Non-stationary sparse time series chain graphical models for reconstructing networks

Bachelor Project Mathematics

June 2015

Student: R.P.W. van Ommeren

First supervisor: Prof.dr. E.C. Wit

Second supervisor: D. Valesin, PhD

Abstract

This paper consists of 2 main parts. In the first part the theory behind the stationary sparse time series chain graphical model (STSCGM) for reconstructing multivariable networks is explained at length. This model is parametrized by a precision matrix and an autoregressive coefficient matrix. By using penalized likelihood and the SCAD penalty the underlying relationships between the variables is explored. The second part consists of two non-stationary alterations of this model. In the first model we look at multivariable data where drastic changes in the autoregressive matrix could occur at different change points, leaving the precision matrix untouched. In the latter, we look at multivariable data where the autoregressive matrix slowly changes over time, leaving the precision matrix untouched again. Combining both non-stationary models gives us a good method to analyze data where changes in the autoregressive coefficient matrix occur at unknown points in time. In the end we will look at the performance of these 3 models using simulated data.

Contents

1 Stationary sparse time series chain graphical models

1.1 Non-sparse time series graphical chain models

Following Dahlhaus and Eichler (2003), we define a time series chain graph as follows.

Definition 1.1. (Time series chain graph) The time series chain graph of a stationary process X over time T is the chain graph $G = (V, E)$ with $V = V_0 \cup \ldots \cup V_T$ and edge set E such that

$$(a, t-u) \rightarrow (b, t) \notin E \Leftrightarrow u \leq 0 \text{ or } X_a(t-u) \perp\!\!\!\perp X_b(t)$$
$$(a, t-u) \leftrightarrow (b, t) \notin E \Leftrightarrow u \neq 0 \text{ or } X_a(t) \perp\!\!\!\perp X_b(t)$$

Here $\perp\!\!\!\perp$ means *conditionally independent of.* From the stationarity assumption, we have that for undirected edges

$$(a, t) \leftrightarrow (b, t) \notin E \Leftrightarrow (a, s) \leftrightarrow (b, s) \notin E \quad \forall s \in \{1, \ldots, T\}$$

The same shift-invariance holds for the directed edges. Plus, a directed edge can only occur from one time block V_t to another time block V_u if $t < u$. So X_t cannot be influenced by a state further in time. An undirected edge can only occur in the same time block. Such a block can be seen as a Gaussian graphical model, where no edge between two variables is equivalent to conditional independence.

For simplicity, we will only consider the case where $X_a(t-u) \perp\!\!\!\perp X_b(t)$ for $u > 1$. That is, the state of X_t is conditionally independent of the states $X_0, \ldots, X_{t-2}$. Then we can rewrite the joint probability density function of $X_0, \ldots, X_T$ by using the first-order Markovian property as

$$f(X_0, \ldots, X_T) = f(X_0) f(X_1 \mid X_0) \ldots f(X_T \mid X_{T-1})$$

Since the time series chain graph is stationary, $f(X_t \mid X_{t-1})$ is the same for all t. Now assume that we can approximate this conditional distribution by a multivariate normal distribution of the form

$$X_t | X_{t-1} \sim N(\Gamma X_{t-1}, \Sigma) \tag{1}$$

for some matrix Γ and Σ. The non-zero elements γ_{ij} of Γ represent a directed edge between two successive time blocks. To understand the meaning of Σ, let $\Theta = \Sigma^{-1}$. Then by Whittaker (2008, Chapter 5), the non-zero elements θ_{ij} of Θ represent undirected edges between vertices in the same time block, like in the Gaussian graphical model.

Let the conditional pdf of $X_t|X_{t-1}$ be

$$f(X_t|X_{t-1}) = (2\pi)^{-p/2}\det(\Sigma)^{-1/2}\exp\{-\frac{1}{2}(X_t - \Gamma X_{t-1})'\Sigma^{-1}(X_t - \Gamma X_{t-1})\}.$$

The log-likelihood ℓ is proportional to

$$\ell(\Gamma, \Theta) \propto \log\det(\Theta) - \mathrm{tr}(S_\Gamma \Theta) + c, \tag{2}$$

see Appendix A. Here S_Γ is the maximum likelihood estimator of the covariance matrix given by

$$\begin{aligned} S_\Gamma &= \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=1}^{T}(X_t - \Gamma X_{t-1})(X_t - \Gamma X_{t-1})' \\ &= S_y - S_{yx}\Gamma' - \Gamma S_{yx}' + \Gamma S_x \Gamma' \end{aligned} \tag{3}$$

where we have defined the following matrices:

$$S_y = \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=1}^{T} X_t X_t'. \quad S_{yx} = \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=1}^{T} X_t X_{t-1}'$$

$$S_x = \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=1}^{T} X_{t-1} X_{t-1}'$$

1.2 Sparse time series chain graphical models

For a time series chain graphical model with a first-order Markov property the number of parameters in the model grows exponentially with the number of variables. To improve our estimates we could introduce a penalty function for the elements of Γ and Θ. We want to find a penalty function such that our estimate has the following properties:

- Unbiasedness. When the true parameter is large, the estimate should be close to unbiased to avoid excessive estimation bias.
- Sparsity. Small coefficients should be estimated to zero to reduce model complexity.
- Continuity. The estimator should be continuous in the data to avoid instability in model prediction.

We define two penalty function for the elements of Γ and Θ, the lasso penalty and the SCAD penalty.

1.2.1 Lasso penalty

The least shrinkage and selection operator (lasso) proposed in Tibshirani (1996) is an L^1 penalty. In linear regression the estimator b_{lasso} is found from solving

$$b_{lasso} = \arg\min_{\beta}\Big\{\big(\sum_{i=1}^{n}(y_i - x_i\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\Big\}.$$

Thus the penalty function for each parameter in the model is $P_\lambda(x) = \lambda|x|$. The lasso problem is equivalent to solving

$$b_{lasso} = \arg\min_\beta \Big\{ \big(\sum_{i=1}^{n} (y_i - x_i\beta \big)^2 \Big\}$$

$$\text{such that } \sum_{j=1}^{p} |\beta_j| \leq t,$$

for some t, where there exists a one-to-one relationship between λ and t. To see that this leads to sparsity in the model, consider the case where $p = 2$ in Figure 1.

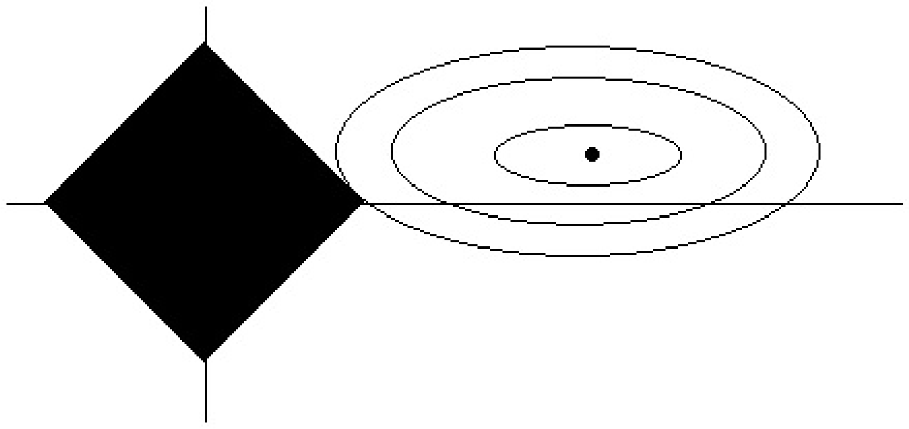


Figure 1: Geometric interpretation of the lasso with 2 parameters. The black dot is the OLS estimator, the elliptical lines are the level curves of the squared residuals. The probability that small coefficients are estimated zero increases by introducing the L^1 penalty.

The lasso uses the good features of both subset selection and ridge regression. Like in ridge regression, the lasso shrinks coefficients to reduce prediction error. OLS estimates tend to have small bias but large variance. Prediction accuracy can sometimes be improved by shrinking coefficients, sacrificing a little bias. Like in subset selection it sets other coefficients to zero leading to an understandable model.

1.2.2 SCAD penalty

The lasso estimate satisfies the last two criteria of our desired estimate. However, it will shrink large coefficients and this will lead to biasedness when the true parameter is large. The SCAD penalty proposed in Fan and Li (2001) is an alteration of the lasso penalty. Its penalty function is given by the derivative

$$P'_\lambda(x) = \lambda\Big\{I\{x \leq \lambda\} + \frac{(a\lambda - x)_+}{(a-1)\lambda} I\{x > \lambda\}\Big\},$$

for some $\lambda > 0$ and some $a > 2$. This defines a quadratic spline with knots at λ and $a\lambda$. From Figure 2 it can be seen that the SCAD penalty is the same as the lasso penalty for small values, so the SCAD penalty will set smaller coefficients to zero. However, the penalty for bigger coefficients is much smaller than the lasso penalty. This means that the SCAD estimate will not shrink large coefficients as much as the lasso does. Optimal values of (λ, a) can be obtained by cross-validation or other model selection criteria, but a value of $a = 3.7$ is recommended by Fan and Li (2001).

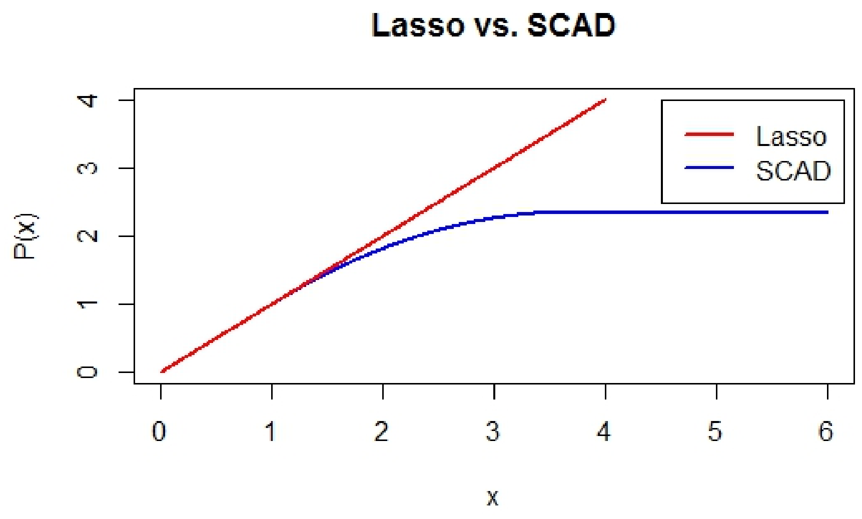


Figure 2: Penalty functions of the lasso and SCAD penalty, with $\lambda = 1$ and $a = 3.7$

1.2.3 Penalized likelihood

Consider the log-likelihood given by (2). We can now define two penalty functions P_λ and P_ρ corresponding to elements of the matrices Γ and Θ respectively. The penalized log-likelihood for the sparse time series chain graphical model (STSCGM) is

$$\ell_{pen}(\Gamma, \Theta) = \log\det(\Theta) - \mathrm{tr}(S_\Gamma \Theta) - \sum_{i \neq j}^{p} P_\lambda(|\theta_{ij}|) - \sum_{i,j}^{p} P_\rho(|\gamma_{ij}|). \quad (4)$$

Solving (4) is very hard when the penalty function is a concave function like the SCAD penalty. To solve it we use the local linear approximation (LLA) proposed by Zou and Li (2008). The first-order Taylor approximation in a neighbourhood of $|x|$ is given by

$$P_\lambda(|x|) \approx P_\lambda(|x_0|) + P'_\lambda(|x_0|)(|x| - |x_0|)$$

In the following section we will explain the algorithm for solving (4).

