sampled data matrix have the order as specified in the input, i.e. the first variable is the Gaussian variable, etc.:

```
> round(data[1:4,],2)
[,1] [,2] [,3] [,4] [,5]
[1,] 1.41    1 0.03    3    2
[2,] 1.23    1 0.38    1    1
[3,] 3.03    1 0.06    1    1
[4,] 1.27    1 0.11    3    2
```

The model in (3) specified by `thresh` and `graph` / `parmatrix` is only normalizable if the following constraints are satisfied: first, the Gaussian sub-matrix must be positive-definite and second, any the interaction of any combination of Exponential and Poisson nodes has to have a positive weight and the thresholds have to be positive. The latter requirement keeps the natural parameter of Exponential and Poisson nodes positive, $\lambda > 0$. For details see Yang *et al.* (2014) and Yang, Ravikumar, Allen, and Liu (2013).

2.3. Estimating Mixed Graphical Models

In this section we use the function `mgmfit` to recover the true graphical model from the data generated in Section 2.2:

```
set.seed(2015)
fit <- mgmfit(data = data, type = type, lev = levs,
              lambda.sel = "CV", gam = .25, d = 2,
              rule.reg = "AND")
```

where `data` is an $n \times p$ data matrix and, as above, `type` is a vector that specifies the type of the conditional random variable (`g`, `p`, `e`, `c` for Gaussian, Poisson, Exponential and Categorical respectively) and `lev` is a vector specifying the number of categories for categorical variables (for continuous variables 1).

The function `mgmfit` has several tuning parameters: `lambda.sel` specifies which method is used for the selection of the ℓ_1-regularization parameter. `"CV"` stands for cross-validation minimizing the deviance of the model. The number of folds can be specified by `folds`. The alternative is `"EBIC"`, where the regularization parameter is selected using the Extended Bayesian Information Criterion (EBIC) which is known to perform well in selecting sparse graphs (Foygel and Drton 2014). `gam` specifies the hyper-parameter γ in the EBIC, which controls a penalty on neighborhood size (for details see Foygel and Drton 2014). `d` specifies the assumption regarding the maximal degree of interactions in the true model (see Section 2.1.4). In our example we know that there are at most pairwise interactions and we therefore choose `d = 2`. `rule.reg` specifies whether the two estimates for an edge (see Algorithm 1) are combined with the AND-rule (`"AND"`, an edge is set to be present if both parameters are nonzero) or the OR-Rule (`"OR"`, an edge is set to be present when at least one of the parameters is nonzero).

We now estimate the mixed Markov random field we created above using Algorithm 1 and get the following matrix of parameter estimates

$$
\begin{array}{c} \\ G \\ P \\ E \\ C_{11} \\ C_{12} \\ C_{13} \\ C_{21} \\ C_{22} \end{array}
\begin{array}{c}
\begin{array}{cccccccc} G & P & E & C_{11} & C_{12} & C_{13} & C_{21} & C_{22} \end{array} \\
\left(\begin{array}{cccccccc}
NA & 0 & 0 & NA & -0.31 & -0.33 & NA & -0.55 \\
0 & NA & 0 & NA & 0 & 0 & NA & 0.16 \\
-0.62 & 0 & NA & NA & 1.46 & -3.29 & NA & -1.04 \\
0.42 & 0 & 0 & NA & NA & NA & NA & -2.58 \\
0 & 0 & 0 & NA & NA & NA & NA & 0 \\
0 & 0 & 0 & NA & NA & NA & NA & 0 \\
0 & 0 & 0 & NA & -1.01 & -1.13 & NA & NA \\
0 & 0 & 0 & NA & 1.01 & 1.13 & NA & NA
\end{array} \right)
\end{array}
\tag{9}
$$

which is stored in the fit object (`fit$mpar.matrix`). Note that the parameterization in (9) is different than in (8). This is due to the fact that we use the parameterization of the **glmnet** package (Friedman *et al.* 2010), which we use to estimates each row by all columns. If X_r and X_t are categorical variables with m and k categories and X_s is a continuous variable, **glmnet** estimates m parameters when regressing X_s on X_r, $m(k-1)$ parameters when regressing X_t on X_r and $(m-1)$ parameters when regressing X_r on X_s (for details see Friedman *et al.*

2010). Accordingly, in (9), the columns C_{11} and C_{21} do not contain parameter estimates, as they correspond to the reference category in the multinomial regression model.

In order to obtain a single edge weight conditional dependence between two variables, we combine parameters of interactions involving categorical variables by taking the average of the absolute parameter values. Applying then the `AND`-rule as described above produces the following weighted adjacency matrix:

$$\begin{array}{c} \\ G \\ P \\ E \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{ccccc} G & P & E & C_1 & C_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0.31 & 0.42 \\ 0 & 0 & 0 & 0.18 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.31 & 0.18 & 0 & 0 & 1.09 \\ 0.42 & 0 & 0 & 1.09 & 0 \end{pmatrix} \end{array} \tag{10}$$

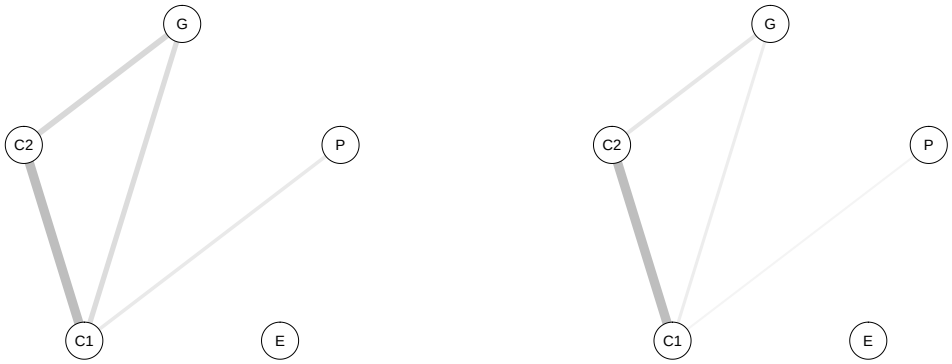
This is the main output of the fit function `mgmfit` and is saved in the fit object (`fit$wadj`). Note that `fit$wadj` is a function of the *absolute value* of the estimated parameters and it therefore contains no information about the signs of parameters. The edge weights can therefore be interpreted as the strength of conditional dependence. We take the absolute value over *all* parameters, also those describing the interaction between two continuous variables, where a sign is defined. We do this in order to avoid confusion about the meaning of positive edge weights (defined and positive or undefined).

However, available information about signs is saved in `fit$signs`. This matrix of the same dimension as `fit$wadj` indicates whether a edge parameter has a positive (1), negative (-1) or undefined (0) sign. This information can be used to visualize a graphical model as in Figure 4: here, edge weights are either green (positive), red (negative) or grey (undefined). All estimated parameters including signs are stored in `fit$mpar.matrix`. From this matrix one can obtain the exact nature of interactions involving categorical variables.

In Figure 2.3 compare the estimated weighted adjacency matrix in (10) with the true weighted adjacency matrix in (7) from which we generated the data. We see that we recovered all edges correctly and did not estimate incorrect additional edges. While the zero pattern in (10) is the estimated Markov random field, the edge-weights can be interpreted as relative importance in the graph. Figure 2.3 provides a visual comparison of the true and estimated weighted adjacency matrix. While the absolute value of the estimated parameters in (10) is lower than the true parameters in (7) due to regularization, we can see in Figure 2.3 that the relative sizes of the estimated parameters closely matches the relative sizes of the true parameters.

