Network and Community Detection Visualization

(A) Introduction

Network science has recently overtaken psychological research (Fried & Cramer, 2017). Not only has psychological network theory inaugurated a new way of conceptualizing psychometric phenomena (such as the positive manifold; Van der Maas, Kan, Marsman, & Stevenson, 2017), network models also offer a novel way to estimate and visualize associative patterns in data. Many researchers have made use of these opportunities with the help of the qgraph and igraph packages; however, several researchers might have been left behind, as being able to code in R is a prerequisite to apply the network methodology. Therefore, this very app was created for two reasons: 1. Make network estimation and visualization possible by a click and 2. implement a novel function in the qgraph package: detection and visualization of network communities.

The **Network Community Detection Visualization App** is the first app which implements community detection and visualization in qgraph. Community detection and visualization is a helpful tool for the psychological network researcher. Network communities refer to groups of nodes in a graph that are more strongly and/or densely connected with each other than with other nodes in the graph (Fortunato, 2010). It follows that nodes in a community should affect each other more than nodes belonging to different communities. For instance, a psychometrician might use community detection to detect and visualize groups of nodes (i.e. items) that represent sub-components in a larger network, such as personality-trait communities in a personality network. Furthermore, a clinical psychologist might be interested to detect and visualize communities in psychopathology networks in order to see how symptoms cluster together and which group of nodes are more and less likely to affect each other. The clinical psychologist might then want to base interventions on these insights (e.g. choose which community to intervene on first).

The text below demonstrates what functions the **Network Community Detection Visualization App** provides and how users can interact with the app.

(B) Walk-Through

(1) Starting the app

Currently, the app is only accessible through running the shiny code in R. You can download the ui.R (code for the user interface) and the server.R (code for functionality) scripts from my GitHub page (Note: **evaluators** for the **Programming: the next step course** can find both files in the Zip File in which this document was saved).

Save both scripts in the same directory (e.g. in their own folder) and open them in Rstudio (if you do not have R or Rstudio installed, you would need to do that first; first install R then install RStudio). If you have both up and running, you can open both the ui.R and server.R script in Rstudio. Make sure to set the working directory in RStudio to the directory in which you saved the server.R and ui.R scripts.