1.3 Solving the STSCGM

1.3.1 Step 1: Initial estimates

We start with an initial estimate for the transition matrix Γ. This is done by QR-decomposition. We then define the two local linear approximations in the neighbourhood of $|\gamma|$ and $|\theta|$,

$$P_\lambda(|\theta|) \approx P_\lambda(|\theta_0|) + P'_\lambda(|\theta_0|)(|\theta| - |\theta_0|), \tag{5}$$

$$P_\rho(|\gamma|) \approx P_\rho(|\gamma_0|) + P'_\rho(|\gamma_0|)(|\gamma| - |\gamma_0|). \tag{6}$$

Because we will differentiate the penalized likelihood with respect to θ or λ, we will be left with the term depending on the first derivative of the penalty function.

1.3.2 Step 2: Solving for Θ

First we update the estimate of Θ. Solving for Θ with Γ fixed gives the optimization problem

$$\arg\max_{\Theta} \Big\{ \log\det(\Theta) - \operatorname{tr}(S_\Gamma\Theta) + \sum_{i\neq j}^{p} P_\lambda(|\theta_{ij}|) \Big\} \tag{7}$$

Let W be the matrix whose elements are the penalties for θ_{ij}. That is, $w_{ij} = P_\lambda(|\theta_{ij}|)$. We will not penalize the diagonals of Θ, so $w_{ii} = 0$. The reason we do this is because we assume that the variance of each variable is not equal to zero. Plus, this assumption will be useful in step 3. Given our estimate $\hat{\Theta}^{(k)}$, we could calculate W. However, we could find a slightly better estimate $\hat{\Theta}^{(k+1)}_{lasso}$ using the lasso penalty in (7). So we first solve

$$\hat{\Theta}^{(k+1)}_{lasso} = \arg\max_{\Theta} \Big\{ \log\det(\Theta) - \operatorname{tr}(S_\Gamma\Theta) + \lambda \sum_{i\neq j}^{p} |\theta_{ij}| \Big\}$$

The graphical lasso algorithm from Friedman et al. (2007) solves this optimizition problem very fast. After obtaining the better estimate for Θ, we calculate the penalty matrix W using (5). Then we find $\Theta^{(k+1)}$ by

$$\hat{\Theta}^{(k+1)} = \arg\max_{\Theta} \Big\{ \log\det(\Theta) - \operatorname{tr}(S_{\Gamma^{(k)}}\Theta) + \sum_{i\neq j}^{p} w_{ij}(|\theta_{ij}|) \Big\}.$$

Here we also use the graphical lasso algorithm.

1.3.3 Step 3: Solving for Γ

We start by updating the penalty matrix P using (6). An element ρ_{ij} of P is the penalty for γ_{ij}. Then, for Θ fixed, we let

$$\begin{aligned}\hat{\Gamma}^{(k+1)} &= \arg\max_{\Gamma} \{\ell(\Theta,\Gamma) - \sum_{i,j}^{p} \rho_{ij}(|\gamma_{ij}|)\} \\ &= \arg\max_{\Gamma} \{ - \operatorname{tr}(S_{\Gamma}\Theta) - \sum_{i,j}^{p} \rho_{ij}(|\gamma_{ij}|)\} \\ &= \arg\max_{\Gamma} \Big\{ \operatorname{tr}(S_{xy}\Gamma'\Theta + \Gamma S_{xy}'\Theta - \Gamma S_x \Gamma'\Theta) - \sum_{i,j}^{p} \rho_{ij}|\gamma_{ij}| \Big\}\end{aligned}$$

We solve this optimization problem by using a coordinate descent algorithm. While keeping all other elements of Γ fixed, we let

$$\begin{aligned}\hat{\gamma}_{ij}^{(k+1)} &= \arg\max_{\gamma_{ij}} \Big\{ \operatorname{tr}(S_{xy}\Gamma'\Theta + \Gamma S_{xy}'\Theta - \Gamma S_x \Gamma'\Theta) - \rho_{ij}|\gamma_{ij}| \Big\} \\ &= \arg\max_{\gamma_{ij}} \Big\{ g(\gamma_{ij}) - \rho_{ij}|\gamma_{ij}| \Big\} \qquad (8)\end{aligned}$$

Since the second derivative of g is a negative constant, see Appendix A.1, we have that g is strictly concave. Subtracting the penalty term from this function sets some elements of Γ to zero, see Figure 3.

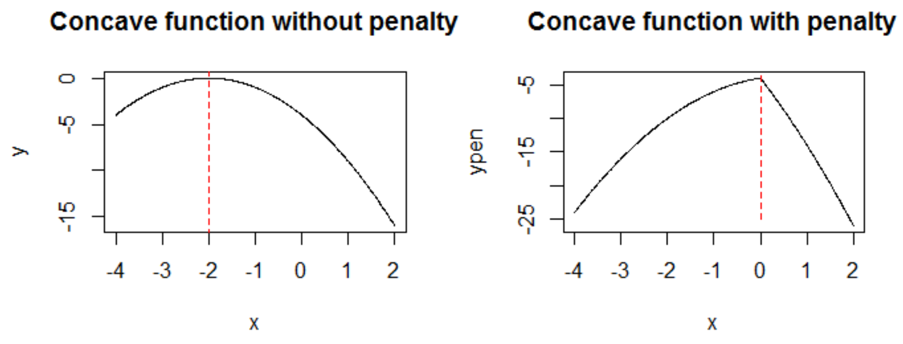


Figure 3: Graph of $f(x) = -(x+2)^2$ [left] and $f(x) - 5|x|$ [right]

It takes some effort to see that $\hat{\gamma}_{ij}^{(k+1)} = 0$ if and only if $|g'(0)| \leq \rho_{ij}$. Also, adding a L^1 penalty does not change the sign of the estimate in a single updating step. That is, the penalty can only shrink the estimate to zero or set it zero, see Appendix A.1.

Differentiating the expression in (8) with respect to γ_{ij} yields a linear function with discontinuity at $\gamma_{ij} = 0$. Setting this score function equal to zero yields

$$\frac{\partial \ell_{pen}}{\partial \gamma_{ij}} = 2e_i'(\Theta S_{xy})e_j - 2e_i'(\Theta\Gamma S_x)e_j - \mathrm{sgn}(\gamma_{ij})\rho_{ij} = 0. \tag{9}$$

This equation has no solution if and only if $|2e_i'(\Theta S_{xy})e_j - 2e_i'(\Theta\Gamma S_x)e_j| < \rho_{ij}$, evaluated at $\gamma_{ij} = 0$. But this is exactly when $|g'(0)| < \rho_{ij}$, and therefore also when $\hat{\gamma}_{ij}^{(k+1)} = 0$.

Now that we have dealt with the case when the estimate is set to zero,suppose the estimate should not be set to zero. Then (9) has a solution and it is easy to solve (8) without the penalty term. Although we don't have an exact expression, we can use the Newton-Raphson method to find the exact zero point.

$$\hat{\gamma}_{ij}^{(reg)} = \hat{\gamma}_{ij}^{(k)} - \frac{g'(\hat{\gamma}_{ij}^{(k)})}{g''(\hat{\gamma}_{ij}^{(k)})}$$

Since adding the L^1 penalty does not change the sign of the estimate, we now know the sign of the estimate $\hat{\gamma}_{ij}^{(k+1)}$. Then it is easy to solve (9), using the Newton-Raphson method again. The updating formula for the estimate is

$$\hat{\gamma}_{ij}^{(k+1)} = \hat{\gamma}_{ij}^{(reg)} - \frac{g'(\hat{\gamma}_{ij}^{(reg)}) - \rho_{ij}\ \mathrm{sgn}(\hat{\gamma}_{ij}^{(reg)})}{g''(\hat{\gamma}_{ij}^{(reg)})}.$$

This coordinate descent approach is repeated until $\hat{\Gamma}$ converges. Then the updating steps 2 and 3 are repeated until both $\hat{\Theta}$ and $\hat{\Gamma}$ converge.

1.4 Model Selection

Our final estimates of Γ and Θ are determined by the tuning parameters λ and ρ. Optimal values can be obtained by various model selection criteria, like cross-validation, the Akaike Information Criterion (AIC) or the Bayesian Information Criterion. Like in Abegaz and Wit (2013) we will use the BIC. The BIC is defined as

$$\mathrm{BIC}(\lambda,\rho) = -nT\big[\log\det(\hat{\Theta}_\lambda) - \mathrm{tr}(S_{\hat{\Gamma_\rho}})\big] + \log(nT)(\frac{a}{2} + b + p)$$

Here p is the number of non-zero diagonal entries of $\hat{\Theta_\lambda}$, $\frac{a}{2}$ is the number of off-diagonal non-zero elements of $\hat{\Theta_\lambda}$ divided by two because of symmetry and b is the number of non-zero elements of $\hat{\Gamma_\rho}$ The best pair (λ,ρ) is the one minimizing the BIC. Minimizing the BIC cannot be done in a analytic way. Therefore, a grid-search (or parameter sweep) seems the best option. This is simply an exhausting searching through a manually specified finite subset of reasonable values.

This BIC is incorrect in the sense that the degrees of freedom of the model is unequal to the number of parameters for small amounts of observations. However, it can be shown that the difference of the two is small and, given a model A, we have that,

$$P\{df(A) = p(A)\} \to 1 \text{ as } nT \to \infty,$$

see Zhang et al. (2010). Here $p(A)$ equals the number of parameters of A.

2 Non-stationary sparse time series graphical chain models

In the previous model the transition matrix Γ was the same for all T. That is, the STSCGM is a stationary process. However, so-called shocks can change a multivariate time series like stock markets drastically. Think of events like the burst of the Internet bubble in 2000, the terrorist attacks of September 11th in 2001 or the bankruptcy of the Lehman Brothers in 2008, marking the start of the financial crisis. Thus when analyzing these multivariate time series, we have to take into account that at a sudden point, the model could change over time. We consider two non-stationary alterations of the sgtscm.