2.4. Benchmarks

In Haslbeck and Waldorp (2015), we provide extensive benchmarks for realistic situations for Algorithm 1 with `lambda.sel = "CV"` and `rule.reg = "AND"`. In some datasets, however, cross-validation cannot be applied because it requires all categories of all variables to be present in every possible fold. This condition is unlikely to be satisfied in situations, where some categories have low frequency. Therefore, we also provide the option to select the regularization parameter with the extended Bayesian Information Criterion (EBIC) (Foygel and Drton 2014). In addition, the EBIC can be an attractive alternative for large data sets for which cross-validation is computationally expensive. Figure 3 shows benchmarks of our algorithms with the two options for λ-selection in an Ising-Gaussian model, where the underlying



(a) True (b) Estimated

Figure 2: Visualization of the true and the estimated weighted adjacency matrix.

Markov random field is a random graph with $P_{edge} = 0.1$. Note that we have at most pairwise interactions in the true model and theory requires to augment interactions of order $d = 2$. This is the reason why Algorithm 1 with $d = 1$ does not converge in precision. It is known that the Bayesian Information Criterion (BIC) (Schwarz 1978) is more conservative than the cross-validation, and we see also in Figure 3 that Algorithm 1 with `lambda.sel = "EBIC"` is more conservative than with `lambda.sel = "CV"` and correspondingly that precision is slightly lower for cross validation.

2.5. Application to Autism Dataset

We estimate the conditional independence structure of a dataset capturing various aspects of the life of N = 3521 individuals diagnosed with ASD from the Netherlands (for details see Begeer, Wierda, and Venderbosch 2013). The dataset includes 28 variables of different types, for instance type of housing (categorical), number of treatments (count), and age (continuous). For estimation, we used the exact same configuration as in the example in Section 2.3 except that we used `lambda.sel = "EBIC"` because of the size of the dataset. Figure 4 shows the estimated graph drawn using the force-directed Fruchterman-Reingold algorithm (Fruchterman and Reingold 1991) algorithm as implemented in the **qgraph**-package (Epskamp, Costantini, Cramer, Waldorp, and Borsboom 2015).

The graphical model in Figure 4 provides a rich picture of the unique dependencies between variables. For dependencies between continuous variables we can indicate whether the sign of interaction parameters is positive (green) or negative (red). For dependencies between categorical variables, no sign is defined (grey). While an in-depth interpretation of the graph would go beyond the scope of the present paper, we look at the connections of the variable satisfaction with care. It is central in the graph and connects to well-being and satisfaction with work, which is itself connected to well-being. The latter indicates that both variables are uniquely related to well-being. On the other hand we see that satisfaction with care connects to satisfaction with treatment, medication and education, whereas the latter are not connected to well-being. This indicates that if these satisfaction ratings are related to well-being, then they are related to well-being through satisfaction with care.

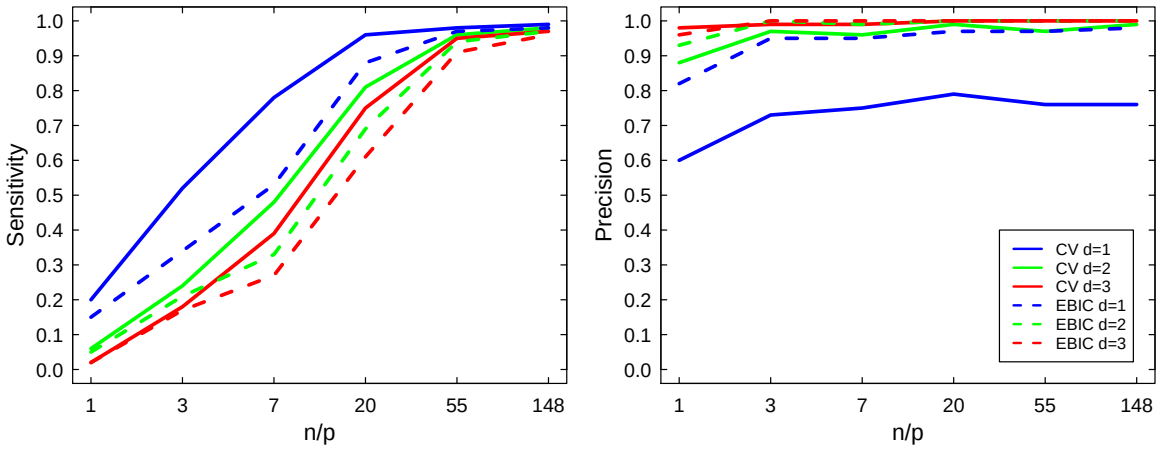


Figure 3: Comparison of the performance of Algorithm 1 when selecting the regularization parameter using cross-validation or the EBIC. The data is generated from a Ising-Gaussian model with a random graph with $P_{edge} = .1$ as underlying MRF. n/p is the ratio between the number of variables p and observations n.

3. Time-varying Mixed Graphical Models

In this section we extend the estimation of mixed graphical models discussed in Section 2 to the estimation of mixed graphical models that vary over time, i.e. to the situation in which each observation (time step) is potentially generated by a different graphical model.

3.1. Background

In Section 2 we considered observations that are independent and identically distributed. Here we drop the second condition and allow graphs to vary across observations (time steps). We define a time index set $\mathcal{T}_n = \{\frac{1}{n}, \frac{2}{n}, \ldots, 1\}$. Following the neighborhood regression approach in Section 2 we estimate the time-varying neighborhood $N(s)_t$ of all nodes $s \in V$ at all time points $t \in \mathcal{T}_n$ and thereby obtain the time-varying graph. As in Section 2.1.4 we estimate the neighborhood through the parameter vector θ of the ℓ_1-regularized regression problem in (4).

Without additional assumptions, however, the problem of estimating a time-varying graph is ill-posed as it would be impossible to aggregate observations even close in time because the model underlying any two adjacent observations might be completely different. We therefore have to assume a type of *local stationarity*: we either assume that there exists a partition $\mathcal{B}$ of $\mathcal{T}_n$ in whose parts $B \in \mathcal{B}$ the parameter vector θ_B is invariant. Or we assume that parameter vector θ is 'well behaved' as a function of time (for a formal definition in terms of derivatives of θ see Kolar *et al.* 2010). This means that if the time difference between any two adjacent time points a, b becomes smaller, then the corresponding parameter vectors $\theta_{t=a}, \theta_{t=b}$ differ less. Here we assume that the parameter vector is 'well behaved', that is, it is a smooth function of time.

To provide a local estimator of θ_t at each time point $t \in \mathcal{T}_n$, we apply a similar approach as Kolar *et al.* (2010), Kolar and Xing (2009) and Song *et al.* (2009) and use a kernel reweighted