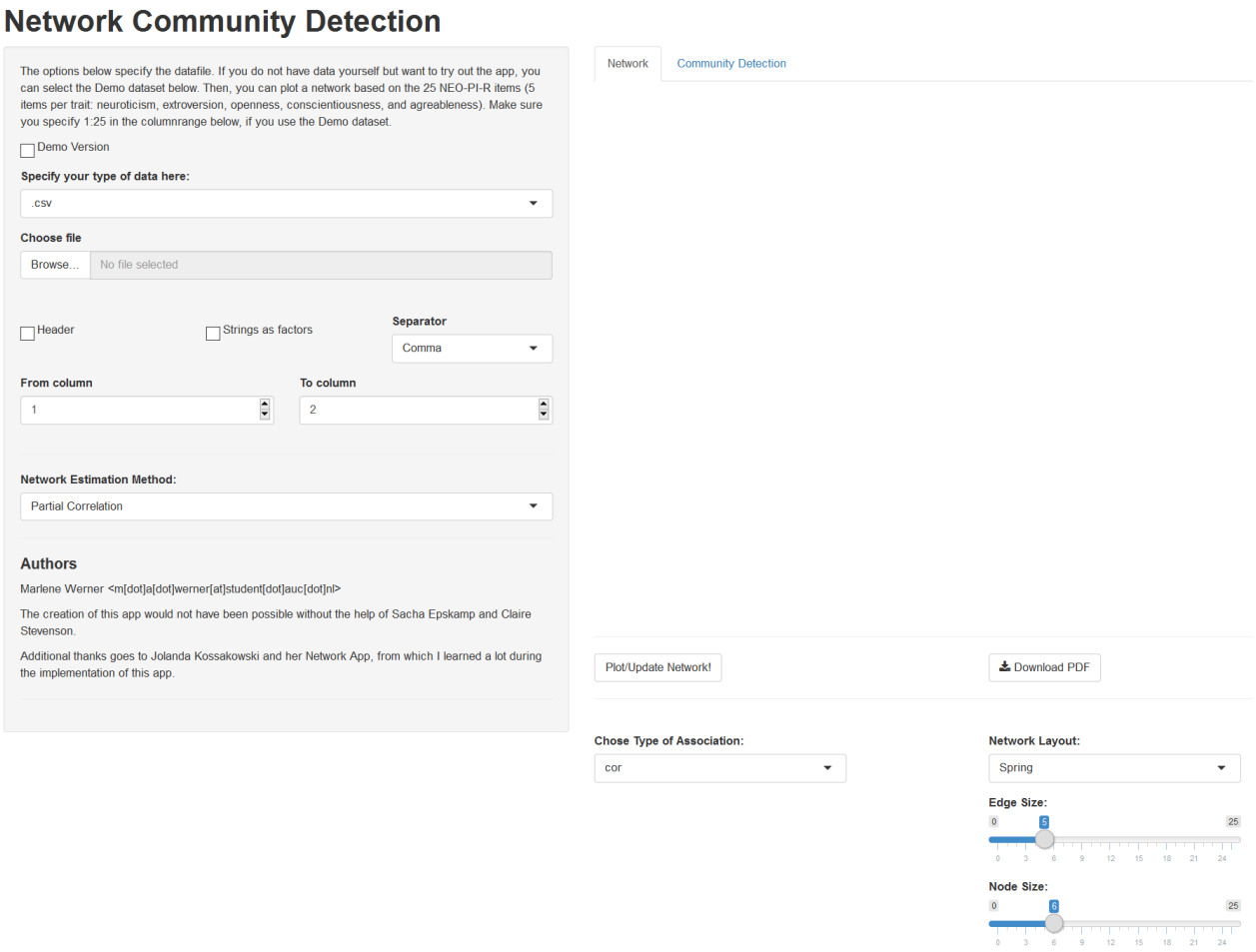
To run the app, you need to have the following packages installed:

1. "shiny"
2. "psych"
3. "qgraph"
4. "bootnet"
5. "igraph"
6. "scales"
7. “foreign”
8. “readxl”

In order to install them, run: **install.packages("nameOfPackage")**.

If you have all dependencies installed, you press the button "Run App" above the script screen in either the server.R or ui.R script.

The app should open in a browser window and look like this:



The app is separated into four different panels. On the left, you find the main panel where data are specified and the network estimation method is chosen. To the right of the main panel, you can find the plotting area. Underneath the plotting area, there is another panel. In this panel, the estimation and layout options are shown; this sub-panel changes depending on which estimation method is chosen. Finally, there is a tab panel above the plotting area. The network tab has to be used before the community detection tab; in the network tab, the network is estimated, which can then be used in the community detection tab. In the community detection tab the sub panel underneath the main panel will show different options. Further details about these and all panel use can be found beneath.

(2) Specifying data

Own Data

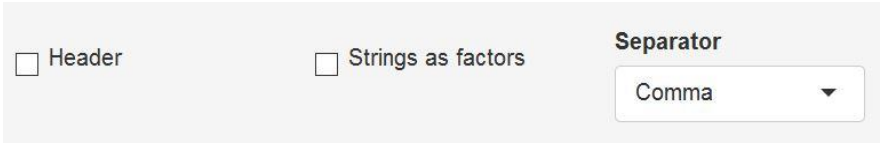
The network app requires data to plot a network. Currently, the app can only handle raw data (i.e. no edge lists or adjacency matrices) saved in **.csv** format only. Data can be uploaded in the left sidebar. You will see an error message in case the data set cannot be transformed into a data frame or includes variables other than numeric, ordered or integer (these variables will be dropped from the estimation).