2.1 Sparse time series graphical chain models with change points

The first alteration we will look at is the following. Assume again that we have n replicates of T time points of p variables. Assume that we know that the transition matrix Γ could change completely at points $T_1, \dots, T_{K-1}$. Then we have that conditional the pdf for $X_1, \dots, X_{T_1}$ is influenced by Γ_1, the conditional pdf for $X_{T_1+1}, \dots, X_{T_2}$ is influenced by Γ_2 and so on. Thus, letting $T_0 = 0$ and $T_K = T$, we have

$$X_t|X_{t-1} \sim N(\Gamma_k X_{t-1}, \Sigma) \text{ for } T_{k-1} + 1 \leq t \leq T_k.$$

The log-likelihood of the conditional pdf is then

$$\ell(\Gamma, \Theta) \propto \log\det(\Theta) - \operatorname{tr}\Big[\sum_{k=1}^{K}(S_{\Gamma_k}\Theta)\Big] + c,$$

see Appendix A.2. The S_k matrices are defined as follows

$$\begin{aligned} S_{\Gamma_k} &= \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=T_{k-1}+1}^{T_k}(X_t - \Gamma_k X_{t-1})(X_t - \Gamma_k X_{t-1})' \\ &= S_{y_k} - S_{yx_k}\Gamma_k' - \Gamma_k S_{yx_k}' + \Gamma_k S_{x_k}\Gamma_k' \end{aligned}$$

where we have that

$$S_{y_k} = \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=T_{k-1}+1}^{T_k} X_t X_t', \quad S_{yx_k} = \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=T_{k-1}+1}^{T_k} X_t X_{t-1}'$$

$$S_{x_k} = \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=T_{k-1}+1}^{T_k} X_{t-1} X_{t-1}'$$

To add sparsity to the model we penalize the likelihood. Because we want to obtain sparse estimates for Θ and $\Gamma_1, \dots, \Gamma_K$, we penalize all elements of all $K + 1$ matrices. Assigning a penalty parameter to each transition matrix is possible, but finding the optimal combination op parameters by a grid search will be a very extensive search. For computational advantages, we choose one penalty parameter for Θ and one for all the Γ_k matrices. This results in the following penalized likelihood.

$$\begin{aligned}\ell_{pen}(\Gamma_1, \ldots, \Gamma_K, \Theta) = \log\det(\Theta) - \operatorname{tr}\Big[\sum_{k=1}^{K}(S_{\Gamma_k}\Theta\Big] - \sum_{i \neq j}^{p} P_\lambda(|\theta_{ij}|) \\ - \sum_{k=1}^{K}\sum_{i,j}^{p} P_\rho(|\gamma_{k_{ij}}|). \qquad (10)\end{aligned}$$

We will use SCAD penalty function, approximated by the LLA. We create K subsets of the data, each subset belonging to the time interval which is affected by Γ_k plus its initial state. Then we can find initial estimates for $\Gamma_1, \ldots, \Gamma_K$ by using QR-decomposition. Maximizing the log-likelihood will then be done in an iterative manner again. We can use almost the same approach as in Abegaz and Wit (2013) by alternatively updating $\hat{\Theta}$ and $\hat{\Gamma}_1, \ldots, \hat{\Gamma}_K$. We start with

$$\hat{\Theta}^{(l+1)} = \arg\max_{\Theta}\Big\{\log\det(\Theta) - \operatorname{tr}\Big[\big(\sum_{k=1}^{K} S_{\Gamma_k}\big)\Theta\Big] + \sum_{i \neq j}^{p} P_\lambda(|\theta_{ij}|)\Big\},$$

exactly as we did in the stationary STSCGM. Using this new value of $\hat{\Theta}$, we obtain estimates by maximizing (10) for Γ_k, letting Θ and every other $\Gamma_{i \neq k}$ fixed.

$$\hat{\Gamma}_k^{(l+1)} = \arg\max_{\Gamma_k}\Big\{ - \operatorname{tr}(S_{\Gamma_k}\Theta) - \sum_{i,j}^{p} P_\rho(|\gamma_{k_{ij}}|)\Big\}$$

It takes little effort to see that this can be solved in the same manner as (8). The optimal pair of (λ, ρ) is obtained by finding the pair that minimizes the BIC function

$$\mathrm{BIC}(\lambda, \rho) = -nT\big[\log\det(\hat{\Theta}_\lambda) - \operatorname{tr}(S_{\hat{\Gamma}_\rho})\big] + \log(nT)\big(\frac{a}{2} + \sum_{i=1}^{K} b_i + p\big) \qquad (11)$$

Here a is the number of off-diagonal non-zero parameters of Θ, p the number of diagonal parameters of Θ, all non-zero, and b_i the number of parameters of transition matrix Γ_i. The best pair of (λ, ρ) is again found by a grid search.

2.2 Slowly changing sparse time series chain graphical model

The second alteration we will look at is the slowly changing STSCGM. Here we assume again that the transition matrix Γ changes over time. However, we don't expect to change very drastically. Small changes in for example the second fold could change the transition matrix a little bit, but we can still use the data to estimate the transition matrix in the first fold.

Let the time points at which the transition matrix changes be T_k and let $T_1 = 0$, the first time it changes. We can then divide the data in K folds. We define the K transition matrices Γ_k as

$$\begin{aligned}
&\Gamma_1 = D_1, \text{ for } T_1 < t \leq T_2\\
&\Gamma_2 = \Gamma_1 + D_2 = D_1 + D_2, \text{ for } T_2 < t \leq T_3\\
&\vdots\\
&\Gamma_K = \Gamma_{k-1} + D_K = D_1 + D_2 + \ldots + D_K, \text{ for } T_K < t \leq T
\end{aligned}$$

So we let the transition matrix slowly change over time, instead of choosing a different Γ for each fold. The previous model with change points cannot be used in this case, since every Γ is dependent of its predecessors.

To introduce sparsity in the model, we could apply the same method as before. We add a penalty term to the log-likelihood and try to solve it in an iterative way. However, simply penalizing the elements of D_k will only introduce sparsity in the D_k matrices and not in the Γ_k matrices. Vice versa, penalizing the elements of Γ_k will not bring sparsity in the D_k matrices. Penalizing both the elements of D_k and Γ_k could be a solution, but the belonging penalized likelihood is hard to maximize. We propose a new approach of obtaining sparse estimates of both D_k and Γ_k.

We start with finding initial estimates for $\Gamma_1, \ldots, \Gamma_K$. Although Γ_i affects its successors, we will not take this into consideration yet. Thus we can easily find the initial estimate for Γ_i by using QR-decomposition. Our initial estimates for D_i become

$$\hat{D}_k = \hat{\Gamma}_k - (\hat{D}_{k-1} + \hat{D}_{k-2} + ... + \hat{D}_1) \text{ for } k = 1, \ldots, K$$

Now we find estimates for Θ and the D_i's in an iterative manner as before.

The updating step for Θ is the same as in the previous two models. Setting

$$\hat{\Gamma}_k = \sum_{i=1}^{k} \hat{D}_i,$$

we define S_{Γ_k} in the same way as in the STSCGM with change points. We then update $\hat{\Theta}$ in the same way.

$$\hat{\Theta}^{(l+1)} = \arg\max_{\Theta} \Big\{ \log\det(\Theta) - \mathrm{tr}\Big[\big(\sum_{k=1}^{K} S_{\Gamma_k}\big)\Theta\Big] + \sum_{i\neq j}^{p} P_\lambda(|\theta_{ij}|)\Big\}$$

For updating $\hat{D}_k$ we will have to transform the data. D_k only affects data for $t > T_k$, so we want to *clean* the data from any influence of the other D_i's. Suppose X_t, $t > T_k$, is affected by $D_1, \dots, D_k, \dots, D_m$. Then X_t can be written as.

$$X_t = (D_1 + \ldots + D_i + \ldots + D_m)X_{t-1} + \epsilon \quad \text{where } \epsilon \sim N(0, \Sigma)$$

We then define the transformed data $\tilde{X}_t$ as follows.

$$\begin{aligned} \tilde{X}_t &= X_t - (D_1 + \ldots + D_{k-1} + D_{k+1} + \ldots + D_m)X_{t-1} \\ &= D_k X_{t-1} + \epsilon \end{aligned}$$

That is, we predict the effect that the $D_{i\neq k}$'s have on the data and subtract these fitted values from the data. Now we can apply the theory of the STSCGM to update D_k. Letting

$$\begin{aligned} S_{D_k} &= \frac{1}{n(T - T_k)} \sum_{i=1}^{n} \sum_{t=T_k+1}^{T} (\tilde{X}_t - D_k X_{t-1})(\tilde{X}_t - D_k X_{t-1})' \\ &= \tilde{S}_y - \tilde{S}_{yx} D_k' - D_k \tilde{S}_{yx}' + D_k \tilde{S}_x D_k' \end{aligned}$$

with

$$\tilde{S}_y = \frac{1}{n(T - T_k)} \sum_{i=1}^{n} \sum_{t=T_k+1}^{T} \tilde{X}_t \tilde{X}_t{}', \quad \tilde{S}_{yx} = \frac{1}{n(T - T_k)} \sum_{i=1}^{n} \sum_{t=T_k+1}^{T} \tilde{X}_t X_{t-1}'$$

$$\tilde{S}_x = \frac{1}{n(T - T_k)} \sum_{i=1}^{n} \sum_{t=T_k+1}^{T} X_{t-1} X_{t-1}'$$

We then update $\hat{D}_k$ by maximizing the penalized log-likelihood with respect to D_k.

$$\hat{D}_k^{(l+1)} = \arg\max_{D_k} \Big\{ -\mathrm{tr}(S_{D_k}\Theta) - \sum_{i\neq j}^{p} P_\rho(|D_{k_{ij}}|)\Big\} \tag{12}$$

This gives us sparse estimate $\hat{D}_k$ and we let $\hat{\Gamma}_k = \hat{\Gamma}_{k-1} + \hat{D}_k$. Now we update $\hat{\Gamma}_k$ in a same manner as in (10). That is,

$$\hat{\Gamma}_k = \arg\max_{\Gamma_k} \Big\{ -\mathrm{tr}(S_{\Gamma_k}\Theta) - \sum_{i,j}^{p} P_\mu(|\gamma_{k_{ij}}|)\Big\}. \tag{13}$$

Now we could update the estimate for D_k again by letting

$$\hat{D}_k = \hat{\Gamma}_k - \hat{\Gamma}_{k-1}$$

and move on to the updating process of D_{k+1}. However, this does not lead to the desired sparsity in both $\hat{\Gamma}_k$ and $\hat{D}_k$. Instead, we update the estimate for D_k in the following way. Find the zeros in $\hat{\Gamma}_k$. Suppose there occurs a zero at (x, y), so $\hat{\Gamma}_{k_{xy}} = 0$. Then change the estimate $\hat{D}_k$ such that $\sum_{i=1}^{k} \hat{D}_{i_x y}$ is zero. That is, let

$$\hat{D}_{k_{xy}} = -\sum_{i=1}^{k-1} \hat{D}_{i_{xy}} \ , \tag{14}$$

completing the updating process for D_k. Notice that there are 3 penalty parameters involved. We have the penalty parameterized by λ for Θ as in the STSCGM. Parameter ρ now regulates the sparsity in the D_k matrices and μ regulates the sparsity in the Γ_k matrices. The last 2 have an interesting property. Letting ρ or μ be smaller gives fewer zeros in D_k, so the transition matrix will change more over time, while letting ρ and μ be bigger gives more zeros in D_k which yields less change over time in the transition matrix.

Consider the following example where the algorithm is briefly explained. Suppose we want to update $\hat{D}_k$. We have an estimate for Γ_{k-1} and an initial estimate $\hat{D}_k$, obtained by solving (12),

$$\hat{\Gamma}_{k-1} \begin{bmatrix} -0.99 & 0 \\ 0 & -0.98 \end{bmatrix} \ , \quad \hat{D}_k = \begin{bmatrix} 0 & 0 \\ 0 & 0.99 \end{bmatrix}$$

Adding both gives $\hat{\Gamma}_k$, which has one very small entry at (2,2) unequal to zero. In (13) we will probably find a zero in that entry, that is, something like

$$\hat{\Gamma}_k = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \tag{15}$$

depending on the choice of penalty parameter μ. In (14), we set

$$\hat{D}_{k_{2,2}} = -\sum_{i=1}^{k-1} \hat{D}_{i_{2,2}} = -\hat{\Gamma}_{k-1_{2,2}} = 0.98$$

So by sacrificing a little bit of prediction accuracy, we obtain sparse estimates for both Γ_k and D_k.

The weakness of the STSCGM with change points is that the change points in the estimated model have to be exactly the same as the true values, if the change in Γ is very large. For the slowly changing STSCGM the same problem holds. The solution of this problem combines the two alterations. Letting the number of folds in the slowly changing STSCGM be large, we obtain a great number of D_i matrices. A matrix with a lot of small or zero elements can only be caused by a transition matrix that doesn't change in that period of time. Vice versa, a matrix with many bigger non-zero parameters can only be caused by a big change in the transition matrix. These matrices reveal the underlying structure of the data, giving us the true change points of the transition matrix. Now both models can be used to find accurate estimates.