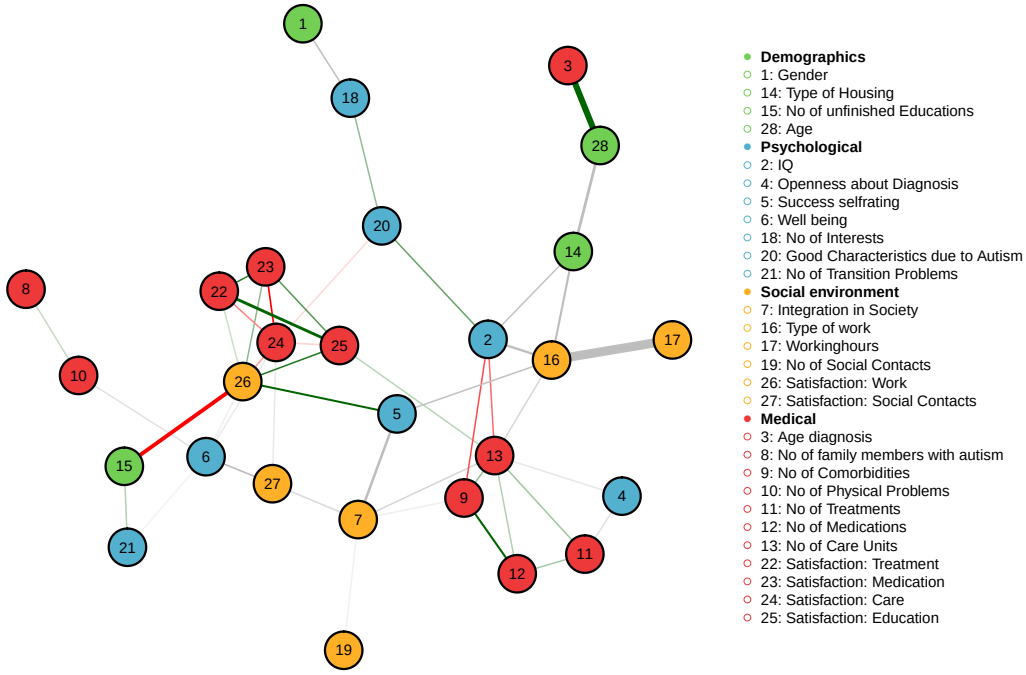


Figure 4: Undirected graphical model estimated from 28 variables capturing various aspects (demographics, psychological, social environment, medical) of the life of individuals diagnosed with Autism Spectrum Disorder.

neighborhood regression with the objective function

$$\min_{(\theta_0,\theta)\in\mathbb{R}^p} \left[\frac{1}{N} \sum_{i=1}^{N} w_{i;t}(y_{i;t} - \theta_{0;t} - X_{\setminus s;i}^T \theta_t)^2 + \lambda_n ||\theta_t||_1 \right]. \tag{11}$$

Note that (11) differs from (4) only in the multiplication by the weight $w_{i;t}$. In order to compute the weight $w_{i;t}$, we normalize the interval $[1, N]$ to $\mathcal{T}_n$. Then we define the weight $w_{i;t}$ as the density of a univariate Gaussian distribution with mean t and standard deviation b at i with a density normalized to $[0, 1]$

$$w_{i;t} = \frac{Z_i}{\max_{(i\in(-\infty,\infty))} (\cup_i Z_i)}, \quad \text{where} \quad Z_i = \frac{1}{\sqrt{2\pi}b} \exp\left\{ -\frac{(i-t)^2}{2b^2} \right\}. \tag{12}$$

We normalize the density to $[0, 1]$ because we can then easily compute the measure N_t of how much of the total data was used for estimating the parameter vector at a particular time point t by taking the sum over the weights $w_{i;t}$

$$N_t = \sum_{i\in\mathcal{T}} w_{i;t}. \tag{13}$$

In the absence of missing data, N_t is constant during the time interval except at time points close to $t = 0$ and $t = 1$ where we naturally use less data (see Figure 5). In the presence of missing data, N_t will vary depending on how many observations are available around time point t. Thus, N_t can be used to make relative statements about the sensitivity of the algorithm at different parts of the time series as a function of missing observations.

The kernel reweighting is illustrated in Figure 5. Here the graph is estimated at the time points $\{0, .25, .5, .75, 1\}$. At a given estimated time point t_e, the weights for all observations $i \in [0, 1]$ are defined by the Gaussian density with mean t_e and standard deviation b. This means that time points close to the estimated time point are associated with a relatively large weight, while observations distant from the estimated time step are associated with a relatively small weight. Indeed, as we use the Gaussian density, the weight decays exponentially with increasing distance from the estimated time point. Note that the relatively strong weighting of temporally close time steps reflects the assumption of local stationarity.

The bandwidth parameter b is an oracle parameter. In practice, however, we can come up with a reasonable guess by making assumptions about the functional form of the parameter vector in relation to the number of observed time points. With a large bandwidth b we can only detect slow changes in the graph. When choosing a smaller b we can detect changes within a smaller time interval (e.g. compare the bandwidth in Figure 5 left/right). A smaller bandwidth, however, comes at the cost of using fewer observations and therefore lower sensitivity. Selecting the optimal b would require the true graph which is what we intend to estimate. Therefore, we have to make assumptions about how the parameter vector varies as a function of time and choose b accordingly. The number of estimated time steps t_e can be made arbitrarily large. The only cost of choosing a large sequence of estimated time steps K is increased computational cost $\mathcal{O}(Kp2^d \log p)$.

The above procedure gives rise to the following algorithm:

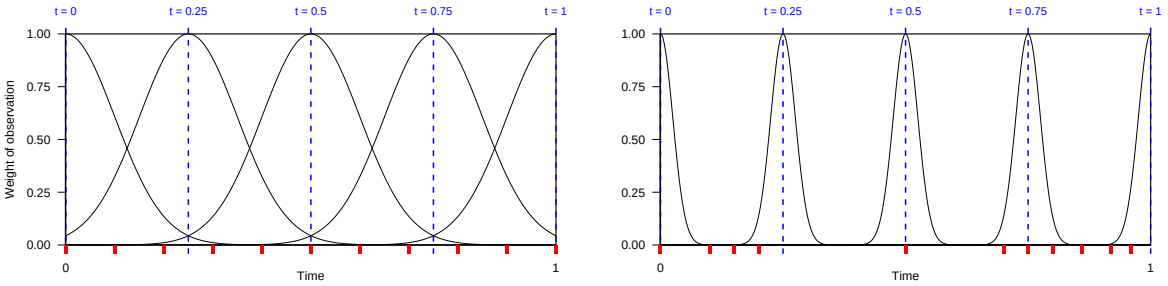


Figure 5: Illustration of the weighting of observations. At each estimated time points (blue vertical lines) we solve the weighted regression problem (11), in which the weights of each time step is proportional to the normal density with mean t and standard deviation $b =$. Red rectangles indicate the measurements in the time interval $[0, 1]$. Measurements can be equidistant in time (left) or distributed unregularly across time (right).

Algorithm 2 *(Nodewise estimation algorithm for time-varying graphs)*

Define a sequence of time points $T_e = \{t_1, \ldots, t_k\}, T_e \subset \mathcal{T}$ *at which the graphical model should be estimated;*

for *all* $t_e \in T_e$ **do**

  for *all* $s \in V$ **do**

    1. *Construct the prediction vector* X *including interactions corresponding to all required candidate neighborhoods*
    2. *Select a regularization parameter* λ_n *using the EBIC*
    3. *Solve the weighted* ℓ_1 *regularized regression problem in (11) and denote the solution by* $\hat{\theta}^t$.
    4. *Threshold the entries of* $\hat{\theta}^t$ *at* τ_n^t

  end

  5. *Combine parameters into weighted adjacency matrix using the AND- or OR-rule*

end

For time-varying graphs we select the regularization parameter λ_n with the EBIC as there is no straight-forward application of cross-validation to time-series.

3.2. Sampling from a time-varying pairwise Mixed Graphical Model

In this section we use the **mgm**-package to sample independent but not identically distributed samples from a time-varying graph. In the following section (3.3) we use Algorithm 2 to recover the true graph from these samples.

