

SINGLE Package Vignette

Ricardo Pio Monti, Christoforos Anagnostopoulos and Giovanni Montana

October 13, 2013

Given multiple longitudinal time series we are often interested in quantifying the relationship between the time series over time. For example, given fMRI data corresponding to various regions of interest in the brain we may be interested in measuring their statistical dependencies over time. These relationships can subsequently be summarised as graphs or networks where each node corresponds to a time series (e.g., a BOLD time series for a region of interest) and edges between nodes (or their absence) provide information regarding the nature of the relationship. Here we present a short tutorial and introduction to using the R package **SINGLE** to estimate such dynamic graphs over time from noisy time series data. The **SINGLE** package provides an implementation of the Smooth Incremental Graphical Lasso Estimation algorithm, full details of which can be found in [4].

1 Introduction

There is an increasing interest in summarising the relationships between time series using undirected graphs. Here each node represents a time series (e.g., this can be the BOLD time series for a brain region) and the edge structure serves to summarise the statistical relationship between nodes. Edge structure is commonly estimated using partial correlations. In this case, the absence of an edge between two nodes implies that the two nodes are conditionally independent given all other nodes.

However it is often the case that only a global graph is estimated using the entire time series. Whilst this may be appropriate in some scenarios, it is often the case that we expect the statistical dependencies between nodes to change over time. A clear example of this can be seen by considering fMRI time series of a subject performing alternating tasks: we naturally expect the relationship between brain regions to change depending on task.

In this vignette, we introduce the **SINGLE** package which can be used to estimate dynamic graphs from noisy time series data. The remainder of this vignette is organised as follows: in Section 2 we give a brief motivating example to show the capabilities of the SINGLE algorithm. In Section 3 we give a brief background description of the SINGLE algorithm. Finally, in Section 4 we give a more detailed description of each of the functions in the SINGLE package.

2 Motivating example

Here we present a brief motivational example to show the capabilities of the SINGLE algorithm and the functionality of the SINGLE package.

First we simulate non-stationary data in order to test the performance of the SINGLE algorithm. We simulate the data using the `generate_random_data` function:

```
> library('SINGLE')
> set.seed(1)
> sim = generate_random_data(ROI=5, length=50, seg=3, sparsity=.15)
```

Here `sim$data` is a matrix with 5 columns and 150 rows. For example, this could correspond to fMRI data from 5 regions of interest over a time. As mentioned previously each column represents a node in our graph and each row is a chronologically ordered observation. Thus `sim$data[i,j]` is the *i*th observation of the *j*th node.

The true partial correlation structure over time is stored in the array `sim$true_cov`. We also note that this dataset contains two changepoints, one at the 50th observation and the other at the 100th. We can view the correlation structure for the first 50 observations as follows:

```
> sim$true_cov[, ,1]

      [,1] [,2] [,3] [,4] [,5]
[1,]    1  0.0  0.0    0    0
[2,]    0  1.0  0.6    0    0
[3,]    0  0.6  1.0    0    0
[4,]    0  0.0  0.0    1    0
[5,]    0  0.0  0.0    0    1
```

We can now run the Smooth Incremental Graphical Lasso Estimation (SINGLE) algorithm to estimate the network structure over time. We then use the `plotSINGLE` function to plot the estimated partial correlations between each of the 5 nodes.

```
> data = sim$data
> S = SINGLE(data=data, radius=5, l1=.75, l2=0.5, k=3)
```

Converged in 30 iterations

The SINGLE algorithm is an iterative algorithm and we can see from the result that it has converged in 30 iterations.

```
> plotSINGLE(object=S, index=c(1,2,3,4,5), x.axis = seq(1,150), n.row=2,
+           col.names=seq(1,5), fix.axis=TRUE)
```