3 Simulations

3.1 Simulation 1

We start by randomly creating two high dimensional ($p = 18$) sparse matrices Γ and Θ. The first one is created in such a way that the absolute value of the sum of each row is smaller than 1, which is a desirable property for simulating data. Also, all values are between 0.5 and 1. The latter is found by creating a sparse upper triangular matrix and premultiply it by its transpose, reversing the Cholesky Decomposition so that we are left with a positive definite matrix. We then simulate 20 replicates of $T = 60$ time points. We will estimate the transition matrix in two ways. We will test both the standard non-stationary STSCGM model and the slowly changing STSCGM, with $K = 5$ transition matrices.

We start by looking at the estimates of the stationary STSCGM. Figure 4 is the directed graph of the estimate $\hat{\Gamma}$. The grey lines represent true positives, edges that are estimated nonzero correctly. The red lines represent false positives, edges that are estimated nonzero falsely. A dashed red line means that the estimate is smaller than 0.05 in absolute value. Dark blue lines represent false negatives, edges that are estimated zero wrongly.

Figure 4: $\hat{\Gamma}$, stationary STSCGM

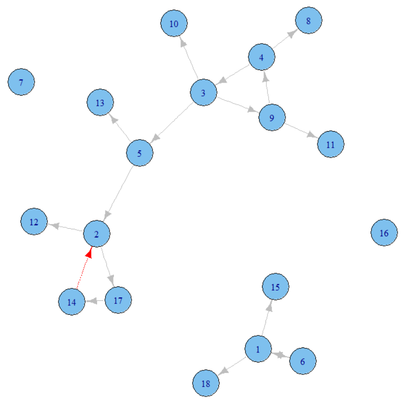
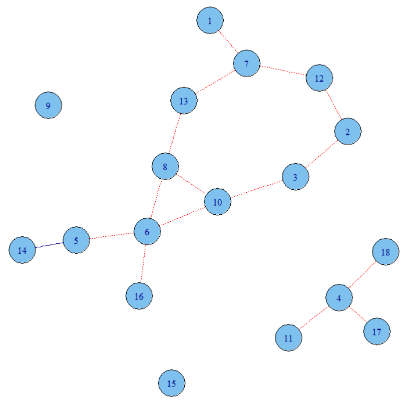


Figure 5: $\hat{\Theta} - \Theta$, stationary STSCGM



Only one extra edge is included in the graph of $\hat{\Gamma}$, but its corresponding value in the matrix is 0.020. Instead of looking at the undirected graph of $\hat{\Gamma}$ with all its edges, we will consider the edges belonging to false positives or false negatives. In the estimate of Θ a few more edges have been included, but all of the corresponding values are smaller than 0.05 in absolute value. Only one edge $(14 - 5)$ has been left out of the model, but this edge belongs to a value of 0.037 in the true precision matrix.

Looking at the slowly changing STSCGM, we find that the estimates are very accurate. Surprisingly, the slowly changing STSCGM has a lower value of the BIC (21,767.14) than the stationary STSCGM (22,089.54). Looking at Figure 6, we see that the first transition matrix graph of the slowly changing STSCGM has all the edges that the graph of the true Γ has, but also two false positive edges from 10 to 16 and 9 to 16. Although these entries are very small, we would like to get rid of them. Ideally, D_2 contains only two entries such that Γ_2 does not have any false positive edges. In Figure 7 we see that the same two edges are present and in Figure 8 we have that the graph of $\hat{\Gamma}_2$ is equal to the graph of Γ, as desired. After this correction, we see no more change in the transition matrix, as D_3 and D_4 is the zero matrix.

Figure 6: $\hat{\Gamma}_1$, changing STSCGM

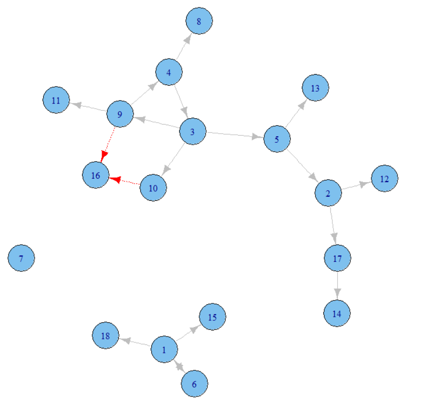


Figure 7: $\hat{D}_2$, changing STSCGM

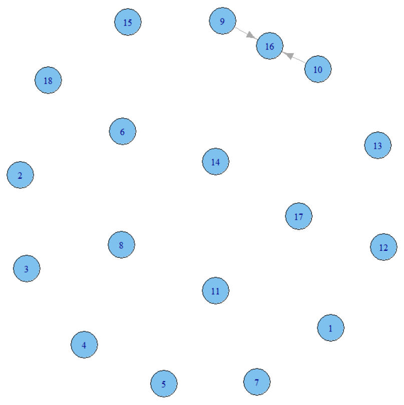


Figure 8: $\hat{\Gamma}_2$, changing STSCGM

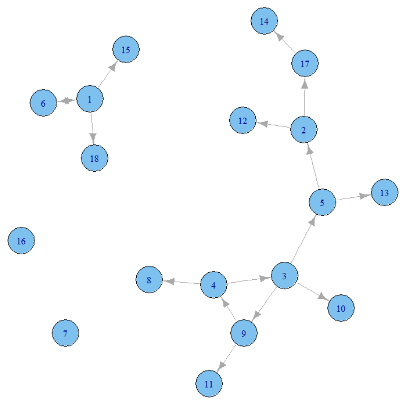
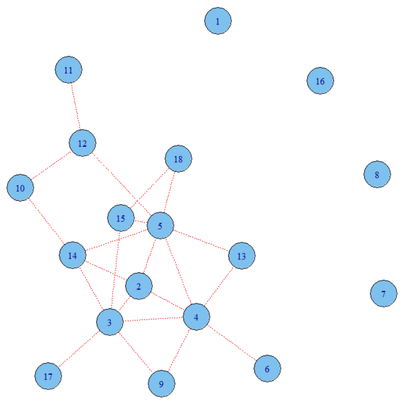


Figure 9: $\hat{\Theta} - \Theta$, changing STSCGM



The difference graph of $\hat{\Theta} - \Theta$ is given by Figure 9. 87.8% of the zeros in Θ were estimated zero correctly. The wrongly included edges all belong to small values in the precision matrix.

Although including the right edges in the model is important we want our estimate to be accurate as well. As a way to measure the distance between matrices, we will use the Frobenius norm given by

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

for an $m \times n$ matrix. We then see that for the stationary STSCGM

$$\left\|\Gamma - \hat{\Gamma}\right\|_F = 0.056, \quad \left\|\Theta - \hat{\Theta}\right\|_F = 0.417$$

and for the changing STSCGM

$$\max_{\Gamma_k}\left\{\left\|\Gamma - \hat{\Gamma}_k\right\|_F\right\} = 0.066, \quad \left\|\Theta - \hat{\Theta}\right\|_F = 0.471$$

All transition matrix estimates are very accurate. We see that the estimates for Θ are a bit less accurate, as we would expect looking at the Figure 9

3.2 Simulation 2

We start by simulating a high dimensional ($p = 18$) precision matrix Θ and three completely random transition matrices in the same manner as Simulation 1. Γ_1 belongs to $1 \leq t \leq 20$, Γ_2 to $21 \leq t \leq 40$ and Γ_3 to $41 \leq t \leq 60$. We simulate 20 replicates of $T = 60$ time points. We could use the STSCGM with change points to immediately estimate the matrices using change points $cp = (20, 40)$, but this seems a bit like cheating.

Suppose we don't know the real change points. Running the stationary STSCGM, we find that the model does not fit the data very well. We suspect that there are some (major) change points. We could try to find these points by using the slowly changing STSCGM and letting the number of transition matrices K be large, $K = 12$. Doing so, we find that only D_1, D_2, D_5 and D_9 are not close to zero. It is then clear that the change points occur at the beginning of D_5 and D_9, that is, the change points are $t \approx 20$ and $t \approx 40$.

Figure 10: $\hat{\Gamma}_1$, STSCGM with cp

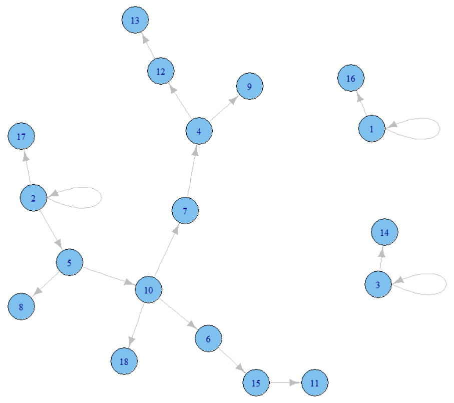


Figure 11: $\hat{\Gamma}_2$, STSCGM with cp

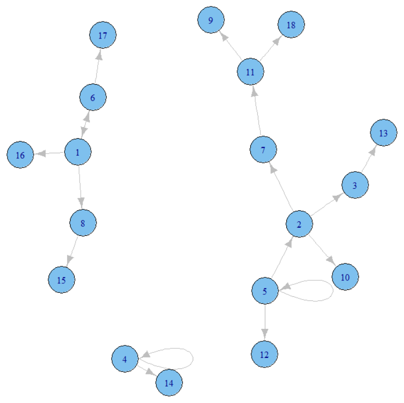


Figure 12: $\hat{\Gamma}_3$, STSCGM with cp

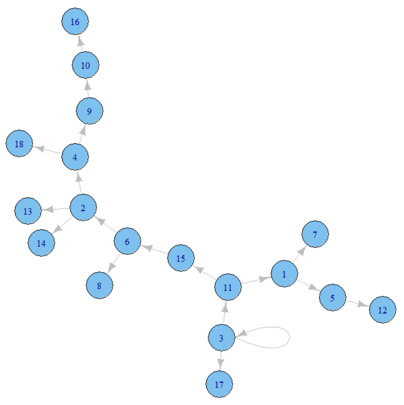
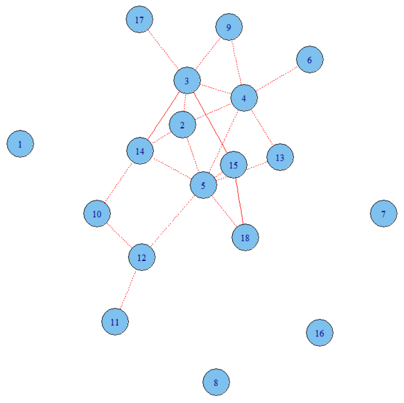


Figure 13: $\hat{\Theta} - \Theta$, STSCGM with cp



The three transition matrices look perfect. No false negative or false positive edges are included in the model. For the estimate of Θ, 82.1% of the real zeros are indeed zero in $\hat{\Theta}$. Out of all the false positives, only 3 edges are belonging to absolute values bigger than 0.05 in the estimated precision matrices.

$$\left\|\Gamma_1 - \hat{\Gamma}_1\right\|_F = 0.069, \quad \left\|\Gamma_2 - \hat{\Gamma}_2\right\|_F = 0.106$$
$$\left\|\Gamma_3 - \hat{\Gamma}_3\right\|_F = 0.081, \quad \left\|\Theta - \hat{\Theta}\right\|_F = 1.610$$

When looking at the norm of the difference of the matrices, we see that the transition matrices are again very close to the true values. The estimate of Θ is a bit less accurate, but considering that it is an 18×18 matrices, the difference is not that big.