DISCLAIMER: *The manual is still written such as if all data formats were supported. Unfortunately, I could not get the data upload to work for all data formats until the deadline. .csv upload works. Furthermore, the demo version works as well. I will debug the other upload functions in the future.*

First, specify the kind of format your data is stored in:

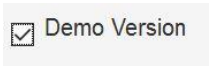
Second, look for your data file and upload it:

Third, specify whether the first row represents the variable names, whether character variables should be coded as factors (i.e. if you have saved variables with names instead of numeric values, this box should be checked), and what kind of symbol represents the separation between variables in the data set:



Demo Version

In case, you do not have data at hand, you can also **try out the app with a build-in data set**. Just **check the demo box** at the top of the sidebar. When you are using this data set, **no data options need to be specified** (i.e. header, strings as factors, separator):



The data comprise the first 25 variables of the NEO-PI-R, which measures 5 personality traits: neuroticism, extroversion, openness, conscientiousness and agreeableness (psych package).

Select Variables

Next, you can select which variables (i.e. columns) should be included in the network. This option is available for the demo and uploaded data sets:

(3) Select estimation method

Then, you select which method should be used for estimating (and regularizing) the associations:

There are currently 5 methods available:
1. Partial correlations
2. GLASSO estimation
3. Ising Sampler

4. Ising Fit
5. Huge

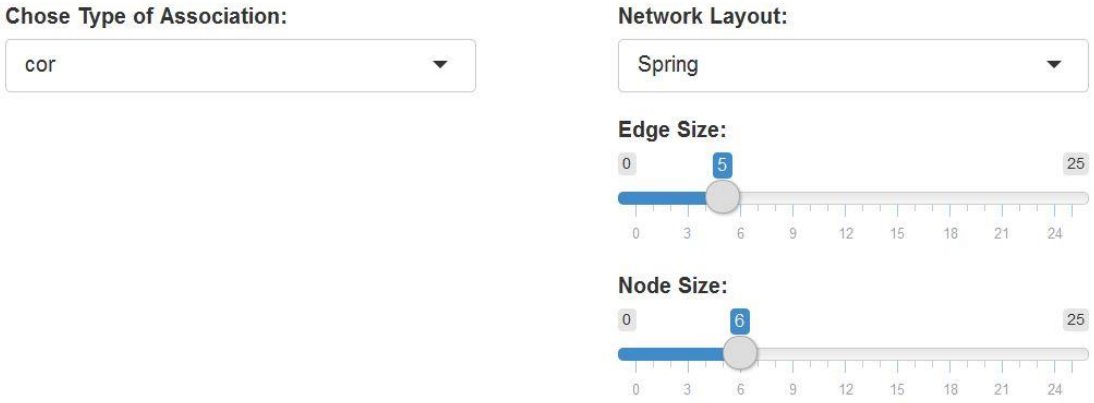
See the estimateNetwork function in the bootnet documentation for an explanation of these methods.
Important to note is that the app currently only handles cross-sectional (i.e. independent cases) data. Use partial correlations or GLASSO estimation if your variables have 5 or more values; only use GLASSO estimation if you want your network to be estimated such that potentially spurious (false positive) relationships are set to zero. We set the tuning hyperparameter to 0.5 by default; 0.5. has been shown to minimize the number of spurious connections while maximizing the number of true connections (Epskamp & Fried, 2017). Use IsingSampler (nonregularized) or IsingFit (regularized) if your data are binary. Use Huge if your data are non-normally distributed and if you do not want to use polychoric correlations (as in partial correlation and GLASSO networks).

(3.1) Estimation and Plotting Options

Depending on the estimation method, the panel underneath the plotting area will change and display different options. After you have specified these options, you can hit the **"Plot/Update Network"** button to plot the network.