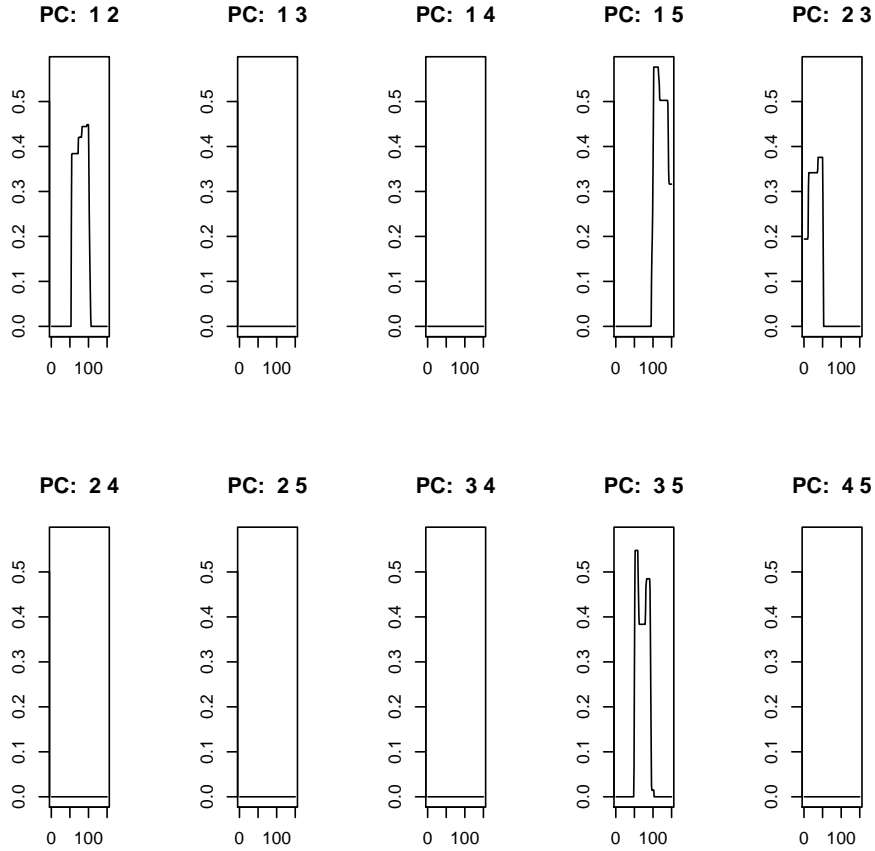


Figure 1: Estimated partial correlations for simulated example

We can see that over the first 50 observations there is only a non-zero partial correlation between nodes 2 and 3 as we would expect. We also note that the estimated partial correlation is lower than 0.6, this is due to the presence of the sparsity (λ_1) penalty. Given this is a simulated example we can assess the performance of the SINGLE algorithm using precision, recall and F scores over time using the `precision_recall` function:

```
> result = precision_recall(true_cov=sim$true_cov, estimated_cov=S$P_)
> plot(result$F1, type='l', ylim=c(0,1), ylab='',
+      main='F Score', xlab='Time')
```