3.3 Simulation 3

In the last example we will consider a transition matrix that slowly changes over time. Consider again a high dimensional (p=18) sparse Θ and Γ. We will simulate $n = 20$ replicates of $T = 100$ time points. For simplicity, we let Γ change every 10 time points by removing one or two edges at random. We will look at the performance of both the STSCGM with change points and the slowly changing STSCGM. Since the transition matrix does not change drastically, we expect that the slowly changing STSCGM performs the best.

Letting the number of folds in both models be equal to the true value, 10, we obtain the two estimated model. The slowly changing STSCGM has the lowest BIC, 19,220.4 compared to the STSCGM with change points with a BIC of 19,513.9.

Figure 14: $\hat{\Gamma}_1$, changing STSCGM

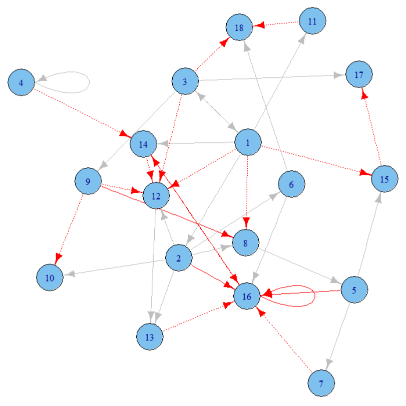


Figure 15: $\hat{\Gamma}_1$, STSCGM with cp

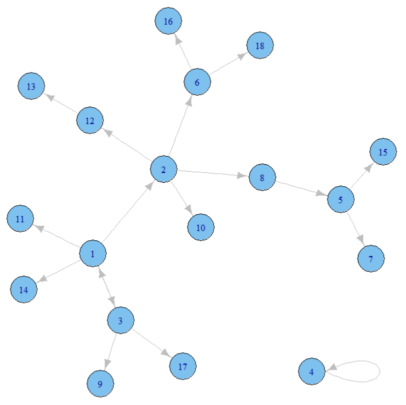


Figure 16: $\hat{\Gamma}_2$, changing STSCGM

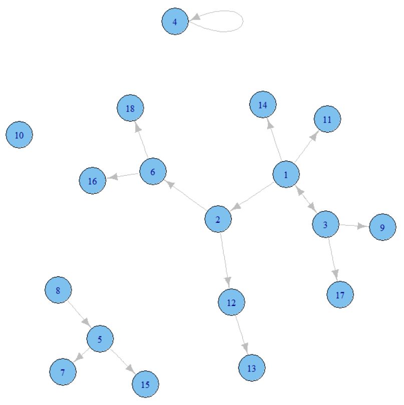


Figure 17: $\hat{\Gamma}_2$, STSCGM with cp

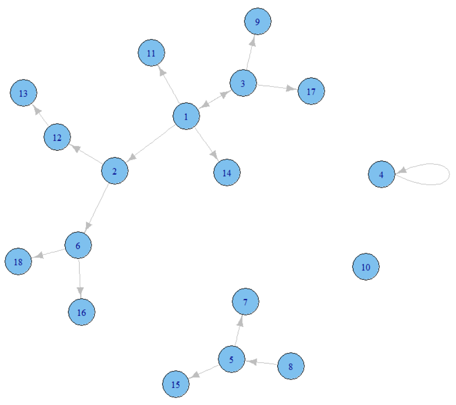


Figure 18: $\hat{\Gamma}_5$, changing STSCGM

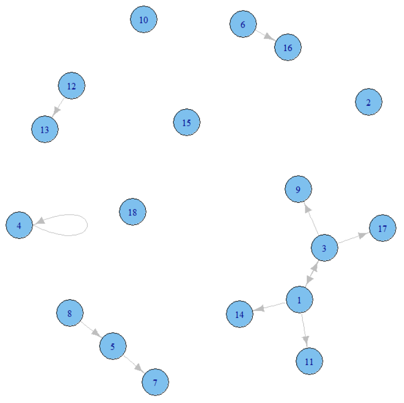


Figure 19: $\hat{\Gamma}_5$, STSCGM with cp

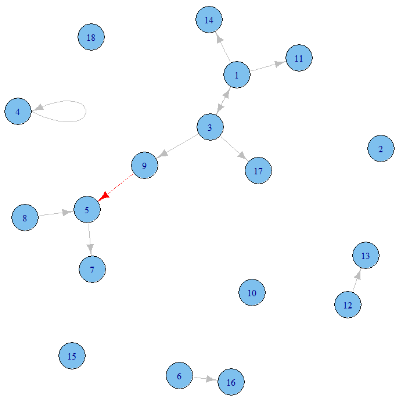


Figure 20: $\hat{\Gamma}_7$, changing STSCGM

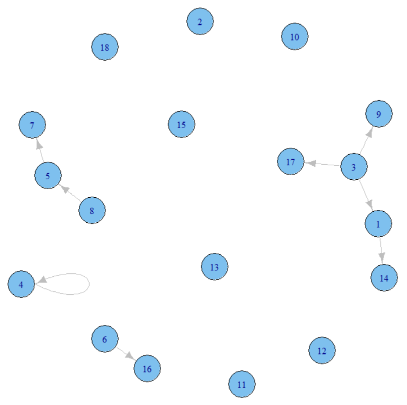


Figure 21: $\hat{\Gamma}_7$, STSCGM with cp

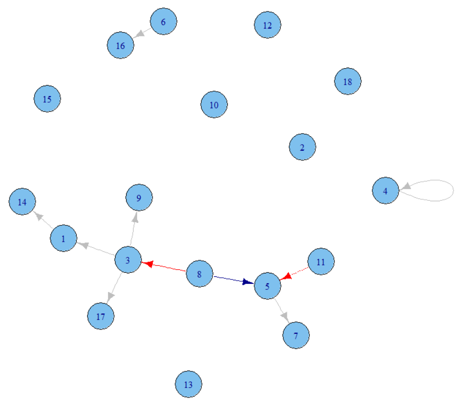


Figure 22: $\hat{\Gamma}_{10}$, changing STSCGM

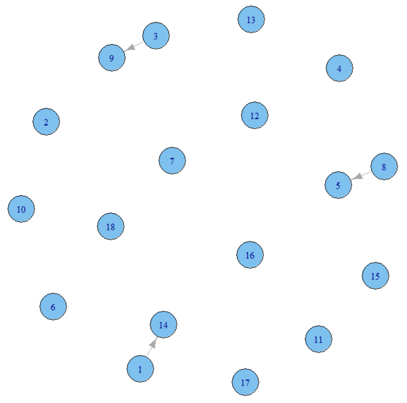
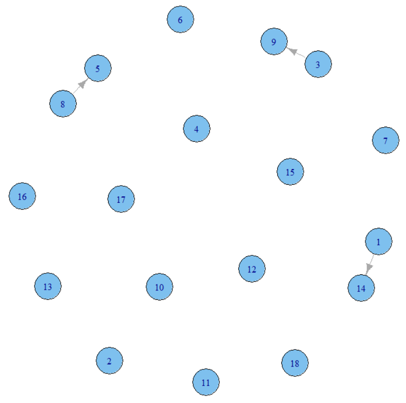


Figure 23: $\hat{\Gamma}_{10}$, STSCGM with cp



As can be seen in Figure 14, in the first transition matrix there are a lot of false positives in the slowly changing STSCGM. However, after this point, not one false negative of false positive has been included in the 9 succeeding matrices. The slowly changing STSCGM has some errors in matrix 5 and 7, but looks very good overall as well.

Figure 24: $\hat{\Theta} - \Theta$, changing STSCGM

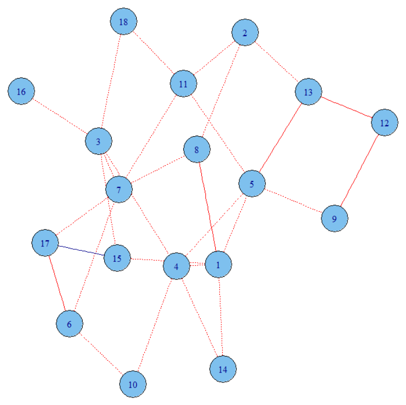
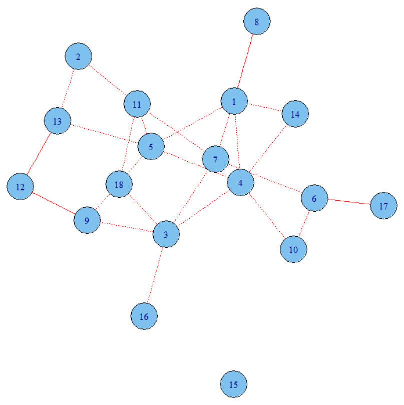


Figure 25: $\hat{\Theta} - \Theta$, STSCGM with cp



As in the previous two simulations, the estimates for Θ include some false positives. For the changing STSCGM and STSCGM with cp, we have

$$\left\|\Theta - \hat{\Theta}\right\|_F = 1.103, \quad \left\|\Theta - \hat{\Theta}\right\|_F = 1.496,$$

respectively. Then we see that effect of the false positives is small and the matrices are very close to the true values. Looking at the transition matrices, we will consider the mean of the norms of the ten difference matrices. Then we see that the slowly changing STSCGM ($\overline{x} = 0.074$) does a bit better than the STSCGM with change points ($\overline{x} = 0.211$), as expected.

4 Conclusion and Discussion

We considered 3 sparse models for time series chain graphs. The models combine the features of the Gaussian graphical model and time series chain graphs to infer conditional relationships between variables. By adding a penalty term to the log-likelihood we obtain sparse estimates. The theory behind the stationary STSCGM has been discussed extensively, explaining each step carefully. Using this theory, we proposed two non-stationary sparse models. In the first model, we considered possible drastic changes in the transition matrix Γ. We showed that almost no extra theory is needed for solving the belonging optimization problem. In the second model we let Γ slowly change over time. We proposed a method for obtaining sparse estimates for both the difference matrices and the transition matrices. For all the models, we used the SCAD penalty to obtain the desired sparsity. The models performed really well on the simulated data, having the desired sparsity and accuracy.

For discussion and further improvement, one could test the robustness of these models by testing on data which violates one of the assumptions. The model assumptions, Gaussian error terms, linear dynamics and first-order Markov properties are quite demanding. One should carefully investigate if all hold for time series data before using the model. Also, the STSCGM with change points requires that the exact change points are known. Although the change points can be approximated by using the slowly changing STSCGM, this doesn't seem the best option. The estimates of the first transition matrix of the slowly changing STSCGM look inaccurate. Although this effect is erased by the second difference transition matrix, there could be a better way. Further studies could also consider different model selection criteria than the BIC and a Markovian property of order $d \geq 1$.