In the case of "Partial Correlations" and "GLASSO estimation" the network changes dynamically after plotting the first network (i.e. you do not have to hit the "Plot/Update Network" button after you have **re**specified an option). "Ising Sample", "IsingFit", and "Huge" require updating every time you change an option. We have decided to delay estimation for these methods, because their estimation takes longer than the estimation in "Partial Correlations" and "GLASSO". If we re-estimated the network every time you changed an option in these methods, the R-workflow could bug.
Follow the specifications below to estimate networks according to the method of your choice:

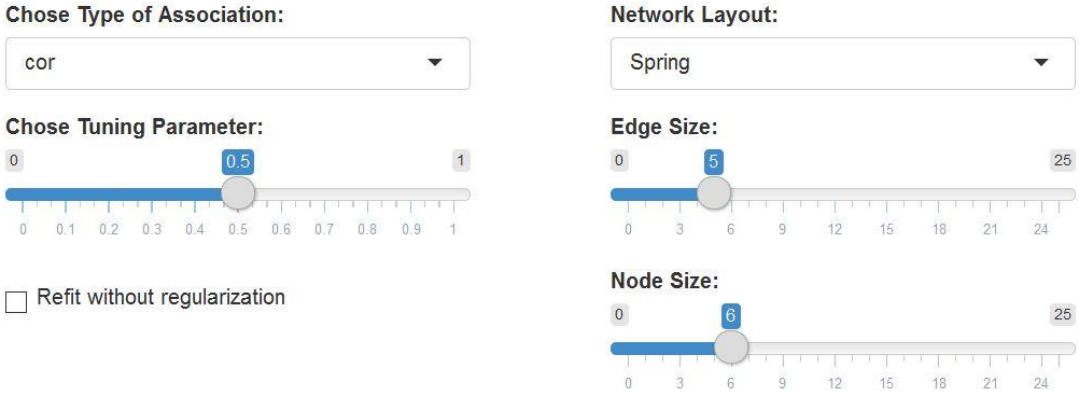
a. Partial Correlations



First, you specify the type of partial association (left column of sub panel). You can choose between several different types. "cor_auto" determines the best type of association for the data at hand. "cor" calculates Pearson correlations. "cov" uses the covariation and "npn" applies a nonparanormal transformation to the data and then calculates correlations.
Second, you can select the layout of your network (right column). Currently, the app supports the circle and spring layout. The spring layout positions the nodes according to a force-dependent algorithm (Fruchterman-Reingold) that determines the distance and position of nodes by means of their association strength.

Finally, you can select the size of the edges and nodes.
The network changes dynamically, if you change an option.

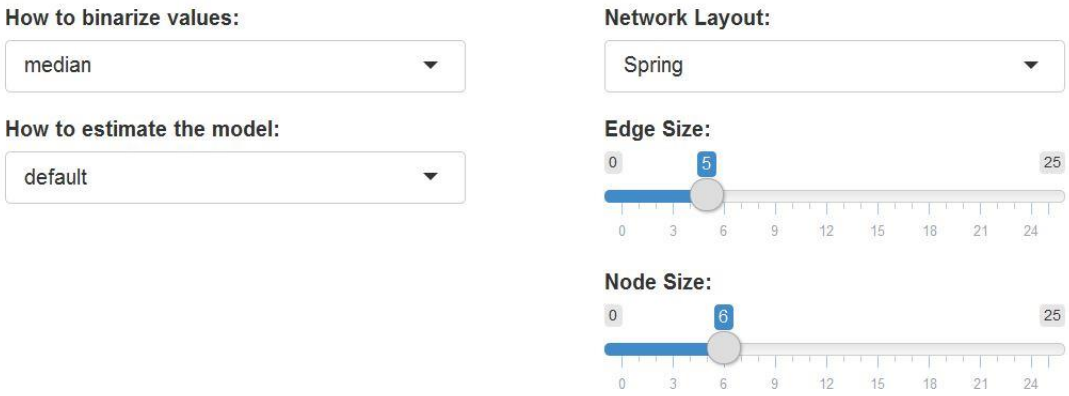
b. GLASSO



In GLASSO estimation, you also need to specify the type of association (left column). Second, you chose the level of regularization (the higher the tuning parameter, the sparser the network). If you wish, you can refit the network without regularization by merely checking the box underneath the tuning slider. Finally, you can choose from the same layout options as for the partial correlation network.
The network changes dynamically, if you change an option.