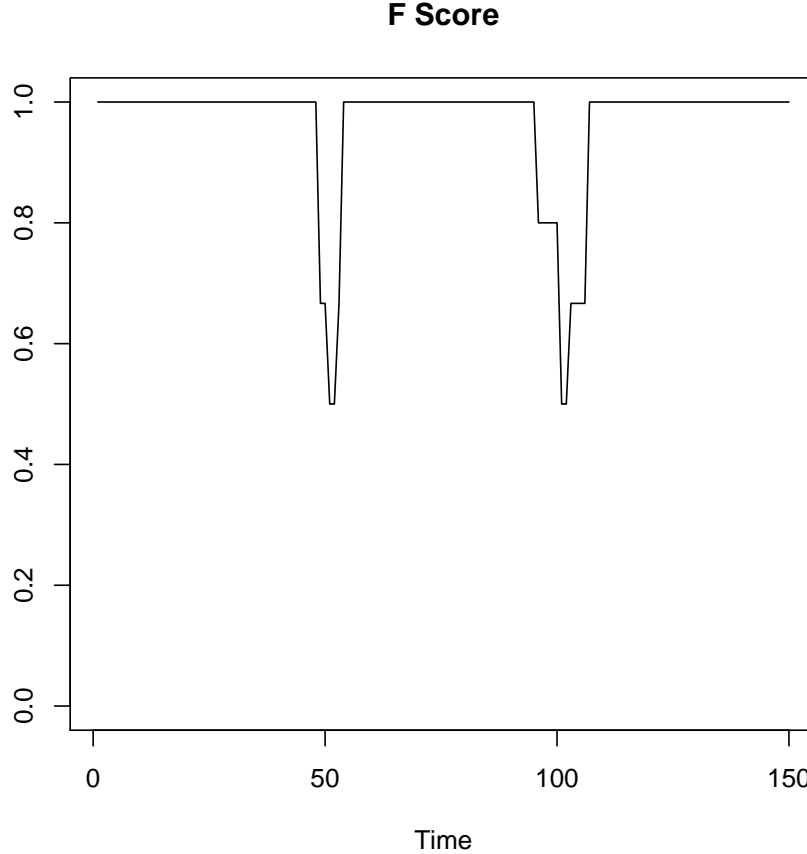


Figure 2: Plot of the F score over time for the SINGLE algorithm on simulated data

3 Background

We assume to have obtained time series denoted by X_1, \dots, X_T , where each vector $X_i \in \mathbb{R}^{1 \times p}$ contains the measurements for each of the p variables of interest at the i th time point. We are interested in inferring a sequence of graphs $\{G_1, G_2, \dots, G_T\}$ where each $G_i = (V, E_i)$ corresponds to the functional connectivity between nodes, V , at time i . The edge structure, E , is determined using partial correlations.

The SINGLE algorithm has the following two desirable properties: (a) each graph G_i is sparse thus allowing for accurate and interpretable estimates of the underlying graphs, and (b) the structure of the estimates graphs varies smoothly over time¹.

At any given time point, we assume that the random vector X_i follows a multivariate Gaussian distribution, however both the mean and the covariance of this distribution are

¹we note that the extent of sparsity and smoothness can be determined by the user

assumed to be dependent on the time index. We write S_i to denote the estimated covariance matrix at time i .

The set of partial correlations is summarised in the precision (inverse covariance) matrix. Thus our objective is equivalent to obtaining a sequence of time-dependent estimates of precision matrices, $\hat{\Theta}_1, \dots, \hat{\Theta}_T$, where each $\hat{\Theta}_i = S_i^{-1}$. The Smooth Incremental Graphical Lasso (SINGLE) obtains these estimates by solving a constrained optimisation problem which balances goodness-of-fit, sparsity and temporal smoothness. This is achieved by formulating the following objective function:

$$f(\{\hat{\Theta}\}) = \sum_{i=1}^T \left[-\log \det \hat{\Theta}_i + \text{trace} (S_i \hat{\Theta}_i) \right] + \lambda_1 \sum_{i=1}^T \|\hat{\Theta}_i\|_1 + \lambda_2 \sum_{|i-j|<k} \|\hat{\Theta}_i - \hat{\Theta}_j\|_1, \quad (1)$$

where $\{\hat{\Theta}\} = \{\hat{\Theta}_1, \dots, \hat{\Theta}_T\}$ contains all the precision matrices indexed by time. By taking a closer look at the objective function we can gain a clear understanding of what the SINGLE algorithm is looking to achieve. The first sum is proportional to the sum of negative log likelihoods of the estimated precision matrices. The first penalty term regularised by λ_1 corresponds to the Graphical Lasso penalty and ensures sparsity in the estimated graphical structure by imposing a penalty on the sum of absolute entries in each $\hat{\Theta}_i$. On the other hand it, the second penalty function regularised by λ_2 , ensures smoothness by penalising the differences between temporally adjacent networks. This penalty can be seen as an extension of the Fused Lasso penalty from the context of penalised regression (i.e., in the Fused Lasso we penalise $|\beta_i - \beta_{i+1}|$ and here this is extended to the difference over graphs). It is the choice of parameter k that governs the extent over which we expect there to be smoothness in the functional connectivity structure over time. It follows that a large choice of k will penalise changes over a longer period of time and thus result in *smoother* estimates.