In the following example we specify a time-varying graph with 4 nodes and 2 edges. Edge $(2, 1)$ has the weight 1 at at $t = 0$ and decreases smoothly to 0 at $t = 0$. Conversely, edge $(4, 3)$ has the weight 0 at $t = 0$ and increases to 1. The exact parameter values as a function of time are depicted in Figure 6 (dashed lines). The specification of the time-varying graphical model is analogous to the specification of the stationary graphical model in `mgmsampler` in

Section 2.2. The only difference is that we specify a (possibly different) graphical model for each time point. This means that we specify the time-varying graph as a $p \times p \times n$-array and the thresholds as a $p \times n$-list, where n is the number of sampled time steps.

```
p <- 4 # 4 nodes
n <- 600 # number of time steps
graph <- matrix(0, p, p)
graphs <- array(dim=c(p, p, n))
k <- 15 # steepness of sigmoid function
sigm_in <- 1/ (1+ exp(-k*(seq(0,1,length=n)-.5))) # sigmoid increasing
sigm_de <- 1/ (1+ exp(k*(seq(0,1,length=n)-.5))) # sigmoid decreasing
graphs <- array(dim=c(p, p, n))
graphs[,,1:n] <- graph
graphs[1,2,] <- graphs[2,1,] <- - sigm_in
graphs[3,4,] <- graphs[4,3,] <- - sigm_de

# specify type and thresholds
type <- c('g', 'g', 'c', 'c') # two Gaussian, two Binary
lev <- c(1, 1, 2, 2)
# zero means (Gaussians) and thresholds (Binary)
thresh <- list(0, 0 , c(0, 0), c(0, 0))
# same means/thresholds at all time steps
threshs <- list(); for(nIter in 1:n) threshs[[nIter]] <- thresh
```

We can now call the `tv.mgmsampler` function, which uses a Gibbs sampler to sample one observation from each of the n specified graphical models:

```
set.seed(1)
data <- tv.mgmsampler(type, lev, graphs, threshs)
```

In the following section we use these data to recover the true time-varying graphical model.

3.3. Estimation of time-varying Mixed Graphical Models

We now use the function `tv.mgmfit` to recover the true time-varying graph from the sampled time series. We provide the function with the data matrix and a vector indicating the type of variables and the number of categories as above. We estimate the graphical model at 50 equidistant time points by specifying `tsteps = 50`. Here we select `gam = 0` and `d = 2` and set the bandwidth to $.8/n^{1/3}$.

```
# estimate
tv.obj <- tv.mgmfit(data = data,
                    type = type,
                    lev = lev,
                    tsteps = 50,
                    bandwidth = .8/n^(1/3),
                    gam = 0,
                    d = 2)
```

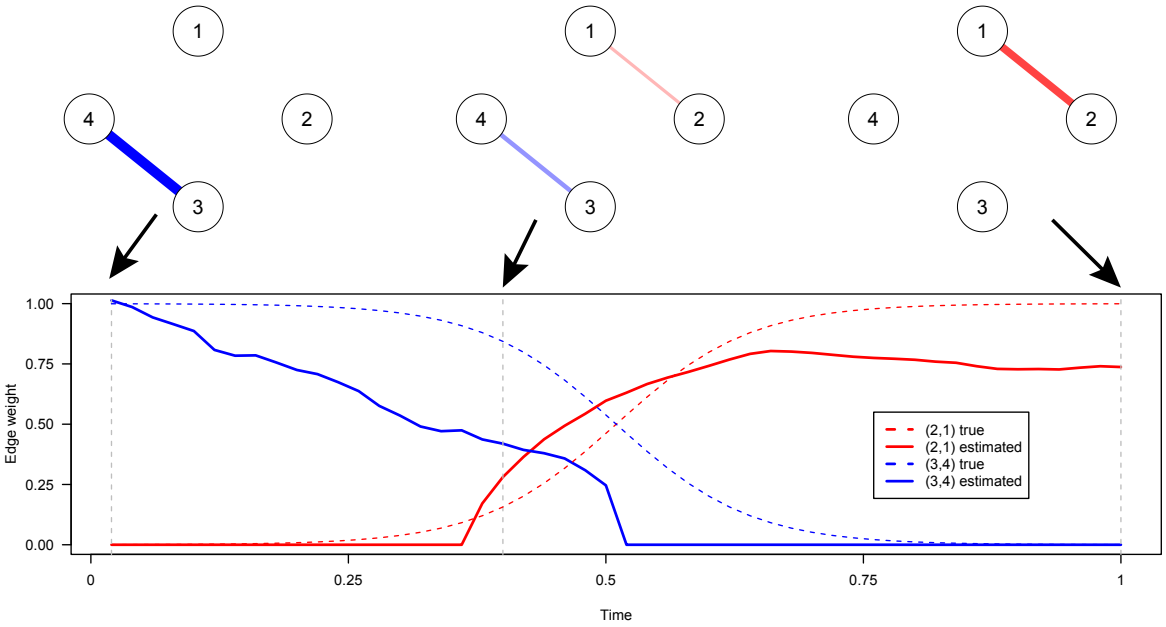


Figure 6: Lower panel: true (dashed) and estimated (solid) edge weights of the time-varying edges (2,1) and (3,4) of the example graph in Section 3.2. Vertical dashed grey lines indicate the time points at which we display the graphs in the upper panel: $\frac{1}{50}$, $\frac{20}{50}$ and $\frac{50}{50}$.

`tv.mgmfit` returns a $p \times p \times tsteps$ array `tv.ob$wadj` that contains the estimated weighted adjacency matrix at each estimated time point. Analogously to `tv.mgmfit`, `tv.ob$signs` is a $p \times p \times tsteps$ matrix containing information about the signs of edge weights for each estimated time point. The `tv.ob$mpar.matrix` contains all estimated parameters at each time point. In addition, `tv.ob$t.models` contains the output of `mgmfit` at each time point, which is used within `tv.mgmfit`. The vector `tv.ob$Nt` contains the effectively used sample size at each estimated time point calculated as in (13). This is interesting in the case of missing data as it indicates how sensitive the algorithm is at different parts of the time series.

In Figure 6 we plotted the estimated time-varying edge weights (solid lines) against the true time-varying edge weights (dashed lines). We see that the estimated edge weight $(2, 1)$ between the two Gaussian nodes more closely follow the true function. This is what we would expect as this edge weight is easier to estimate than the edge $(3, 4)$ between binary variables.

So far we assumed that the time series provided to `tv.mgmfit` is comprised by measurements that are equidistant in time. However, in many cases this might not be the case (see Figure 5 right panel). In a medical setting, for instance, patients might visit their physician at irregular times - or more generally, a measurement device has a varying sampling frequency. In these cases, assuming that all measurements are equidistant in time corrupts the true time scale and can heavily bias the estimated time-varying graph and lead to wrong conclusions. To avoid these problems, the user can provide a non-negative and strictly increasing sequence of time measurements via the argument `timepoints` to `tv.mgmfit`. The true time points are then used to define weights that keep the true time scale intact. Note that this is also a way to deal with missing measurements while maintaining the true time scale. At time points around

which the proportion of missing measurements is large, the algorithm uses less observation and hence the it is less sensitive. In case of equidistant measurements and missing values, one has also the option to set the argument to `missing = 'casewise.zw'`. This procedure sets the weights of measurements with missing values to zero. This is equivalent to providing the true time points and deleting measurements with missing values case wise.

3.4. Benchmarks

We report simulation results in order to illustrate the performance of Algorithm 2 in practical situations, showing performance for both smooth changes and jumps in the edge coefficients. We simulate data from a Binary-Gaussian mixed graphical model with 20 nodes, 10 of which are Gaussians. We generate edges from a random graph with $P_{edge} = .1$ which corresponds to an average 19 edges, which we keep constant across graphs and time points. We report the performance of Algorithm 2 in detecting edges in their change period averaged over 100 runs in Figure 7. We used the tuning parameters `gam = 0` and `d = 1` and combined neighborhood estimates using the AND-rule.