A
Proofs and calculations

A.1 Section 1

Log-likelihood of the stationary model

Let the conditional pdf of $X_t|X_{t-1}$ be

$$f(X_t|X_{t-1}) = (2\pi)^{-p/2}\det(\Sigma)^{-1/2}\exp\{-\frac{1}{2}(X_t-\Gamma X_{t-1})'\Sigma^{-1}(X_t-\Gamma X_{t-1})\}.$$

Assume that we have n replicates of T time points of p variables. Using the equality $\mathrm{tr}(x'Ax) = \mathrm{tr}(xx'A)$, the conditional log-likelihood can be written as

$$\begin{aligned}
\ell(\Gamma,\Theta) &= \sum_{i=1}^{n}\sum_{t=1}^{T}\log f(X_t|X_{t-1}) \\
&= -\frac{npT}{2}\log(2\pi) - \frac{nT}{2}\log\det(\Sigma) - \frac{1}{2}\sum_{i=1}^{n}\sum_{t=1}^{T}(X_t-\Gamma X_{t-1})'\Sigma^{-1}(X_t-\Gamma X_{t-1}) \\
&= \frac{nT}{2}\log\det(\Theta) - \frac{1}{2}\sum_{i=1}^{n}\sum_{t=1}^{T}\mathrm{tr}\left\{(X_t-\Gamma X_{t-1})'\Theta(X_t-\Gamma X_{t-1})\right\} + c \\
&= \frac{nT}{2}\log\det(\Theta) - \frac{1}{2}\mathrm{tr}\left\{\sum_{i=1}^{n}\sum_{t=1}^{T}(X_t-\Gamma X_{t-1})(X_t-\Gamma X_{t-1})'\Theta\right\} + c \\
&= \frac{nT}{2}\log\det(\Theta) - \frac{nT}{2}\mathrm{tr}(S_\Gamma\Theta) + c \\
&\propto \log\det(\Theta) - \mathrm{tr}(S_\Gamma\Theta) + c_2
\end{aligned}$$

Derivatives of g in (8)

Before we differentiate g, we define the matrix whose elements are the partial derivatives of a scalar function $f(A)$ as $\nabla_A f(A)$. Then we have that

$$\begin{aligned}
&\nabla_A \mathrm{tr}(AB) = B' \\
&\nabla_A \mathrm{tr}(A'B) = B
\end{aligned}$$

Using the product rule, we get

$$\begin{aligned}
\nabla_A \mathrm{tr}(ABA'C) &= \nabla_A \mathrm{tr}(ABX'C) + \nabla_A \mathrm{tr}(XBA'C) \quad \text{(letting X = A)} \\
&= (BX'C)' + \nabla_A \mathrm{tr}(CXBA') \\
&= (BX'C)' + \nabla_A \mathrm{tr}(AB'X'C') \\
&= B'AC' + BAC,
\end{aligned}$$

using that the trace is invariant under cyclic permutations. Thus differentiating g with respect to γ_{ij} using the properties of the trace and the symmetry of xtx and Θ gives

$$\begin{aligned}
\nabla_\Gamma\ g(\Gamma) &= \nabla_\Gamma \left[\, \mathrm{tr}(S_{yx}\Gamma'\Theta + \Gamma S'_{yx}\Theta - \Gamma S_x \Gamma'\Theta)\right] \\
&= \nabla_\Gamma\, \mathrm{tr}(S_{yx}\Gamma'\Theta) + \nabla_\Gamma\, \mathrm{tr}(\Gamma S'_{yx}\Theta) - \nabla_\Gamma\, \mathrm{tr}(\Gamma S_x\Gamma'\Theta) \\
&= \nabla_\Gamma\, \mathrm{tr}(\Theta\Gamma S'_{yx}) + \nabla_\Gamma\, \mathrm{tr}(\Gamma S'_{yx}\Theta) - \nabla_\Gamma\, \mathrm{tr}(\Gamma S_x\Gamma'\Theta) \\
&= \nabla_\Gamma\, \mathrm{tr}(\Gamma S'_{yx}\Theta) + \nabla_\Gamma\, \mathrm{tr}(\Gamma S'_{yx}\Theta) - \nabla_\Gamma\, \mathrm{tr}(\Gamma S_x\Gamma'\Theta) \\
&= \Theta S_{yx} + \Theta S_{yx} - \Theta\Gamma S_x - \Theta\Gamma S_x \\
&= 2\ \Theta S_{yx} - 2\ \Theta\Gamma S_x
\end{aligned}$$

For the second derivative of g with respect to γ_{ij} we get

$$\begin{aligned}
\frac{\partial^2 g}{\partial\gamma_{ij}^2} &= \frac{\partial g}{\partial\gamma_{ij}}\left[e'_i(2\ \Theta S_{yx} - 2\ \Theta\Gamma S_x)e'_j\right] \\
&= \frac{\partial g}{\partial\gamma_{ij}}\left[2e'_i(\ \Theta S_{yx})e_j - 2e'_i(\ \Theta\Gamma S_x)e'_j\right] \\
&= 0 - e'_i(\Theta)e_i\ e'_i(S_x)e_j \\
&= -\Theta_{ii}\ S_{x_{jj}}\ .
\end{aligned}$$

The diagonals of Θ and xtx are strictly positive. Therefore, the second derivative is strictly negative.

Properties of concave function with L^1 penalty term

Proposition 1. Let $f(x)$ be a strictly concave continuously differentiable function and let $g(x) = f(x) - \lambda|x|$, $\lambda > 0$. Then

$$\begin{aligned}
&\mathrm{sgn}(\max_x g(x)) = \mathrm{sgn}(\max_x f(x)) \text{ or } \max_x g(x) = 0 \\
&\max_x g(x) = 0 \Leftrightarrow |f'(0)| \leq \lambda
\end{aligned}$$

Proof. Let's start with the first part. Without loss of generality, assume $\mathrm{sgn}(x_{\max}) = \mathrm{sgn}(\max_x f(x)) = -1$. For an example, see Figure 3. Then $f'(x_{\max}) = 0$ and because f is strictly concave, $f'(0) < 0$. That means that for any $x > 0$, $g'(x) < 0$. So the maximum of g cannot occur at any positive value of x. Because of the discontinuity it could occur at $x = 0$ or a negative value of x.

We will proof the second part in two parts.
($\Leftarrow$) Let $|f'(0)| \leq \lambda$. Then we have that

$$\lim_{x\uparrow 0} g'(x) = f'(0) + \lambda \geq 0$$
$$\lim_{x\downarrow 0} g'(x) = f'(0) - \lambda \leq 0$$

It takes little effort to see that g achieves its maximum at $x = 0$.
($\Rightarrow$) Let $\max_x g(x) = 0$. Then it must be that

$$\lim_{x\uparrow 0} g'(x) = f'(0) + \lambda \geq 0$$
$$\lim_{x\downarrow 0} g'(x) = f'(0) - \lambda \leq 0$$

Therefore it must be that $|f'(0)| \leq \lambda$ □

A.2 Section 2

Log-likelihood of stscgm with change points

Let the conditional pdf of X_t be

$$X_t|X_{t-1} \sim N(\Gamma_k X_{t-1}, \Sigma) \text{ for } T_{k-1} + 1 \leq t \leq T_k.$$

The likelihood the conditional pdf is

$$L(\Gamma_1, \ldots, \Gamma_k, \Theta) = \prod_{i=1}^{n} \prod_{t=1}^{T} f(X_t|X_{t-1}) = \prod_{i=1}^{n} \prod_{k=1}^{K} \prod_{t=T_{k-1}+1}^{T_k} f_k(X_t|X_{t-1}),$$

The log-likelihood can then be written as

$$
\begin{aligned}
\ell(\Gamma, \Theta) &= \sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{t=T_{k-1}+1}^{T_k} \log f_k(X_t|X_{t-1}) \\
&= -\frac{npT}{2}\log(2\pi) - \frac{nT}{2}\log\det(\Sigma) \\
&\quad - \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{t=T_{k-1}+1}^{T_k} (X_t - \Gamma_k X_{t-1})'\Sigma^{-1}(X_t - \Gamma_k X_{t-1}) \\
&= \frac{nT}{2}\log\det(\Theta) - \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{t=T_{k-1}+1}^{T_k} \operatorname{tr}\Big[(X_t - \Gamma_k X_{t-1})(X_t - \Gamma_k X_{t-1})'\Theta\Big] + c \\
&= \frac{nT}{2}\log\det(\Theta) - \frac{nT}{2}\sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{t=T_{k-1}+1}^{T_k} \frac{1}{nT}\operatorname{tr}\Big[(X_t - \Gamma_k X_{t-1})(X_t - \Gamma_k X_{t-1})'\Theta\Big] + c \\
&\propto \log\det(\Theta) - \sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{t=T_{k-1}+1}^{T_k} \frac{1}{nT}\operatorname{tr}\Big[(X_t - \Gamma_k X_{t-1})(X_t - \Gamma_k X_{t-1})'\Theta\Big] + c \\
&= \log\det(\Theta) - \operatorname{tr}\Big[\sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{t=T_{k-1}+1}^{T_k} \frac{1}{nT}(X_t - \Gamma_k X_{t-1})(X_t - \Gamma_k X_{t-1})'\Theta\Big] + c \\
&= \log\det(\Theta) - \operatorname{tr}\Big[\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{t=T_{k-1}+1}^{T_k} \frac{1}{nT}(X_t - \Gamma_k X_{t-1})(X_t - \Gamma_k X_{t-1})'\Theta\Big] + c \\
&= \log\det(\Theta) - \operatorname{tr}\Big[\sum_{k=1}^{K}(S_{\Gamma_k}\Theta)\Big] + c,
\end{aligned}
$$

where the S_k matrices are defined as follows

$$
\begin{aligned}
S_{\Gamma_k} &= \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=T_{k-1}+1}^{T_k} (X_t - \Gamma_k X_{t-1})(X_t - \Gamma_k X_{t-1})' \\
&= S_{y_k} - S_{yx_k}\Gamma_k' - \Gamma_k S_{yx_k}' + \Gamma_k S_{x_k}\Gamma_k'
\end{aligned}
$$

B
R code: functions

cp.stscgm

```
cp.stscgm <-function(data = data, lam = lam, rho = rho, cp = NULL,
setting = setting)
{
  dim.data <- dim(data)
  t <- dim.data[1] -1
  p <- dim.data[2]
  n <- dim.data[3]

  #if setting is not entered, use default setting
  if (is.null(setting)){
    setting = list()
    setting$maxit.out = 50
    setting$maxit.in = 15
    setting$tol.out = 1e-2
    setting$tol.in = 1e-2
    setting$silent = FALSE
  }

  res <- BIC.cp.stscgm(data = data, lam = lam, rho = rho, cp = cp,
setting=setting)

  return(res)
  }
```

BIC.cp.stscgm

```
BIC.cp.stscgm <- function(data = data, lam = lam, rho = rho, cp = cp,
setting = setting)
{
  t = dim(data)[1] -1
  p = dim(data)[2]
  n = dim(data)[3]
  K = length(cp) + 1

  BIC <- Inf
  for (u in lam){
    for (v in rho){
      res.tmp <- compute.cp.stscgm(data, lam = u, rho = v, cp = cp,
setting = setting)
      theta = res.tmp$theta
      S = res.tmp$S
      G.array = res.tmp$G.array
      nzp.T = sum(theta !=0) - p
      nzp.G <- 0
      for (i in 1:K) nzp.G = nzp.G + sum(G.array[,,i] != 0)
      BIC.tmp = -n*t *( log(det(theta)) - sum(diag(S %*% theta))) +
log(n*t)*(nzp.T/2 + nzp.G + p) #BIC
      if (BIC.tmp < BIC){ #save best BIC
        BIC = BIC.tmp
        res <- res.tmp
        lam.opt = u
        rho.opt = v
      }
    }
  }
  return(list(theta = res$theta, G.array = res$G.array, lam.opt = lam.opt,
rho.opt = rho.opt, BIC = BIC))
}
```