The SINGLE estimation procedure consists of two independent steps performed in sequence: initially, recursive estimates of the covariance matrices S_1, \dots, S_T are obtained using a Gaussian kernel; then an iterative optimisation algorithm is run until convergence in order to produce a sequence of estimated graphs. Full details of the SINGLE algorithm are given in [4].

4 The SINGLE package

In this section we give a more detailed description of each of the functions contained in the SINGLE package.

4.1 generate_random_data

This function allows for the generation of data with a random correlation structure that is piecewise continuous. This is achieved using Vector Autoregressive Processes (VAR) and Erdos-Renyi random graphs. The choice of VAR processes here is motivated by their ability to encode autocorrelation within a time series as well as cross-correlations across time series.

First a random correlation structure is simulated using Erdos-Renyi random graphs. This is then used to simulate a VAR process with the given correlation structure.

As shown in the motivational example, we can easily simulate data using this function:

```
> sim = generate_random_data(ROI=5, length=50, seg=3, sparsity=.15)
```

We note that while the number of piecewise continuous segments of data and their lengths can be specified using `seg` and `length_` respectively. Random data with segments of different lengths can be generated using the `generate_random_data` command several times as follows:

```
> sim1 = generate_random_data(ROI=5, length_=25, seg=1, sparsity=.15)
> sim2 = generate_random_data(ROI=5, length_=75, seg=1, sparsity=.15)
> data = rbind(sim1$data, sim2$data)
```

This function depends on the `igraph` [1] and `dse` packages [2].

4.2 SINGLE

The `SINGLE` function is the main function within the `SINGLE` package and provides an implementation of the `SINGLE` algorithm. We note that this function requires the input of the following user-specified parameters:

1. **radius**: This is the radius, h , used in the Gaussian kernel. The choice of this radius is particularly important. Setting h to be too large will result in overly smooth estimates whereas choosing h to be too small will result in noisy estimation. Intuitively it is advised to set h to be as small as possible without compromising the stability of estimated covariances S_1, \dots, S_T . Ultimately, the choice of h should depend on the dimensionality of the data, p , and the rate of change of the underlying networks, often referred to as the drift speed. Assuming the drift speed is sufficiently slow, setting $h = p$ can be used as a rule of thumb. However, if this is not the case then reducing the value of h will improve performance.
2. **l1**: This is value of λ_1 in the `SINGLE` objective function. Increasing the value of λ_1 will increase the sparsity of the estimated dynamic networks.
3. **l2**: This is the value of λ_2 in the `SINGLE` objective. Increasing the value of λ_2 will encourage sparsity in the difference of temporally adjacent networks, thus resulting in *smoother* estimates.
4. **k**: This determines the extent of the λ_2 penalty in the `SINGLE` objective and can be interpreted as the extent over which we expect there to be smoothness in network structure over time.

As shown in Section 2 the `SINGLE` algorithm can be run as follows:

```
> data = sim$data
> S = SINGLE(data=data, radius=5, l1=.75, l2=0.3, k=3)
```

Since the `SINGLE` algorithm relies on the Fused Lasso for one step of its iterative optimisation this function depends on the `flsa` package [3].

4.3 precision_recall

The `precision_recall` function allows us to measure the performance of the `SINGLE` algorithm. Precision is defined as the fraction of reported edges which are true edges while recall is defined as the fraction of true edges which are correctly reported. It follows that the closer

precision and recall are to 1 the better the performance. Finally we also define the F score as follows:

$$F = 2 \cdot \frac{P \cdot R}{P + R},$$

where P and R refer to the precision and recall respectively.

This function calculates the precision and recall for an iteration of the SINGLE algorithm at each observation. That is, at the i th observation the precision, recall and F score is calculated by comparing the true correlation structure at the time and the correlation structure estimated by the SINGLE algorithm.