We first consider the case of a continuous change. We created five equidistant sub-intervals of $[0, 1]$. In the outer sub-intervals all edge-weights are constant. In each of the remaining three sub-intervals, 3 edges change smoothly to 0 and 3 edges change smoothly .8 (see top left panel in Figure 7). We only consider edges that change from 0 to .8 and the three sub-intervals with changes including the preceding and subsequent sub-interval in Figure 7 (for the edges changing from .8 to zero the results are the same in reversed time order). Thus, in the first third of the time interval presented in Figure 7 all true edges are zero and hence we only look at the precision of the algorithm. In the remainder of the time series all true edges are nonzero and hence we only look at sensitivity. We report the performance separately for the three different types of edges: edges between Gaussian nodes, between binary nodes and between Gaussian and binary nodes. Irrespective of the type of edge, precision is always high and sensitivity increases when moving further away from the change interval toward the part where the edge weights become constant. This is exactly what we would expect from the kernel weighted local estimator in (11): the farther we move the weight kernel through the time series, the more weight we put on edges that are nonzero. Learning the linear relationship between two Gaussian nodes is the easiest estimation problem and hence sensitivity is highest for these edges. The relationships between Gaussian and Binary variables are the hardest to learn as they are non-linear and hence we observe the lowest sensitivity for this edge-type.

In this simulation we used the bandwidth parameter $b = \frac{.9}{N^{1/3}}$. Across rows we vary the number of time steps $N = \{60, 300, 600\}$. N_t denotes the average number of effectively used time steps (see 13). Note that here N_t is an average as it is lower at the borders. N_t/p is the ratio of the average effectively used observations and the number of nodes in the graph and is a measure for the signal/noise ratio. Increasing the number of observations in the time interval increases the performance both in precision and sensitivity. Intuitively this makes sense because the more measurements there are in an interval, the more data we have around time point t_e to estimate G_{t_e}. A complementary perspective is that the time series with more measurements is 'stretched' and therefore more locally stationary and hence easier to estimate.

We now consider the case of a sudden change in parameter weights. The time series were generated similarly as in the continuous case, with the only difference that the changing

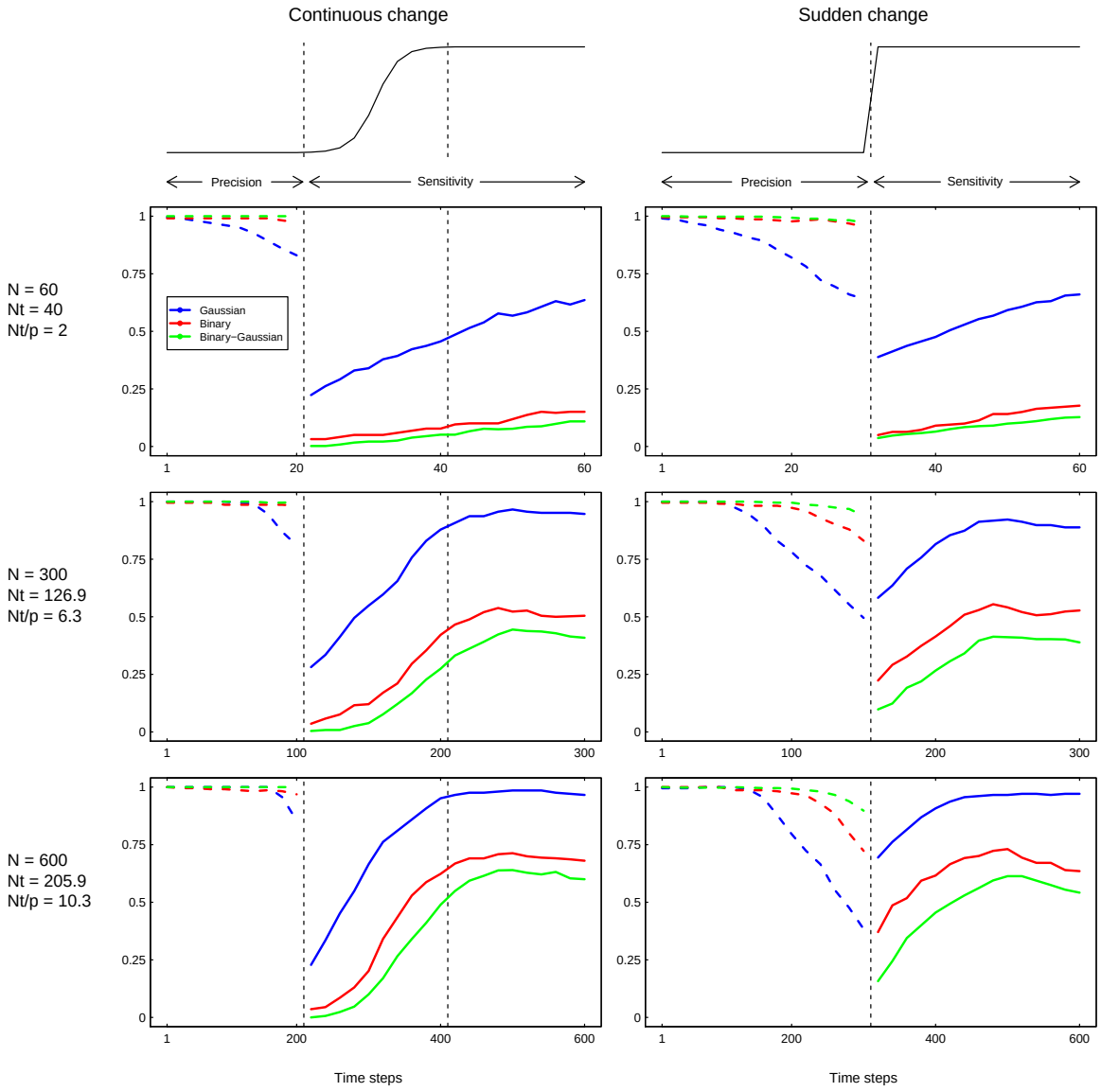


Figure 7: Performance of Algorithm 2 in estimating edges in the interval around a a continuous (left column) or a sudden (right column) change. The performance is reported separately for edges between Gaussians nodes (blue), between Binary nodes (red) and between Gaussian and binary nodes (green). N is the number of observations in the time interval around the change. N_t is the effectively used number of observations and is determined by the bandwidth b. N_t/p is a measure for ratio of signal and noise.

parameter weights transitioned from 0 to .8 as a step function instead of a sigmoid function (see top panel right in Figure 7). Analogously to the continuous case, in the first half of the presented time interval, all true edges are zero and hence we only present the precision of the algorithm. In the second half of the time interval, all true edges are nonzero and we hence look only at sensitivity.

Here we observe low precision around the change point. This is because the sudden change of edge-weights violates the assumption that edge-weights close in time are similar: at time points close to the change point, roughly half of the weighted observations are generated by a graphical model with edge-weight $\theta_{(s,t)} = 0$, while the other half of the observations are generated by a graphical model with edge weight $\theta_{(s,t)} = .8$. Clearly, Algorithm 2 will perform poorly at these time points close to the change point. If one assumes that the time-varying graph consists of piecewise constant partitions instead of continuous changes, one can use methods using the (group-) fused lasso, which are designed to learn such partitions (see for example Gibberd and Nelson 2014, 2015).

3.5. Application to Event Sampling Dataset in Psychopathology

In this section we use `tv.mgmfit` to re-analyze a time series that consists of responses to 43 questions asked up to 10 times a day via a smart phone application that has been partially analyzed in Wichers, Groot, Psychosystems, Group, and others (2016). The respondent is a patient suffering from depression and is undergoing a double-blinded reduction of the dose of administered antidepressant during a 34 week period (see Figure 8). The dots on the time line in Figure 8 indicate the begin and the end of the dose reduction. Next to the questions that are asked multiple times a day, the patient weekly filled in the SCL-90-R depression subscale (see panel at the center of Figure 8). We estimate the time-varying mixed graphical models at 20 equally spaced time steps, of which four are depicted in Figure 8. The last row of Figure 8 shows the total number of connections within each group of variables at each estimated time point.