compute.cp.stscgm

```
compute.cp.stscgm <- function(data = data, lam, rho, cp = cp, setting
= setting)
{
  t = dim(data)[1] -1
  p = dim(data)[2]
  n = dim(data)[3]
  K = length(cp) + 1

  change = c(1,cp,t+1)
  data.list = list(data[change[1]:change[2],,]) #create data.list
  if (length(change) > 2){
    for (j in (2:(length(change)-1))){
      data.list <- append(data.list,list(data[(change[j]:change[j+1]),,]))
    }}

  #storage arrays
  G.array <-array(NA, dim=(c(p,p,K)))
  xtx.array <- array(NA, dim=(c(p,p,K)))
  xty.array <- array(NA, dim=(c(p,p,K)))
  yty.array <- array(NA, dim=(c(p,p,K)))
  xtxt.array <- array(NA, dim=(c(p,p,K)))
  samp.cov.arr <- array(NA, dim=(c(p,p,K)))
  #storage vectors
  mab.vec <- vector(mode="integer", length = K)
  G.dist.vec <- vector(mode="integer", length = K)
  for (i in 1:K)
  {
    data.tmp = data.list[[i]]
    tmp = datamatrices(data.tmp)
    xtx.array[,,i] = tmp$xtx
    xty.array[,,i] = tmp$xty
    yty.array[,,i] = tmp$yty

    t.tmp = (dim(data.tmp))[1] -1
    G.tmp = t(qr.solve(tmp$xtx + n*t.tmp*rho*diag(p), tmp$xty))
    if(!is.numeric(G.tmp)) G.tmp <- matrix(0,p,p)
    G.array[,,i] = G.tmp

    mab.tmp = sum(abs(G.tmp))
    mab.vec[i] = mab.tmp
  }

  k.out = 0
  while(1)
  {
    k.out = k.out + 1

    #calculate S
    S = matrix(0,p,p)
    for (i in 1:K)  S = S + (yty.array[,,i] - t(xty.array[,,i]) %*%
t(G.array[,,i]) - G.array[,,i] %*% xty.array[,,i] +
                            G.array[,,i] %*% xtx.array[,,i] %*% t(G.array[,,i]))/(n*t)

    #update theta
    theta = update.theta(S = S, rho = lam)

    #update gammas
    warmstart = 1
    if (k.out == 1) warmstart = 0

    for (i in 1:K)
    {
      G.tmp = G.array[,,i]
      tmp = SCAD(M = G.tmp, a = 3.7, lam = rho)
      wt.G.tmp = tmp*n*t
      xty.tmp = xty.array[,,i]
      xtx.tmp = xtx.array[,,i]
      mab.tmp = mab.vec[i]
      new.G.tmp = rblasso(xtx = xtx.tmp, xty = xty.tmp, wt = wt.G.tmp,
tol = 1e-5, sbols = mab.tmp,
                          maxit =setting$maxit.in, warm = warmstart,
old.T = theta, old.G = G.tmp)

      if(!is.numeric(new.G.tmp)) new.G.tmp <- matrix(0, nrow = p, ncol
= p)

      G.dist.tmp = sum(abs(G.tmp - new.G.tmp))
      G.dist.vec[i] = G.dist.tmp
      G.array[,,i] = new.G.tmp
    }

    tol = (setting$tol.out)*mab.vec
    if (sum(G.dist.vec < tol) == K) break
    if (k.out > setting$maxit.out) break
    cat(G.dist.vec,"\n")
  } #end while loop


  #extra sparsity
  for (i in 1:K) G.array[,,i] = G.array[,,i]*(1*(abs(G.array[,,i])
> 0.01))

  #calculate S
  S = matrix(0,p,p)
  for (i in 1:K)  S = S + (yty.array[,,i] - t(xty.array[,,i]) %*% t(G.array[,,i])
- G.array[,,i] %*% xty.array[,,i] +
                             G.array[,,i] %*% xtx.array[,,i] %*% t(G.array[,,i]))/(n*t)

  if (!setting$silent) cat(k.out, "\n")

  return(list(theta = theta, G.array = G.array, S = S))
}
```

datamatrices

```
datamatrices <- function(data.m = data.m)
{
  dim = dim(data.m)
  t = dim[1]
  X = data.m[1:(t-1),,]
  Y = data.m[2:t,,]

  dim = dim(X)
  T <- dim[1]
  p <- dim[2]
  n <- dim[3]
  xty.i <- array(NA, c(p,p,n))
  xtx.i <- array(NA, c(p,p,n))
  yty.i <- array(NA, c(p,p,n))

  for(i in 1:n){
    XX <- X[,,i]
    YY <- Y[,,i]
    xty.i[,,i]=crossprod(XX,YY)
    xtx.i[,,i]=crossprod(XX)
    yty.i[,,i]=crossprod(YY)
  }
  xty =apply(xty.i, c(1,2), sum)
  xtx =apply(xtx.i, c(1,2), sum)
  yty =apply(yty.i, c(1,2), sum)

  return(list(xty = xty,xtx = xtx,yty = yty))
}
```

update.theta

```
update.theta <- function(S = S, rho = rho)
{
  p = dim(S)[1]
  #estimate theta with lasso penalty
  lasso.out <- glasso(s=S, rho=rho, thr=1.0e-4, maxit=1e4, penalize.diagonal=FALSE,
approx=FALSE)
  lasso.T=lasso.out$wi
  lasso.T.i=lasso.out$w
  if(!is.numeric(lasso.T)) lasso.T <- diag(p)
  if(!is.numeric(lasso.T.i)) lasso.T.i <- diag(p)

  wt.T <- SCAD(M = lasso.T, a = 3.7, lam = rho) #approximate scad penalty
with LLA

  #scad estimate
  SCAD.out <- glasso(s=S, rho=wt.T, thr=1.0e-4, maxit=1e4, penalize.diagonal=FALSE,
                     approx=FALSE, start="warm", w.init=lasso.T.i,
wi.init=lasso.T)
  old.T = SCAD.out$wi
  return(theta = old.T)
}
```

SCAD

```
SCAD <- function(M = M, a = a, lam = lam)
{
  #approximate scad penalty with LLA
  wt <- matrix(NA,ncol =dim(M)[1],nrow = dim(M)[2])
  for(i in 1:dim(M)[1]){
    for(j in 1:dim(M)[2]){
      if(abs(M[i,j]) <= lam ) wt[i,j] <- lam
      else { if((lam <= abs(M[i,j])) & (abs(M[i,j]) <  a*lam )) {
        wt[i,j] <- ((a*lam-abs(M[i,j]))/((a-1))) }
        else wt[i,j] <-  0 }
    }}
  return(wt)
}
```

rblasso

```
rblasso <- function(xtx = xtx, xty = xty, wt = wt, tol = tol, sbols
= sbols,
                    maxit = maxit, warm = warm, old.T = old.T, old.G
= old.G)
{
  p = dim(xtx)[1] #dim
  tol1 = tol*sbols
  G = matrix(0, nrow=p, ncol = p)
  if(warm==1) G = old.G
  k.it = 0

  while(1){
    Gdiff = 0
    k.it = k.it + 1
    for(i in 1:p){
      for(j in 1:p){
        #update G_ij
        gij = (old.T[i,i]*xtx[j,j])*G[i,j] + ((old.T%*%t(xty))[i,j]-
old.T%*%G%*%xtx)[i,j];
        tmp =(abs(gij) - wt[i,j])/(old.T[i,i]*xtx[j,j]);
        gnew=0;
        if(tmp  > 0 ){
          if(gij > 0) gnew = tmp;
          if(gij < 0) gnew = -tmp;
        }
        Gdiff= Gdiff + abs(G[i,j]-gnew);
        G[i,j]=gnew;
      }}
    if  (k.it > maxit) break
    if  (Gdiff < tol1) break
  }
  return(G)
}
```

nonst.stscgm

```
nonst.stscgm <- function (data = data, lam = lam, rho = rho , mu =
mu, K = 1, setting = NULL)
{
  t = dim(data)[1]
  n = dim(data)[3]
  if (K%%1 != 0){
    stop
    cat("number of folds must be integer geq 1", "\n")
  }
  if (K < 1){
    stop
    cat("number of folds to must be geq 1", "\n")}
  if (n*t/20 < (K+1)){
    stop
    cat("number of folds to big for data of dim t,n", "\n")
  }

  #if setting is not entered, use default setting
  if (is.null(setting)){
    setting = list()
    setting$maxit.out = 50
    setting$maxit.in = 15
    setting$tol.out = 1e-2
    setting$tol.in = 1e-2
    setting$silent = FALSE
  }

  res <- BIC.nonst.stscgm(data = data, lam = lam, rho = rho, mu = mu,
K = K, setting = setting)

  return(res)
}
```

BIC.nonst.stscgm

```
BIC.nonst.stscgm <- function(data=data,lam=lam, rho=rho, mu=mu, K =
K, setting = setting)
{
  t <- dim(data)[1] -1
  p <- dim(data)[2]
  n <- dim(data)[3]
  BIC <- Inf
  for (u in lam){
    for (v in rho){
      for (w in mu){
        res.tmp <- compute.nonst.stscgm(data, pen.pm = c(u,v,w), K
= K, setting = setting)
        tmp.T = res.tmp$theta
        sgamma = res.tmp$S
        D.array = res.tmp$D.array
        nzp.T = sum(tmp.T !=0) - p #number of nonzero off-diagonal
parameters
        nzp.D = 0
        for (i in 1:K) nzp.D = nzp.D + sum(D.array[,,i] != 0) #number
of nonzero parameters of D_i

        BIC.tmp = -n*t *( log(det(tmp.T)) - sum(diag(sgamma %*% tmp.T)))
+ log(n*t)*(nzp.T/2 + nzp.D + p)

        if (BIC.tmp < BIC){ #save best BIC
          BIC = BIC.tmp
          res <- res.tmp
        }}}}

  return(list(theta = res$theta, D.array = res$D.array, G.array = res$G.array,
lam= res$lam, rho = res$rho,
              mu = res$mu, BIC = BIC))
}
```