The `precision_recall` function is run directly on the array of estimated precision matrices (`S$P_` in the example above) as follows:

```
> result = precision_recall(true_cov=sim$true_cov, estimated_cov=S$P_)
```

4.4 plotSINGLE

The `plotSINGLE` function allows for the visualisation of the estimated correlation structures over time. This is achieved by plotting the pairwise partial correlations between each of the nodes. Since there may be a potentially large number of nodes (resulting in an even larger number of pairwise correlations to plot) the `plotSINGLE` function allows the user to specify a subset of nodes to plot using its `index` command. In the example from before we plot the partial correlations for all nodes. Alternatively, we could plot only the partial correlations between the 1st, 3rd and 5th nodes as follows:

References

- [1] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.
- [2] P. D. Gilbert. *Brief User's Guide: Dynamic Systems Estimation distributed with the dse package*, 2006.
- [3] H. Hoefling. *flsa: Path algorithm for the general Fused Lasso Signal Approximator*, 2013. R package version 1.05.
- [4] R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana. Estimating Time-varying Brain Connectivity Networks from Functional MRI Time Series. *Preprint*, 2013.

```
> plotSINGLE(object=S, index=c(1,3,5), x.axis = seq(1,150), n.row=1,
+           col.names=c(1,3,5), fix.axis=TRUE)
```

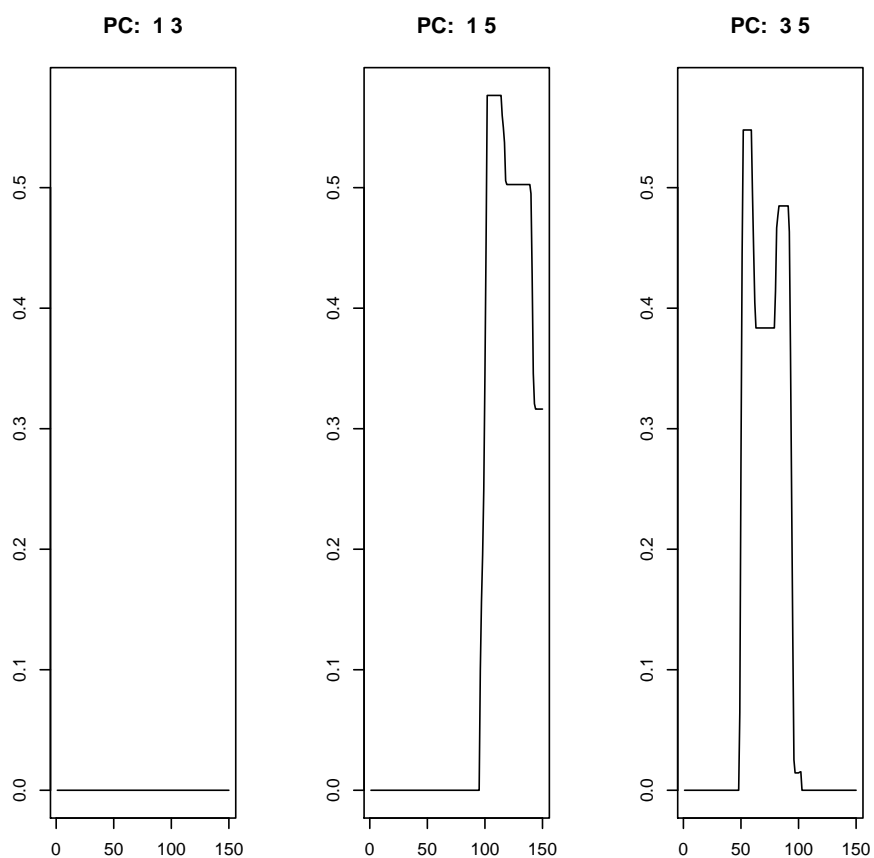


Figure 3: Estimated partial correlations for simulated example discussed previously