While a detailed interpretation of this rich dataset goes far beyond the scope of the present paper, we see that the estimated graphical models differ considerably across time points. This highlights the importance of modeling graphical models as a function of time because it shows that estimating stationary graphs from time series data may be misleading. Furthermore, modeling the system of interest as it evolves over time provides additional insights about organizational processes, information diffusion, vulnerabilities and the potential impact of interventions.

3.6. Time-varying Mixed Autoregressive Model

In this section we use the `tv_var.mgm` function to estimate a time-varying VAR from a subset of the ESM data used in Section 3.5. Specifically, we consider the 9 variables related to the mood of the patient (red nodes in Figure 8). The items were phrased 'I feel relaxed', 'I feel irritated', etc. and the answer format ranged from 1 (not) to 7 (very).

Here we consider all variables to be jointly Gaussian and specify the type and levels of each variables accordingly via `type` and `lev`. We set the order of lags to 1 and estimate the time-varying graph at 50 equally spaced time points across the time series. We select the tuning parameter values `bandwidth = .05`, `gam = 0` and `d = 1`.

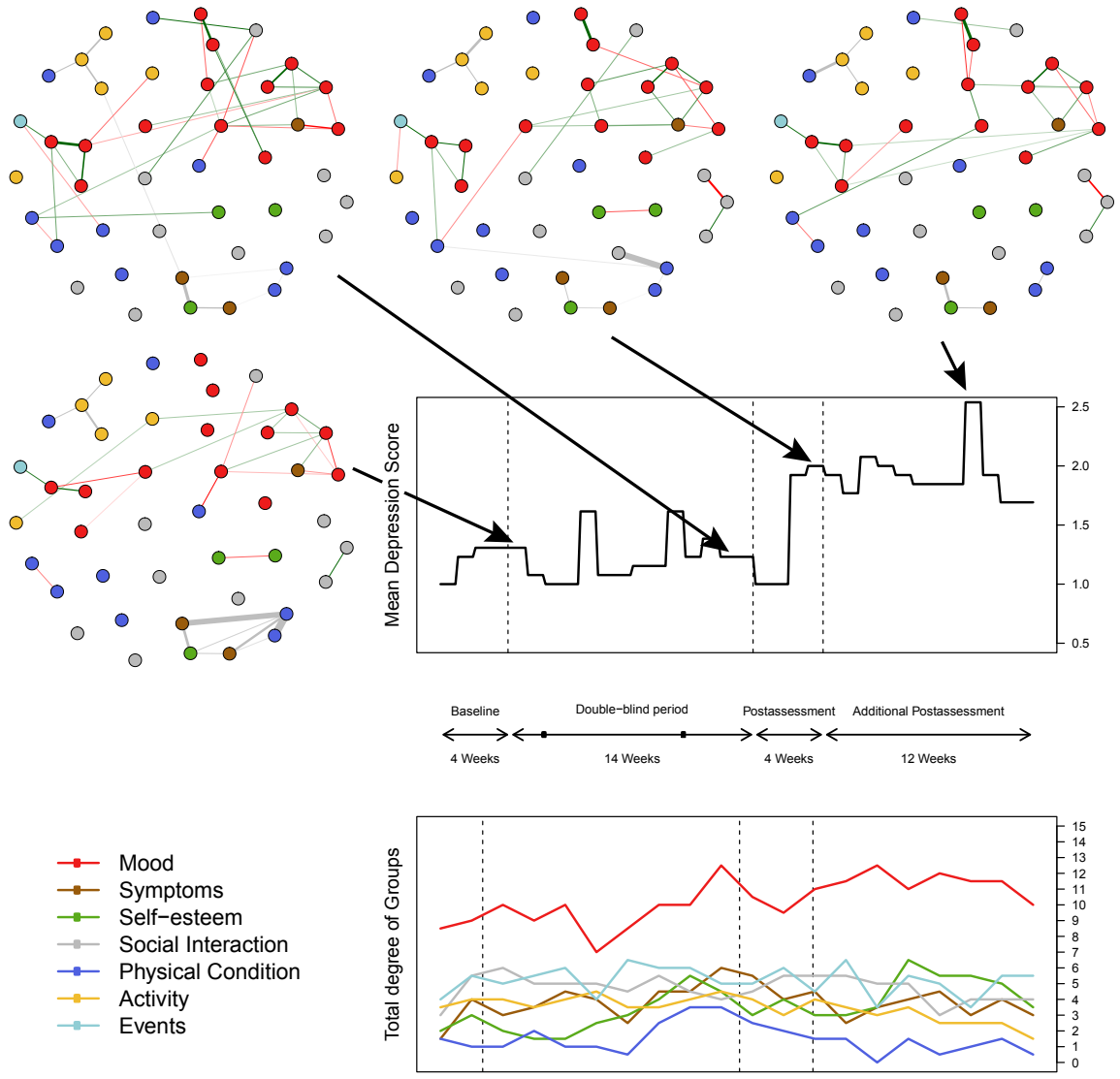


Figure 8: The time series consists of 43 variables from different domains (see legend) that are measured multiple times a day for 34 weeks. The time series consists of a base line period, a double blind period and two post assessment periods. The dots on the time line indicate the begin and the end of the dosis reduction. At the center of the figure we show the weekly measured average score of a depression scale. We depict four estimated graphical models at different time points. In the last row we show the absolute degree of the variable groups as a function of time.

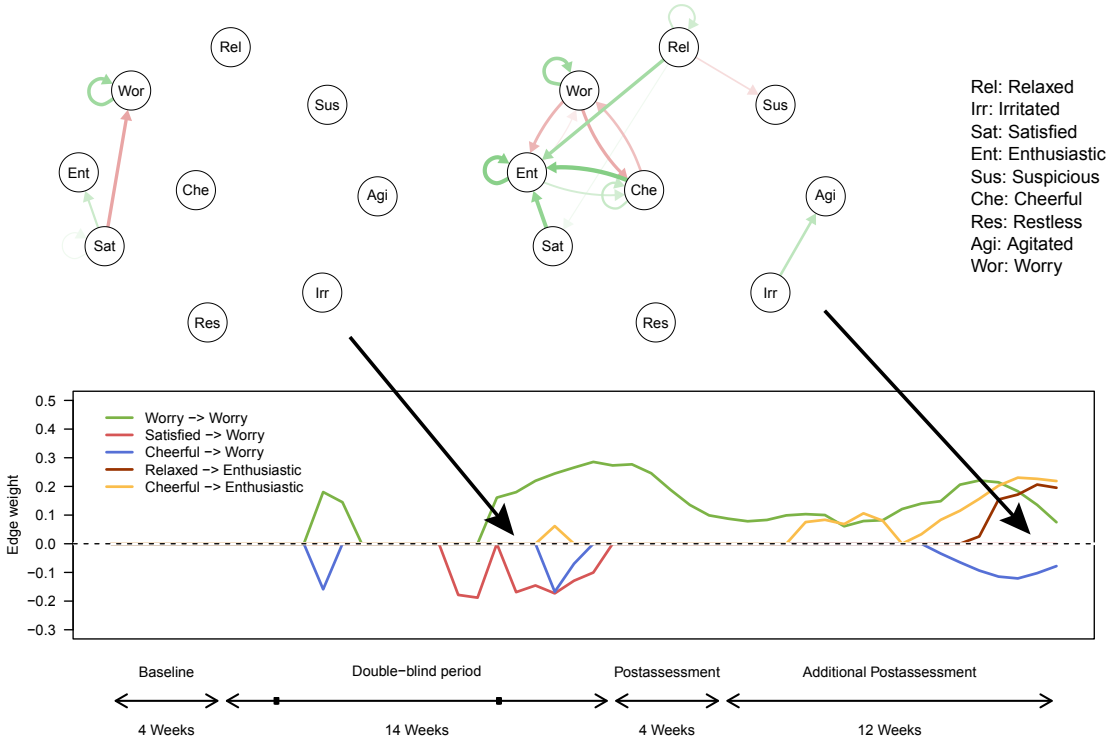


Figure 9: A time-varying vector autoregressive (VAR) model estimated on the mood related variables in the dataset from Section 3.5. The top panel shows the VAR model at time points 22 and 48. The lower panel shows the edge-weight for a selected number of edges over all estimated time points.

```
fit <- tv_var.mgm(data, type=rep('g', 9), lev=rep(1, 9),
                  lags = 1, tsteps = 50, bandwidth = .05,
                  gam = 0, d = 1)
```