c. Ising Sampler

IsingSampler can estimate the associations through various methods: 1. Maximization of the pseudolikelihood 2. Univariate logistic regressions of each node on all other nodes 3. Multinomial logistic regression of each pair of nodes on all other nodes 4. Loglinear modeling with at most pairwise interactions.



You can select the method by clicking on the "How to estimate the model widget". In case you have non-binary data but want to estimate an Ising Model, you can choose between mean or median binarization using the widget "How to binarize values". Finally, you can define the layout as for the methods before.
Click **Plot/Update Network** as soon as you specified all options or want to update after re-specifying options.

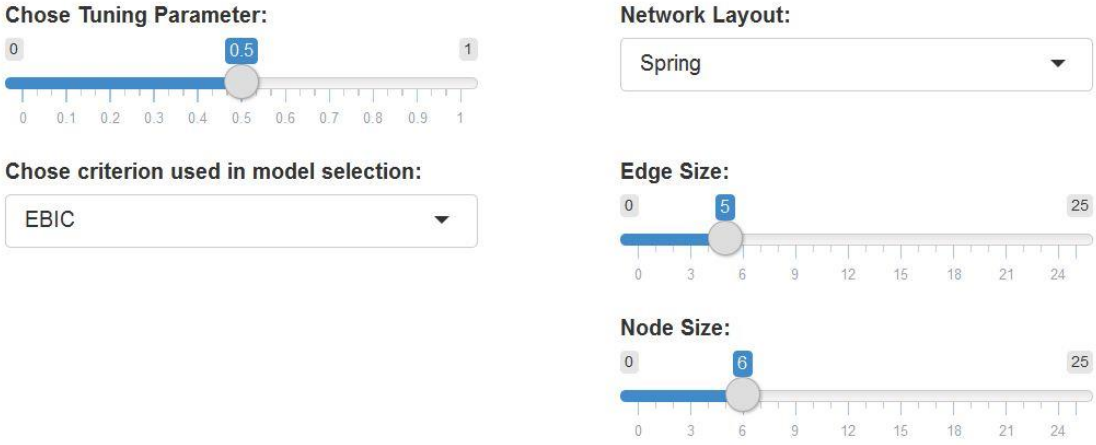
d. Ising Fit

IsingFit estimates relationships through l1-regularized logistic regression with model selection based on the Extended Bayesian Information Criterion (EBIC).

First, you need to specify how data should be binarized. Second, you need to choose the rule that selects an edge in nodewise estimation. Select "AND" to include an edge if both regression coefficients are nonzero and "OR" if only one is nonzero. Third, specify the hyper(EBIC)-tuning-parameter. Fourth, you can select the network layout options.
Click **Plot/Update Network** as soon as you specified all options or want to update after re-specifying options.

e. Huge

In case your data is non-normally distributed, you can estimate associations via the huge package. A nonparanormal transformation is applied to the data before partial correlations are estimated.

