Symptom network models in depression research

van Borkulo, Claudia Debora

APPENDIX D

A TUTORIAL ON R PACKAGE ISINGFIT

Adapted from:

https://cran.r-project.org/web/packages/IsingFit/IsingFit.pdf

This tutorial provides and explanation of R package IsingFit (Van Borkulo, Epskamp, & Robitzsch, 2016). IsingFit fits Ising models (Ising, 1925) using the *eLasso* method. eLasso combines ℓ_1-regularized logistic regression with model selection based on the Extended Bayesian Information Criterion (EBIC). EBIC is a fit measure that identifies relevant relationships between binary variables. The resulting network consists of variables as nodes and relevant relationships as edges. In this tutorial, we will provide a short introduction to *eLasso* — please see Chapter 4 and Appendix A for more detailed information — and illustrate how `IsingFit()` can be used, what options there are and how applying different options affect results. To illustrate this, we provide examples with real data and provide the accompanying R code. The various arguments of `IsingFit()` are explained and the impact of adjusting the arguments is illustrated with data of the Virginia Adult Twin Study of Psychiatric and Substance Use Disorders (VATSPUD; Kendler & Prescott, 2006; Prescott et al., 2000). For this tutorial, we use the data of presence/absence of the 14 disaggregated symptoms of MDD for at least 5 days during the previous year of 8973 individuals from the population as used in Chapter 3 of this dissertation. This Appendix is an extended version of `https://cran.r-project.org/web/packages/IsingFit/IsingFit.pdf`.

D.1 Introduction

The edges in a network represent conditional dependencies: if there is an edge between symptom X_1 and X_2, they are related even after conditioning on (or controlling for) all other variables in the network. This means that the relationship between X_1 and X_2 cannot be explained by any of the other variables. It is not a spurious relationship that arose because the two are related through another (third) confounding variable in the dataset. Conversely, if two variables are not connected, say X_2 and X_3, this means that they are not related. Not *directly* related, to be more specific. They might be correlated, but only because they share connections to other variables in the network. When conditioned on all other variables, the relationship between X_2 and X_3 disappears — it is explained away by the other variables. The network representing conditional dependencies is also called a *Markov Random Field* (Kindermann & Snell, 1980; Lauritzen, 1996).

For continuous data, conditional dependencies can be established by means of the covariance matrix of the observations of the variables. A zero entry in

the inverse of the covariance matrix represents a conditional independence between the focal variables, given the other variables (Speed & Kiiveri, 1986). The simplest model that can explain the data as adequately as possible according to the principle of parsimony, can be found with different strategies to find a sparse approximation of the inverse covariance matrix. One of the strategies to find such a sparse approximation is to impose an ℓ_1-penalty (also called lasso) on the estimation of the inverse covariance matrix (Foygel & Drton, 2010; Friedman et al., 2008; Ravikumar et al., 2011). The ℓ_1-penalty guarantees shrinkage of partial correlations and puts others exactly to zero (Tibshirani, 1996). Another strategy entails estimating the neighborhood of each variable individually with ℓ_1-penalized regression (Meinshausen & Bühlmann, 2006), instead of imposing an ℓ_1-penalty on the inverse covariance matrix and is, therefore, an approximation to the ℓ_1-penalized inverse covariance matrix. This approximation method using ℓ_1-penalized regression is an interesting alternative — it is computationally efficient and asymptotically consistent (Meinshausen & Bühlmann, 2006) — that can also be applied to binary data and is implemented in IsingFit.

D.2 Arguments

The main function of package IsingFit is function `IsingFit()`. To run `IsingFit()`, the only input that is required is the data set. This results in output according to default settings of several arguments:

```
IsingFit(x, AND = TRUE, gamma = 0.25, plot = TRUE, progressbar
= TRUE,
lowerbound.lambda = NA, ...)
```

x

The first argument (x) — and the only one you have to enter to run the default function — is the data. This must be a matrix in which each variable is represented by a column and each observation by a row. Observations should be independent. An example of a data set with independent observations is one in which rows contain scores on items of a questionnaire of different individuals. Whether person *i*

(on row i) has a set of scores on items does not depend on the scores of person j (on row j).

The following code is used to estimate a default network with our example data, in which `Data` is the name of the VATSPUD dataset and the output of `IsingFit` is stored under the name `Res`.

```
Res <- IsingFit(Data)
```

The estimated network structure is shown in Figure D.1, which is simular to the empirical network in Figure 3.3, Chapter 3.

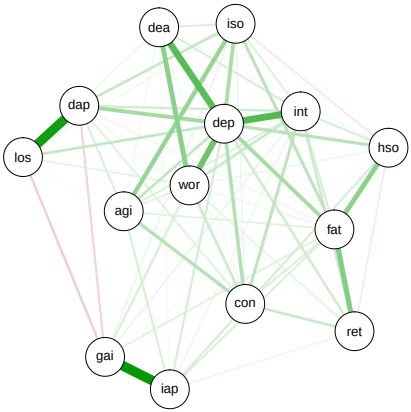

FIGURE D.1. The estimated network structure based on the VATSPUD data, with the default setting of `IsingFit()`. dep - depressed mood; int - loss of interest; los - weight loss; gai - weight gain; dap - decreased appetite; iap - increased appetite; iso - insomnia; hso - hypersomnia; ret - psychomotor retardation; agi - psychomotor agitation; fat - fatigue; wor - feelings of worthlessness; con - concentration problems; dea - thoughts of death.

AND

This is a logical indicating whether the AND-rule is used or not. When `AND = TRUE`, the AND-rule is applied, requiring both regression coefficients β_{jk} and β_{kj} to be nonzero to result in an edge between node j and k (see Chapter 4 and

Appendix A for an extensive explanation of the estimation procedure). A more lenient option is to set the AND argument to `AND = FALSE`, thereby applying the OR-rule, requiring only one of β_{jk} and β_{kj} to be nonzero. For our dataset there is no observable difference in the resulting network structure when applying the AND- or OR-rule. This is because of the high power due to high sample size ($n = 8973$). When the power is not high enough to estimate a reasonable network structure — due to an unfavorable ratio of number of nodes and sample size — one might use the OR-rule. Note that this might result in more spurious connections (false positives).

In Figure D.2, we show the result of applying the AND and OR-rule when power is low. As can be seen in the code below, we used a subsample of our original VATSPUD dataset in which only the first 150 individuals are included.

```
Res.150.AND <- IsingFit(Data[1:150,], AND=TRUE)
Res.150.OR <- IsingFit(Data[1:150,], AND=FALSE)
```

As you can see in Figure D.2, there are a few more edges retrieved when the OR rule is applied (right panel), compared to when the AND rule is applied (left panel).

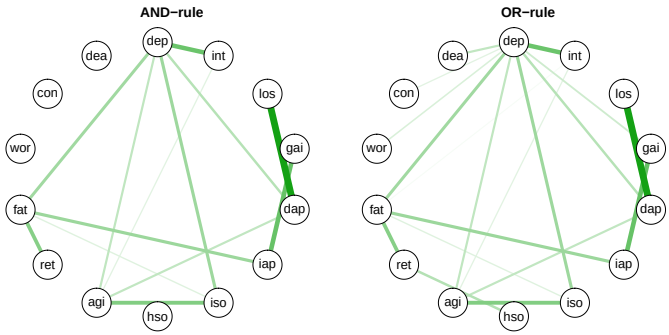

FIGURE D.2. Network estimation with AND- (left panel) and OR-rule (right panel). Only the first 150 observations are included in the network estimation procedure.

gamma

With this argument, you can adjust the value of hyperparameter γ. Hyperparameter γ plays a role in Goodness-of-Fit measure EBIC (equation A.10) with which the optimal tuning parameter (i.e., ρ which represents the best set of neighbors of the focal node) is selected (again, see Chapter 4 and Appendix A for an extensive explanation of the estimation procedure of IsingFit). In essence, γ determines the strength of prior information on the size of the model space. It penalizes the number of nodes in neighborhood selection and puts an *extra* penalty on the number of neighbors (see equation A.10). Note that — and this is often misunderstood — when γ is set to zero, there will still be regularization and the number of neighbors is still penalized. However, instead of an extra penalty on the size of the model space, the optimal tuning parameter is now selected with the ordinary BIC instead of the *extended* BIC.

By increasing γ, the strength of the *extra* penalty on the size of the model space increases. A lower value of γ — maybe even zero — could be advantageous when the power is low and applying EBIC results in very few edges or none at all. A higher value of γ could be favorable when you have high power (a good number of nodes to observations ratio) and want the least number of false positives.

To illustrate the effect of different values of γ, we used the subsample of 150 individuals and estimated network structures with $\gamma = 0, .3, .6, 1$. As you can see in Figure D.3, the network based on $\gamma = 0$ (most left panel) is least sparse (most connections) and $\gamma = 1$ (most right panel) is most sparse. The connections in the latter network have the highest probability of being true positives: they are still present, while being estimated with a high value of γ (i.e., a high *extra* penalty on the number of neighbors).

plot

A logical indicating whether the estimated network should be plotted.

progressbar

Logical. Should the progressbar be plotted in order to see the progress of the estimation procedure? Especially with a large data set, it may take a while before

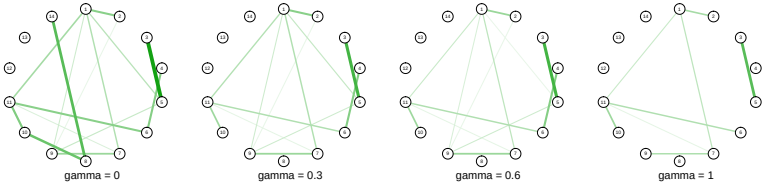

FIGURE D.3. Networks estimation based on the first 150 observations of the VATSPUD data with increasing values of γ (from left to right).

all node-wise regressions are performed. With the progressbar, you can keep track of the progress of the estimation procedure. Defaults to TRUE.

lowerbound.lambda

The minimum value of tuning parameter lambda (regularization parameter). Can be used to compare networks that are based on different sample sizes. The lowerbound.lambda is based on the number of observations n in the smallest group: $\sqrt{\frac{\log p}{n}}$, where p is the number of variables, that should be the same in both groups. When both networks are estimated with the same lowerbound for lambda (based on the smallest group), the two networks can be visually compared. For a statistical comparison, NCT can be used (see Chapter 5 and Appendix E for a tutorial on the NCT package).

D.3 Output

Function `IsingFit` returns an *IsingFit* object and contains several items. Below follows a description of each item.

weiadj

The weighted adjacency matrix. This contains the edge weights. You can use this item to plot the network with `qgraph` and adjust the plot as you like. Below you can find the code to use the output of `IsingFit` to make your own plot.

```
qgraph(Res$weiadj, layout='circular')
```

TABLE D.1. Thresholds of the symptoms based on the VATSPUD data with default setting of `IsingFit`. See Figure D.1 for abbreviations.

symptom	threshold
dep	-2.31
int	-3.19
los	-4.31
gai	-3.83
dap	-3.92
iap	-3.9
iso	-3.02
hso	-4.45
agi	-3.18
ret	-4.34
fat	-2.83
wor	-4.43
con	-4.04
dea	-5.83

thresholds

A vector that contains the thresholds of the variables. The threshold of a variable represents the autonomous disposition to be *present*. For the network in Figure D.1 based on the whole sample, the thresholds are all negative (see Table D.1). This means that all symptoms have an autonomeous disposition to be *absent*. When zooming in on specific symptoms, "depressed mood" (dep) has the highest threshold (-2.31). This means that this symptom has the highest probability of being *present* in the sample compared to the other symptoms. "Thoughts of death" (dea) has the lowest threshold and, consequently, has the lowest probability of being *present* in the sample.

q

The object that is returned by `qgraph` (class *qgraph*).

gamma

The value of γ that was applied.

AND

A logical indicating whether the AND-rule is applied. If `FALSE`, the OR-rule was applied.

time

The time it took to estimate the network.

asymm.weights

The asymmetrical weigthed adjacency matrix before applying the AND/OR-rule. These are the *original* regression coefficients from the nodewise regressions (see Section 4.2.1).

lambda.values

The values of the tuning parameter per node that ensured the best fitting set of neighbors.