Analogously to `tv.mgmfit`, `tv_var.mgm` returns a fit object `fit` that includes a weighted adjacency matrix `fit$wadj` and all estimated model parameters `fit$mpar.matrix` for each estimated time point. In addition, the information about the estimated model at each time point is saved in `fit$t.models`. Note that `fit$wadj` always contains the mean of the absolute value of all parameters describing the interaction between two nodes. However, as in the case of `tv.mgmfit`, we can retrieve information about the signs of edge weights from `fit$signs`. As we only considered continuous variables in this example, we have a sign defined for each edge weight. In addition, as in `tv.mgmfit`, all estimated parameters including signs are spored in `fit$mpar.matrix` for all time steps.

In Figure 9 we show the VAR model at time points 22 and 48. Green arrows indicate edge weights with positive sign, red arrows indicate edge weights with negative sign. In the lower panel, the edge-weight for six selected parameters is depicted across time. While a thorough interpretation goes beyond the scope of this paper, we see for instance that the autocorrelation of worry becomes nonzero at the beginning of the onset of the antidepressant reduction and

stays nonzero after the end of the antidepressant reduction. The sign of the edge weights make sense intuitively: for example, one would expect that feelings of satisfaction and worry are negatively related, and that feelings of cheerfulness and enthusiasm are rather positively related.

4. Concluding comments

Data sets including variables of mixed types are ubiquitous in a large array of disciplines. In the **mgm** package, we implement a method that both avoids transforms of non-Gaussian continuous variables and is able to incorporate (nominal) categorical variables.

In many situations the system under investigation evolves over time. In order to gain insights about the organizational processes, the diffusion of information, detecting vulnerabilities and the potential impact of interventions, one needs a model that can change across time. Thus, in addition to an estimation function for stationary mixed graphical models, we provide an estimation function for time-varying mixed graphical models, under the assumption that the edge coefficients are a smooth as a function of time.

In addition, we provide variations of the algorithms for the estimation of (time-varying) mixed graphical models for the estimation of (time-varying) mixed vector autoregressive (VAR) models. To our knowledge, there are currently no available implementations for the above mentioned methods.

References

Albert R, Barabasi AL (2002). "Statistical mechanics of complex networks." *Reviews of modern physics*, **74**(1), 47. URL http://journals.aps.org/rmp/abstract/10.1103/RevModPhys.74.47.

Banerjee O, El Ghaoui L, d'Aspremont A (2008). "Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data." *The Journal of Machine Learning Research*, **9**, 485–516. URL http://dl.acm.org/citation.cfm?id=1390696.

Begeer S, Wierda M, Venderbosch S (2013). "Allemaal Autisme, Allemaal Anders. Rapport NVA enquete 2013 [All Autism, All Different. Dutch Autism Society Survey 2013]." *Bilthoven: NVA.*, p. 83.

Borkulo Cv, Epskamp S, Robitzsch wcfA (2014). "IsingFit: Fitting Ising models using the eLasso method." URL https://cran.r-project.org/web/packages/IsingFit/index.html.

Borsboom D, Cramer AO (2013). "Network Analysis: An Integrative Approach to the Structure of Psychopathology." *Annual Review of Clinical Psychology*, **9**(1), 91–121. ISSN 1548-5943, 1548-5951. doi:10.1146/annurev-clinpsy-050212-185608. URL http://www.annualreviews.org/doi/abs/10.1146/annurev-clinpsy-050212-185608.

Bringmann LF, Vissers N, Wichers M, Geschwind N, Kuppens P, Peeters F, Borsboom D, Tuerlinckx F (2013). "A Network Approach to Psychopathology: New Insights into Clinical

Longitudinal Data." *PLoS ONE*, **8**(4), e60188. ISSN 1932-6203. doi:10.1371/journal.pone.0060188. URL http://dx.plos.org/10.1371/journal.pone.0060188.

Chen X, He Y (2015). "Inference of high-dimensional linear models with time-varying coefficients." *arXiv preprint arXiv:1506.03909*. URL http://arxiv.org/abs/1506.03909.

Dobra A, Lenkoski A (2011). "Copula Gaussian graphical models and their application to modeling functional disability data." *The Annals of Applied Statistics*, **5**(2A), 969–993. ISSN 1932-6157. doi:10.1214/10-AOAS397. URL http://projecteuclid.org/euclid.aoas/1310562213.

Epskamp S, Costantini G, Cramer AOJ, Waldorp LJ, Borsboom VDSaD (2015). "qgraph: Graph Plotting Methods, Psychometric Data Visualization and Graphical Model Estimation." URL https://cran.r-project.org/web/packages/qgraph/index.html.

Foygel R, Drton M (2011). "Bayesian model choice and information criteria in sparse generalized linear models." *arXiv preprint arXiv:1112.5635*. URL http://arxiv.org/abs/1112.5635.

Foygel R, Drton M (2014). "High-dimensional Ising model selection with Bayesian information criteria." *arXiv preprint arXiv:1403.3374*. URL http://arxiv.org/abs/1403.3374.

Friedman J, Hastie T, Tibshirani R (2008). "Sparse inverse covariance estimation with the graphical lasso." *Biostatistics*, **9**(3), 432–441. ISSN 1465-4644, 1468-4357. doi:10.1093/biostatistics/kxm045. URL http://biostatistics.oxfordjournals.org/cgi/doi/10.1093/biostatistics/kxm045.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of statistical software*, **33**(1), 1. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2929880/.

Friedman J, Tibshirani THaR (2014). "glasso: Graphical lasso- estimation of Gaussian graphical models." URL https://cran.r-project.org/web/packages/glasso/index.html.

Friedman N, Linial M, Nachman I, Pe'er D (2000). "Using Bayesian Networks to Analyze Expression Data." *Journal of Computational Biology*, **7**(3-4), 601–620. ISSN 1066-5277. doi:10.1089/106652700750050961. URL http://online.liebertpub.com/doi/abs/10.1089/106652700750050961.

Fruchterman TMJ, Reingold EM (1991). "Graph drawing by force-directed placement." *Software: Practice and Experience*, **21**(11), 1129–1164. ISSN 1097-024X. doi:10.1002/spe.4380211102. URL http://onlinelibrary.wiley.com/doi/10.1002/spe.4380211102/abstract.

Ghazalpour A, Doss S, Zhang B, Wang S, Plaisier C, Castellanos R, Brozell A, Schadt EE, Drake TA, Lusis AJ, Horvath S (2006). "Integrating Genetic and Network Analysis to Characterize Genes Related to Mouse Weight." *PLoS Genetics*, **2**(8), e130. ISSN 1553-7390, 1553-7404. doi:10.1371/journal.pgen.0020130. URL http://dx.plos.org/10.1371/journal.pgen.0020130.