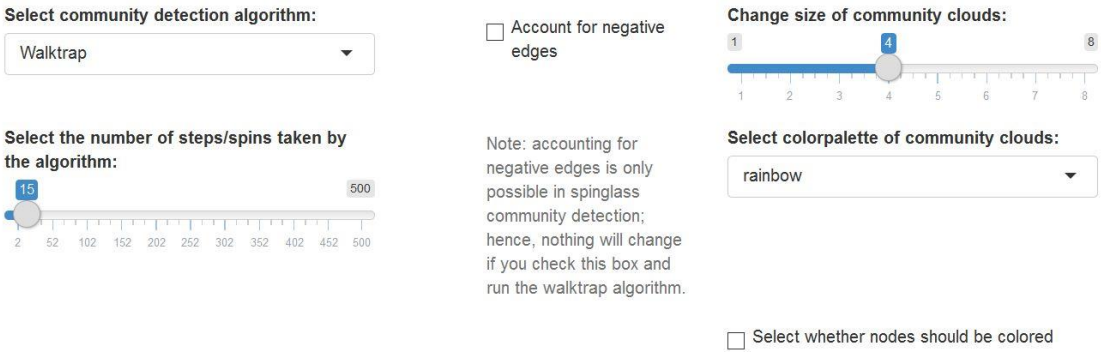


First, specify the level of the hyper-tuning-parameter. Second, specify which model selection criterion should be used. Third, specify the network layout options.
Click **Plot/Update Network** as soon as you specified all options or want to update after re-specifying options.

(4) Community Detection

After you have successfully estimated and visualized your network, you can switch to the "Community Detection Tab" by clicking on the tab above the plotting area.

The plotting window will clear and a new sub panel with different options will appear.



(4.1) Community Detection

Networks tend to show particular topographical layouts (Barabási, 2016). For instance, networks can be clustered more strongly or loosely depending on how strongly nodes associate with each other. Furthermore, sub-groups of nodes can cluster together more strongly among each other than with other nodes in the network. Such strongly clustered groups of nodes are called communities (Fortunato, 2010). Often, such communities can be detected visually in a network. However, statistical community detection algorithms can determine whether a clustered group represents a community in statistically replicable terms (Fortunato & Hric, 2016). There are several kinds of community detection algorithms. The app includes two options. These two options, called **walktrap** and **spinglass community detection**, were chosen because they have most frequently been applied in psychological research and are able to take edge weights into account when determining the community structure (Heeren & McNally, 2016; Knefel, Tran, & Lueger-Schuster, 2016; Fried, 2016).

Walktrap Community Detection

Walktrap detects communities through random walks across the network structure (Pons & Latapy, n.d.); steps should get "trapped" in a community and all communities should get detected if several random starting points of a walk are taken. You can specify the number of steps the algorithm takes by changing the slider on the left.

Spinglass Community Detection

Spinglass detects communities through simulations that follow the Potts-model from statistical physics (Reichardt & Bornholdt, 2006;Traag & Bruggeman, 2008). The simulation determines whether any two nodes tend to be in a similar state which is more likely for nodes that belong to one community. You can specify the number of spins the algorithm takes by changing the slider on the left. The spinglass algorithm can take negative edges into account; by so doing, it diminishes the likelihood that two nodes that are negatively related belong to the same community (i.e. negative edges are more likely to exist between communities). Only use this option if that makes sense for your topic of research.

(4.2) Layout Options

You can change the size of the enclosing clouds by changing the slider on the right of the panel. Moreover, you can specify the color palette the clouds will use. You can choose among: "rainbow", "heat.colors", "cm.colors", "topo.colors", "terrain.colors", "rainbow_hcl", "diverge_hcl", "terrain_hcl", "sequential_hcl".
See the R-Color-Cheatsheet for a preview of these palettes.

In case, your network is strongly clustered overall, we suggest to check the box "Select whether nodes should be colored". Nodes will be set to different gray scales depending on their community membership. Setting off the nodes in this additional way makes it easier to see which nodes belong to which community in case the clouds overlap to a large degree and enclose nodes that do not belong to the same community.

Hit **"Plot/Update Communities"** after you have specified all options. In case you change one of the options, you need to press this button again:

(5) Download Networks

In case you want to download your plot, you can click the button **Download PDF** to save a pdf with the network on your computer.

DISCLAIMER: *Unfortunately, the download button for the network visualization did not work until the very last day, which is why I did not work further on the download button for the community plots. I thought that if I got the network download button to work, it would be easy to translate that to the community plot button. However, that was not the case (I will discuss why in section* ***(7) Future Possibilities)****. It is no problem to download the plots in RStudio itself.*