compute.nonst.stscgm

```
compute.nonst.stscgm <- function(data = data, pen.pm = pen.pm, K =
K, setting = setting)
{
  t <- dim(data)[1]
  p <- dim(data)[2]
  n <- dim(data)[3]
  lam = pen.pm[1]
  rho = pen.pm[2]
  mu = pen.pm[3]


  cp = seq(1,t,length.out = K+1)

  data.list = list(data[cp[1]:cp[2],,])
  if (length(cp) > 2){
    for (j in (2:(length(cp)-1))){
      data.list <- append(data.list,list(data[(cp[j]:cp[j+1]),,]))
    }}
  #storage arrays
  D.array <- array(NA, dim=c(p,p,K))
  G.array = array(NA, dim=c(p,p,K))
  xtx.array <- array(NA, dim=(c(p,p,K)))
  xty.array <- array(NA, dim=(c(p,p,K)))
  yty.array <- array(NA, dim=(c(p,p,K)))

  #storage vectors
  t.vec <- vector(mode="integer", length = K)
  G.dist.vec <- vector(mode="integer", length = K)
  mab1 <- vector(mode="integer", length =K)
  mab2 <- vector(mode="integer", length = K)

  #initial steps
  for (i in 1:K){
    data.tmp = data.list[[i]]
    tmp = datamatrices(data.tmp) #create xtx_i etc. for all K data
parts
    xtx.array[,,i] = tmp$xtx
    xty.array[,,i] = tmp$xty
    yty.array[,,i] = tmp$yty
    t.tmp = (dim(data.tmp))[1] -1
    t.vec[i] = t.tmp
    G.array[,,i] = t(qr.solve(tmp$xtx + n*t.tmp*mu * diag(p), tmp$xty))
    if(!is.numeric(G.array[,,i])) G.array[,,i] <- matrix(0,p,p)
    if(i == 1) D.array[,,i] <- G.array[,,i]
    if(i != 1) D.array[,,i] <- G.array[,,i] - apply(D.array[,,1:(i-1)],c(1,2),sum)
    mab1[i] = sum(abs(D.array[,,i]))
    mab2[i] = sum(abs(G.array[,,i]))
  }
  rm(tmp); rm(data.tmp); rm(t.tmp)

  k.out = 0
  while(1){
    k.out = k.out + 1
    warmstart = 1
    if (k.out == 1) warmstart = 0

    #calculate S
    S = matrix(0,p,p)
    for (i in 1:K)  S = S + (yty.array[,,i] - t(xty.array[,,i]) %*%
t(G.array[,,i]) - G.array[,,i] %*% xty.array[,,i] +
                               G.array[,,i] %*% xtx.array[,,i] %*%
t(G.array[,,i]))/(n*t)
    #update theta
    theta = update.theta(S = S, rho = lam)
  #update D_k matrices
  for (i in 1:K){
   data.tf <- data #transform data for updating D_i
   D.tmp <- D.array[,,i]

   #remove predicted effect of the K-1 D_k matrices
   rm.eff = (1:K)[-which((1:K)==i)] #select all matrices but D_i
   for(j in rm.eff){
    for(l in 1:n){
     for(m in t:(cp[j]+1)){
      data.tf[m,,l] = data.tf[m,,l] - D.array[,,j]%*%data[m-1,,l]
     }}}

   tmp = datamatrices2(data.tf = data.tf, data.or = data, cp = cp[i])
   xtx.tmp = tmp$xtx; xty.tmp = tmp$xty;

   tmp = SCAD(M = D.array[,,i], a = 3.7, lam = rho) #estimate SCAD
penalty with LLA
   t.tmp = t - cp[i] #time X is affected by D_i
   wt.tmp = tmp*t.tmp*n

   new.D.tmp = rblasso(xtx = xtx.tmp, xty = xty.tmp, wt = wt.tmp,
tol = 1e-5, sbols = mab1[i],
                maxit =setting$maxit.in, warm = warmstart,
old.T = theta, old.G = D.tmp)
   if(!is.numeric(new.D.tmp)) new.D.tmp <- matrix(0, nrow = p, ncol
= p)

   if (i != 1){
    #update G_k matrices
    G.tmp = apply(D.array[,,1:i-1],c(1,2),sum) + new.D.tmp
    tmp = SCAD(M = G.tmp, a = 3.7, lam = mu)
    wt.tmp = tmp*t*n
    xtx.tmp = xtx.array[,,i]; xty.tmp = xty.array[,,i]
    new.G.tmp = rblasso(xtx = xtx.tmp, xty = xty.tmp, wt = wt.tmp,
tol = 1e-5, sbols = mab2[i],
                 maxit =setting$maxit.in, warm = warmstart,
old.T = theta, old.G = G.tmp)
    if(!is.numeric(new.G.tmp)) new.G.tmp <- matrix(0,p,p)

    for (x in 1:p){
     for (y in 1:p){
      if (abs(new.G.tmp[x,y]) < 0.01) new.D.tmp[x,y]= - (apply(D.array[,,1:(i-1)],c(1
        }}
      }

      G.dist.vec[i] <- sum(abs(D.array[,,i] - new.D.tmp))
      D.array[,,i] <- new.D.tmp
      if (i == 1) G.array[,,i] = new.D.tmp
      if (i != 1) G.array[,,i] = apply(D.array[,,1:(i-1)],c(1,2),sum)
+ new.D.tmp

    }


    cat(G.dist.vec,"\n")
    if (sum(G.dist.vec < setting$tol.out) == K) break #break if all
G/D converge
    if (k.out > setting$maxit.out) break
  }


  #calculate S
  S = matrix(0,p,p)
  for (i in 1:K)  S = S + (yty.array[,,i] - t(xty.array[,,i]) %*% t(G.array[,,i])
- G.array[,,i] %*% xty.array[,,i] +
                           G.array[,,i] %*% xtx.array[,,i] %*% t(G.array[,,i]))/(n*t)

  if (!setting$silent) cat(k.out, "\n")

  return(list(theta = theta, D.array = D.array, G.array = G.array,
S = S, lam = lam, rho = rho, mu = mu))
}
```

datamatrices2

```
datamatrices2 <- function(data.tf = data.tf, data.or = data.or, cp)
{
  if (dim(data.tf)[1] != dim(data.or)[1]) cat("something went terribly
wrong in datamatrices2","\n")
  t = dim(data.or)[1]
  X = data.or[cp:(t-1),,]
  Y = data.tf[(cp+1):t,,]

  dim = dim(X)
  T <- dim[1]
  p <- dim[2]
  n <- dim[3]
  xty.i <- array(NA, c(p,p,n))
  xtx.i <- array(NA, c(p,p,n))
  yty.i <- array(NA, c(p,p,n))

  for(i in 1:n){
    XX <- X[,,i]
    YY <- Y[,,i]
    xty.i[,,i]=crossprod(XX,YY)
    xtx.i[,,i]=crossprod(XX)
    yty.i[,,i]=crossprod(YY)
  }
  xty =apply(xty.i, c(1,2), sum)
  xtx =apply(xtx.i, c(1,2), sum)
  yty =apply(yty.i, c(1,2), sum)
  xtxt =apply(xtx.i, c(1,2), sum)/(n*T)

  return(list(xty = xty,xtx = xtx,yty = yty,xtxt = xtxt))
}
```

sparse.G

```
sparse.G <- function(p.spar = p.spar, p = p)
{
  M <- matrix(runif(p^2),p,p)
  G <- matrix(0,p,p)
  for(x in 1:p){
    for(y in 1:p){
      if (M[x,y] < p.spar){
        G[x,y] = runif(1,.5,1)*sign(runif(1,-1,1))
        if (sum(abs(G[x,])) > 1) G[x,y] = 0
      }}}
  return(G)
}
```

sparse.T

```
sparse.T <- function(p.spar = p.spar, p = p)
{
  L = diag(p)
  for (x in 1:p){
    for (y in 1:p){
      if (y>x){
        r = runif(1,0,1)
        if (r < p.spar){
          if (r< p.spar/2){
          L[x,y] = runif(1,0.5,1)
          } else{
```

```
          L[x,y] = runif(1,-1,-0.5)
        }}}}}
  S.i = t(L)%*%L
  return(S.i = S.i)
}
```

C
R code: simulations

Simulation 1

```
library(MASS); library(glasso); library(igraph)

source("BIC.cp.stscgm.R");
source("BIC.nonst.stscgm.R");
source("compute.cp.stscgm.R");
source("compute.nonst.stscgm.R");
source("cp.stscgm.R");
source("datamatrices.R");
source("datamatrices2.R");
source("nonst.stscgm.R");
source("rblasso.R");
source("SCAD.R");
source("update.theta.R")
source("sparseG.R")

#simulation 1
n = 20;p = 18;t = 61;
set.seed(123)
G = sparse.G(p.spar = 0.2,p)
theta = sparse.T(p.spar = 0.1,p)
sigma = solve(theta)

data = array(NA,dim=c(t,p,n))
set.seed(123)
for (i in 1:n){
  data[1,,i] = rnorm(p,0,1) #starting values
}
for (j in 2:t){
  for (i in 1:n){
    data[j,,i] = mvrnorm(n=1, G%*%data[j-1,,i],sigma)
  }}

res1.1 <- cp.stscgm(data = data, lam = 0.06, rho = 0.15, cp = NULL,
setting = NULL)
res1.2 <- nonst.stscgm(data =data, lam = 0.06, rho = .2, mu = .25,
K = 5, setting= NULL)

norm(G - res1.1$G.array[,,1], type = "F")
norm(G - res1.2$G.array[,,1], type = "F")

#for plots, see plots.R
```

```
#simulation 2
#model with drastic changes in Gamma
set.seed(124)
n = 20;p = 18;t = 61;
G1 = sparse.G(p.spar = 0.2,p)
G2 = sparse.G(p.spar = 0.2,p)
G3 = sparse.G(p.spar = 0.2,p)
theta = sparse.T(p.spar = 0.1,p)
sigma = solve(theta)

data = array(NA,dim=c(t,p,n))
for (i in 1:n){
  data[1,,i] = rnorm(p,0,1) #starting values
}
for (j in 2:21){
  for (i in 1:n){
    data[j,,i] = mvrnorm(n=1, G1%*%data[j-1,,i],sigma)
  }}
for (j in 22:41){
  for (i in 1:n){
    data[j,,i] = mvrnorm(n=1, G2%*%data[j-1,,i],sigma)
  }}
for (j in 42:61){
  for (i in 1:n){
    data[j,,i] = mvrnorm(n=1, G3%*%data[j-1,,i],sigma)
  }}

lam = 0.06; rho = 0.14; mu =0.2
res2.0 <- nonst.stscgm(data =data, lam = lam, rho = rho, mu = mu, K
= 12, setting= NULL)
res2.1 <- cp.stscgm(data = data, lam = 0.06, rho = 0.14, cp = c(21,41),
setting = NULL)

norm(G1 - res2.1$G.array[,,1], type = "F")
norm(G2 - res2.1$G.array[,,2], type = "F")
norm(G3 - res2.1$G.array[,,3], type = "F")
norm(theta - res2.1$theta, type = "F")

#simulation 3
set.seed(125)
n = 10; p = 18; t = 101
G = sparse.G(p.spar = 0.3,p = p)
theta = sparse.T(p.spar = 0.2, p = p)
sigma = solve(theta)
```

```
K = 10
cp = seq(1,t,length.out = K+1)
cp = cp[-1]
cp[K] = cp[K]+1
G.array = array(dim=c(p,p,K))
G.array[,,1] = G

data = array(NA,dim=c(t,p,n))
a=1
for (i in 1:n){
  data[1,,i] = rnorm(p,0,1) #starting values
}
for(j in 2:t){
  for (i in 1:n){
    data[j,,i] = mvrnorm(n=1, G.array[,,a]%*%data[j-1,,i],sigma)
  }
  if (j == cp[a]){
    nonz = which(G.array[,,a] != 0)
    q = sample(c(1,2),1)
    rem = sample(nonz,q)
    G.array[,,a+1] = G.array[,,a]
    G.array[,,a+1][rem] = 0
    a = a+1
  }
}

lam = 0.05; rho = 0.14; mu = 0.22
res3.1 <- cp.stscgm(data=data, lam = lam, rho = rho, cp = seq(11,91,length.out
= 9), setting = NULL)

lam = 0.04; rho = 0.21; mu = 0.23
res3.2 <- nonst.stscgm(data = data, lam = lam, rho = rho, mu = mu,
K = 10, setting = NULL)

dif3.1 <- vector(mode="integer", length = K)
dif3.2 <- vector(mode="integer", length = K)
for (i in 1:10){
  dif3.1[i] = norm(G.array[,,i] - res3.1$G.array[,,i], type = "F")
}
for (i in 1:10){
  dif3.2[i] = norm(G.array[,,i] - res3.2$G.array[,,i], type = "F")
}

mean(dif3.1)
mean(dif3.2)
norm(theta - res3.1$theta, type = "F")
norm(theta - res3.2$theta, type = "F")
```

References

Fentaw Abegaz and Ernst Wit. Sparse time series chain graphical models for reconstructing genetic networks. *Biostatistics*, 14(3):586–599, 2013.

Rainer Dahlhaus and Michael Eichler. Causality and graphical models in time series analysis. *Universitat Heidelberg*, 2003.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96:1348–1360, 2001.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 0(0):1–10, 2007.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.

Joe Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley, 2008.

Yiyun Zhang, Runze Li, and Chih-Ling Tsai. Regularization parameter selections via generalized information criterion. *Journal of the American Statistical Association*, 105(489):312–323, 2010.

Hui Zou and Runze Li. One-step sparse estimates in noncave penalized likelihood models. *The Annals of Statistics*, 36(4):1509–1533, 2008.