Gibberd AJ, Nelson JD (2014). "High dimensional changepoint detection with a dynamic graphical Lasso." In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 2684–2688. IEEE. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6854087.

Gibberd AJ, Nelson JD (2015). "Regularized Estimation of Piecewise Constant Gaussian Graphical Models: The Group-Fused Graphical Lasso." *arXiv preprint arXiv:1512.06171*. URL http://arxiv.org/abs/1512.06171.

Hamilton JD (1994). *Time Series Analysis*. 1 edition edition. Princeton University Press, Princeton, N.J. ISBN 978-0-691-04289-3.

Haslbeck JMB, Waldorp LJ (2015). "Structure estimation for mixed graphical models in high-dimensional data." *arXiv:1510.05677 [math, stat]*. ArXiv: 1510.05677, URL http://arxiv.org/abs/1510.05677.

Huang S, Li J, Sun L, Ye J, Fleisher A, Wu T, Chen K, Reiman E (2010). "Learning brain connectivity of Alzheimer's disease by sparse inverse covariance estimation." *NeuroImage*, **50**(3), 935–949. ISSN 1053-8119. doi:10.1016/j.neuroimage.2009.12.120. URL http://www.sciencedirect.com/science/article/pii/S1053811909014281.

Kolar M, Song L, Ahmed A, Xing EP (2010). "Estimating time-varying networks." *The Annals of Applied Statistics*, **4**(1), 94–123. ISSN 1932-6157. doi:10.1214/09-AOAS308. URL http://projecteuclid.org/euclid.aoas/1273584449.

Kolar M, Xing EP (2009). "Sparsistent estimation of time-varying discrete Markov random fields." *arXiv preprint arXiv:0907.2337*. URL http://arxiv.org/abs/0907.2337.

Kolar M, Xing EP (2012). "Estimating networks with jumps." *Electronic Journal of Statistics*, **6**(0), 2069–2106. ISSN 1935-7524. doi:10.1214/12-EJS739. URL http://projecteuclid.org/euclid.ejs/1351865118.

Kuppens P, Allen NB, Sheeber LB (2010). "Emotional Inertia and Psychological Maladjustment." *Psychological Science*, **21**(7), 984–991. ISSN 0956-7976, 1467-9280. doi:10.1177/0956797610372634. URL http://pss.sagepub.com/lookup/doi/10.1177/0956797610372634.

Lauritzen SL (1996). *Graphical models*. Number 17 in Oxford statistical science series. Clarendon Press ; Oxford University Press, Oxford, New York. ISBN 0-19-852219-3.

Liu H, Lafferty J, Wasserman L (2009). "The nonparanormal: Semiparametric estimation of high dimensional undirected graphs." *The Journal of Machine Learning Research*, **10**, 2295–2328. URL http://dl.acm.org/citation.cfm?id=1755863.

Loh PL, Wainwright MJ (2013). "Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses." *The Annals of Statistics*, **41**(6), 3022–3049. URL http://projecteuclid.org/euclid.aos/1388545677.

Meinshausen N, Bühlmann P (2006). "High-dimensional graphs and variable selection with the Lasso." *The Annals of Statistics*, **34**(3), 1436–1462. ISSN 0090-5364. doi:10.1214/009053606000000281. URL http://projecteuclid.org/Dienst/getRecord?id=euclid.aos/1152540754/.

Monti RP, Hellyer P, Sharp D, Leech R, Anagnostopoulos C, Montana G (2014). "Estimating time-varying brain connectivity networks from functional MRI time series." *Neuroimage*, **103**, 427–443. URL http://www.sciencedirect.com/science/article/pii/S1053811914006168.

Pfaff B (2008). *Analysis of Integrated and Cointegrated Time Series with R*. 2nd edition edition. Springer, New York. ISBN 978-0-387-75966-1.

Ravikumar P, Wainwright MJ, Lafferty JD (2010). "High-dimensional Ising model selection using âĎŞ 1 -regularized logistic regression." *The Annals of Statistics*, **38**(3), 1287–1319. ISSN 0090-5364. doi:10.1214/09-AOS691. URL http://projecteuclid.org/euclid.aos/1268056617.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **71**(2), 319–392. ISSN 1467-9868. doi:10.1111/j.1467-9868.2008.00700.x. URL http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9868.2008.00700.x/abstract.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464. ISSN 0090-5364, 2168-8966. doi:10.1214/aos/1176344136. URL http://projecteuclid.org/euclid.aos/1176344136.

Song L, Kolar M, Xing EP (2009). "KELLER: estimating time-varying interactions between genes." *Bioinformatics*, **25**(12), i128–i136. ISSN 1367-4803, 1460-2059. doi:10.1093/bioinformatics/btp192. URL http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btp192.

Tao Q, Huang X, Wang S, Xi X, Li L (2016). "Multiple Gaussian Graphical Estimation with Jointly Sparse Penalty." *Signal Processing*. ISSN 01651684. doi:10.1016/j.sigpro.2016.03.009. URL http://linkinghub.elsevier.com/retrieve/pii/S0165168416300032.

Toda HY, Yamamoto T (1995). "Statistical inference in vector autoregressions with possibly integrated processes." *Journal of Econometrics*, **66**(1âĂŞ2), 225–250. ISSN 0304-4076. doi:10.1016/0304-4076(94)01616-8. URL http://www.sciencedirect.com/science/article/pii/0304407694016168.

van Borkulo CD, Borsboom D, Epskamp S, Blanken TF, Boschloo L, Schoevers RA, Waldorp LJ (2014). "A new method for constructing networks from binary data." *Scientific Reports*, **4**. ISSN 2045-2322. doi:10.1038/srep05918. URL http://www.nature.com/doifinder/10.1038/srep05918.

Wainwright MJ, Jordan MI (2008). "Graphical Models, Exponential Families, and Variational Inference." *Foundations and Trends in Machine Learning*, **1**(1âĂŞ2), 1–305. ISSN 1935-8237, 1935-8245. doi:10.1561/2200000001. URL http://www.nowpublishers.com/product.aspx?product=MAL&doi=2200000001.

Wichers M, Groot PC, Psychosystems ESM, Group EWS, others (2016). "Critical Slowing Down as a Personalized Early Warning Signal for Depression." *Psychotherapy and psychosomatics*, **85**(2), 114–116. URL http://www.karger.com/Article/FullText/441458.

Yang E, Baker Y, Ravikumar P, Allen G, Liu Z (2014). "Mixed Graphical Models via Exponential Families." In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pp. 1042–1050. URL http://jmlr.org/proceedings/papers/v33/yang14a.pdf.

Yang E, Ravikumar P, Allen GI, Liu Z (2013). "On graphical models via univariate exponential family distributions." *arXiv preprint arXiv:1301.4183*. URL http://arxiv.org/abs/1301.4183.

Zhao T, Li X, Liu H, Roeder K, Lafferty J, Wasserman L (2015). "huge: High-Dimensional Undirected Graph Estimation." URL https://cran.r-project.org/web/packages/huge/index.html.

Zhou S, Lafferty J, Wasserman L (2010). "Time varying undirected graphs." *Machine Learning*, **80**(2-3), 295–319. ISSN 0885-6125, 1573-0565. doi:10.1007/s10994-010-5180-0. URL http://link.springer.com/10.1007/s10994-010-5180-0.

Affiliation:

Jonas Haslbeck
Psychological Methods
Nieuwe Achtergracht 129-B
Postbus 15906
1018 WT, Amsterdam
The Netherlands
E-Mail: jonashaslbeck@gmail.com
Website: http://jmbh.github.io

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd