

# User's Manual

*Jolanda J. Kossakowski, MSc*

*[mail@jolandakossakowski.eu](mailto:mail@jolandakossakowski.eu)*

## Contents

<b>Introduction</b>	<b>4</b>
<b>Uploading Data</b>	<b>4</b>
Data Types . . . . .	5
Missing Data . . . . .	6
Example Data . . . . .	6
<b>Network Analysis</b>	<b>6</b>
Non-Paranormal Transformation . . . . .	7
Network Estimation . . . . .	7
FDR Network . . . . .	8
GLASSO . . . . .	9
Graphical VAR: PCC . . . . .	9
Graphical VAR: PDC . . . . .	9
IC-Algorithm: DAG . . . . .	10
IC-Algorithm: Skeleton . . . . .	10
Multilevel VAR: 2-level . . . . .	11
Multilevel VAR: 3-level . . . . .	12
Partial Correlation . . . . .	12
Pearson Correlation . . . . .	13
VAR-model . . . . .	13
Small World Index . . . . .	13
Download Network . . . . .	14
<b>Network Esthetics</b>	<b>14</b>
General . . . . .	14
Title . . . . .	14
Network Layout . . . . .	14
Graph Details . . . . .	15
Node Esthetics . . . . .	15
Node Size . . . . .	15

Node Shape . . . . .	15
Node Labels . . . . .	15
Edge Esthetics . . . . .	16
Edge Size . . . . .	16
Edge Weights . . . . .	16
Directed Edges . . . . .	16
Plot Self-Loops . . . . .	16
Edge Threshold . . . . .	17
Minimum Edge Weight . . . . .	17
Maximum Edge Weight . . . . .	17
Edge Cut-Off Value . . . . .	17
<b>Centrality Analyses</b>	<b>18</b>
Centrality plot . . . . .	18
Standardized Centrality Measures . . . . .	18
Centrality Plot Layout . . . . .	19
Strength . . . . .	19
Betweenness . . . . .	19
Closeness . . . . .	19
In Strength . . . . .	19
Out Strength . . . . .	20
Download Centrality Plot . . . . .	20
Centrality table . . . . .	20
Download Centrality Table . . . . .	21
<b>Clustering Analyses</b>	<b>21</b>
Clustering Plot . . . . .	21
Standardized Clustering Coefficients . . . . .	21
Clustering Plot Layout . . . . .	21
WS . . . . .	21
Zhang . . . . .	22
Onnela . . . . .	22
Barrat . . . . .	22
Download Clustering Plot . . . . .	22
Clustering Table . . . . .	22
Download Clustering Table . . . . .	22

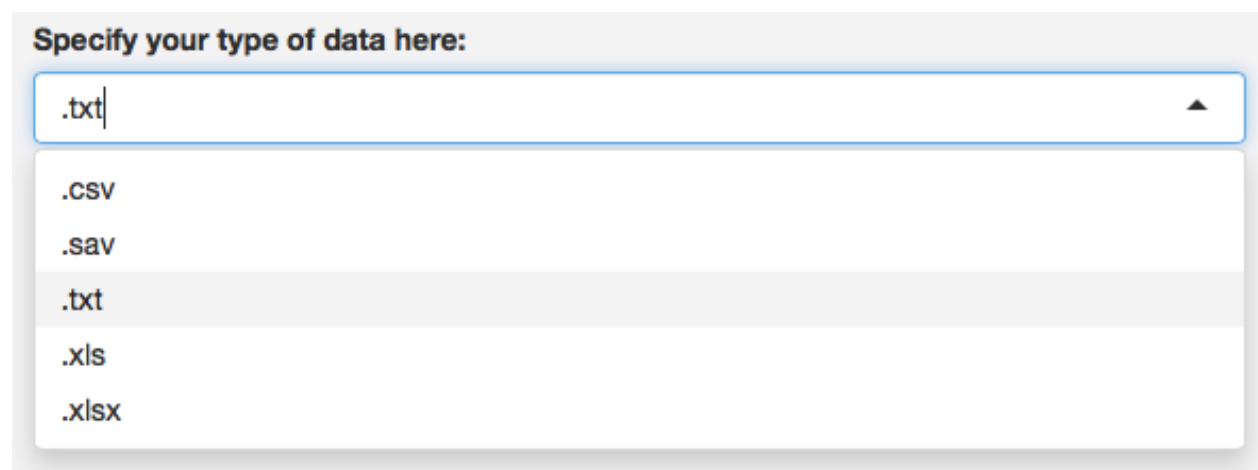
<b>Network Comparison</b>	<b>23</b>
Uploading Data . . . . .	23
Specifications . . . . .	24
Iterations . . . . .	24
Gamma . . . . .	24
Edge defining rule . . . . .	24
Weighted Networks . . . . .	27
Binary Data . . . . .	27
<b>References</b>	<b>28</b>

## Introduction

Network analysis has gained popularity in recent years. By the development of R-packages like *qgraph* (S. Epskamp et al. 2012) and *igraph* (Csardi and Nepusz 2006), it has become easier to visualize network structures and apply different estimation methods. The drawback of this is that users have to learn R in order to use these packages. The goal of this application was to design and programme a web application that enables users to visualize and analyze networks without having to know the R-language itself. This manual will aid users and explain all the functions and options that are currently available within the application.

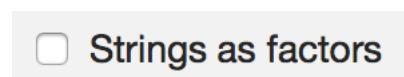
## Uploading Data

In the grey left panel of the application, users can upload either a “.txt” a “.csv”, a “.xls”, a “.xlsx” or a “.sav” file.



The image shows a web interface element titled "Specify your type of data here:". Below the title is a dropdown menu. The menu is currently open, showing a list of file extensions: ".txt", ".csv", ".sav", ".xls", and ".xlsx". The ".txt" option is currently selected and highlighted with a light grey background. The dropdown menu has a blue border and a small upward-pointing triangle on the right side of the input field.

Users only have to specify the type of data that they are uploading, how missing values, if present, are coded within the data file, and whether character variables (variables that contain words instead of numbers) should be coded as *factors*. In the case that the user discriminated between different scores by using words, this option should be checked.



The image shows a checkbox labeled "Strings as factors". The checkbox is currently unchecked, and the label is in a bold, dark font.

Next to these specifications, the user can choose to only use a certain selection of columns that are present in their data, or to upload all the columns.



The image shows a web interface element for column selection. It consists of two dropdown menus labeled "From column" and "To column", both of which currently show the value "1". To the right of these dropdowns is a checkbox labeled "All columns", which is currently unchecked. The entire interface is set against a light grey background.

## Data Types

**Specify the kind of data that is uploaded:**

Raw Data ▲

Raw Data

Adjacency Matrix

Edgelist

The user can enter three different types of data. The first option is “raw data”. When you have raw data, your datafile consists of columns, which indicate the individual items or variables, and rows that signify each individual participant or measurement of those items or variables. Below is a simplified example of what raw data looks like.

##	participant	Item1	Item2
## 1	1	6	2
## 2	2	2	3
## 3	3	4	1
## 4	4	6	1
## 5	5	5	5
## 6	6	3	2
## 7	7	5	5
## 8	8	7	2
## 9	9	3	2
## 10	10	4	4

The next available option as a data type is the “adjacency matrix”. When you use an adjacency matrix, the data will have the individual items or variables on both the rows and the columns, and each cell indicate whether there is a relation between the two variables, and if so, how strong this relation is. To illustrate, let us take the following adjacency matrix:

##	Item1	Item2	Item3	Item4
## 1	0	1	2	1
## 2	1	0	3	2
## 3	2	3	0	1
## 4	1	2	1	0

In this example, it can be seen that there are four variables, as there are four rows and columns. Between variable 1 and 2, a relation exists with strength 1. In this example, the diagonal consists of zeroes, which is not obligatory; it is possible to construct a network with so-called *self-loops*: edges that start and finish at the same node without the interference of some other node. Furthermore, the example above depicts a symmetrical matrix, but this does not have to be the case: networks can be constructed in which the relation from A to B is different from the relation from B to A.

The last available data type is the “edge list”. An edge list is characterized by two columns that depict the starting and ending node for each individual edge, which are represented by the rows. Take for example the edge list below:

```
##   from to
## 1    1  2
## 2    2  3
## 3    3  4
## 4    4  5
## 5    5  1
```

In this example, an edge originates in node 1, and ends in node 2, without the interference of some other node.

In the examples that were previously used, a header existed that shows the variable name for each column. Note that this may help in the identification of nodes when a network is constructed, but that a header is not necessary in order to construct a network.


## Missing Data

After you have specified your data, you can specify the symbol used for missing data, if these exist in your data file. If you do not have any missing data, you can leave this box empty.

**Missing value coding:**

## Example Data

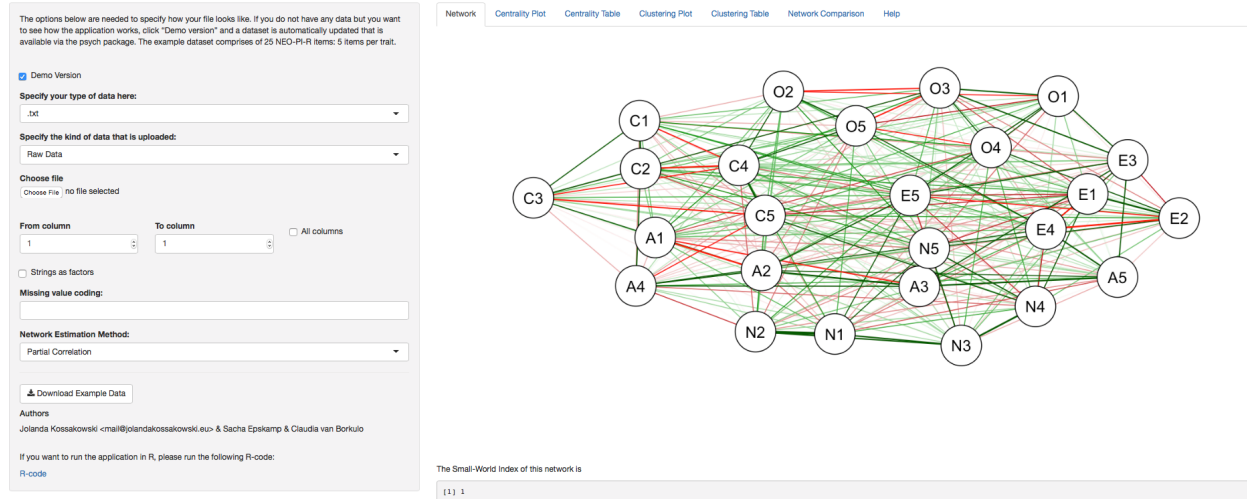
If you do not have any data (yet) that can be used to construct networks, you can download an example data set, which is a raw data file that contains the first 25 variables of the NEO-PI-R. This data is made available via the *psych* package.

 **Download Example Data**

## Network Analysis

After entering all specifications with respect to the data, a network is automatically constructed and displayed on the right, in the tab *Network*:

## Network App



This network is constructed by means of the default values that exist. Almost everything can be changed with respect to the network esthetics, and the estimation of the network. I will explain all options in more detail below.

New in version 1.0.0 is that the network estimation box is now in the grey left panel. This ensures that, when the user selects a certain estimation method, only those options that the user must specify are available to the user.

## Non-Paranormal Transformation

In case that you have continuous data that is non-normally distributed, you can check the *Non-Paranormal Transformation* box. This will transform your data into normally distributed data (Zhao et al. 2014).

☐ Non-Paranormal Transformation

## Network Estimation

There are various methods for estimating a network structure. The NetworkApp has employed 11 methods that are used often. Note that the estimation methods only with when you use raw data. As the absence/presence of edges and/or their strength is already specified in an adjacency matrix or edge list, the estimation methods become redundant. The following estimations methods are currently available:

FDRnetwork

GLASSO

Graphical VAR: PCC

Graphical VAR: PDC

IC-Algorithm: DAG

IC-Algorithm: Skeleton

Multilevel VAR: 2-level

Multilevel VAR: 3-level

Partial Correlation

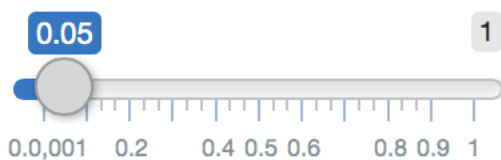
Pearson Correlation

VAR-model

### FDR Network

The *FDR Network* estimation method estimates a network based on the False Discovery Rate (FDR). By setting a cutoff value between 0 and 1, edges with an FDR higher than the cutoff value will be removed from the network.


#### Cut-off value FDR network:



There are three different methods that can be used to create an FDR network:



### Method for FDR Network:



One can use a local false discovery rate, the  $p$ -value, or the  $q$ -value. When estimating an FDR network, one is obliged to specify the method and a cut-off value.

### GLASSO

The *GLASSO* estimation method is used to control for spurious correlations that may arise due to multiple testing. To create a sparse network, the EBICglasso, which uses penalized maximum likelihood estimation, is applied to set spurious connections to zero. The result is a network in which connections between nodes represent a nonzero partial correlation, while controlling for all other nodes in the network (Kossakowski et al. 2015).

No further action is required of the user to construct a GLASSO network: by selecting GLASSO, the application will automatically construct this type of network.

### Graphical VAR: PCC

The *Graphical VAR: PCC* estimation method estimates a Graphical Vector Autoregressive (VAR) model with Partial Contemporaneous Correlations (PCC). By using a LASSO estimation together with the Extended Bayesian Information Criterion (EBIC), sparse networks are constructed. The result of this estimation method is a network in which the edges denote the correlation between two variables at the same time point after removing all linear effects of other variables at the same time point and of all variables at the previous time point (Wild et al. 2010). Note that the Graphical VAR network is appropriate when the data consists of multiple measurements of one participant.

This method is useful when the user wants to estimate a network based on time series data. In order to estimate a proper PCC-network, one has to have all the time points on the rows, and the variables on the columns.

No further action is required of the user to construct a PCC network: by selecting Graphical VAR: PCC, the application will automatically construct this type of network.

### Graphical VAR: PDC

The *Graphical VAR: PDC* estimation method estimates a Graphical Vector Autoregressive (VAR) model with Partial Directed Correlations (PDC). By using a LASSO estimation together with the Extended Bayesian Information Criterion (EBIC), sparse networks are constructed. The result of this estimation method is a lag-1

network in which the edges denote the linear association between two variables, with one being at time point  $t$ , and the other at timepoint  $t-1$ , after removing all linear effects of all variables at the previous timepoint. Note that the Graphical VAR network is appropriate when the data consists of multiple measurements of one participant.

This method is useful when the user wants to estimate a network based on time series data. In order to estimate a proper PDC-network, one has to have all the time points on the rows, and the variables on the columns.

No further action is required of the user to construct a PDC network: by selecting Graphical VAR: PDC, the application will automatically construct this type of network.

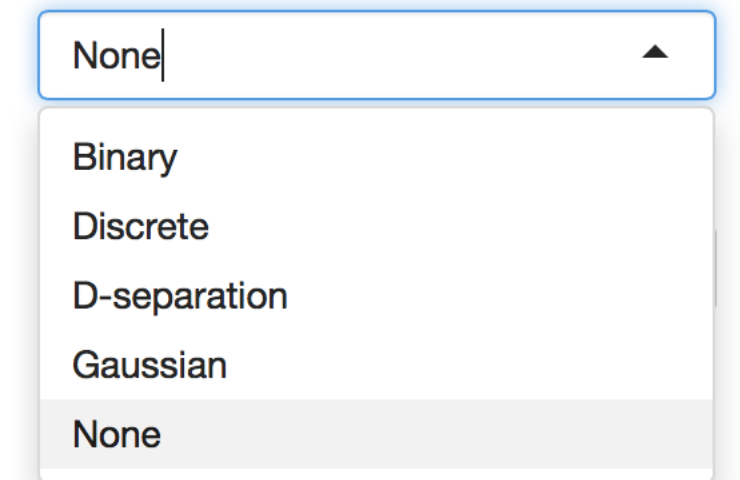
### IC-Algorithm: DAG

The *IC-Algorithm: DAG* estimation method can be used to construct a *Directed Acyclic Graph* (DAG) by using the *Inferred Causation (IC) Algorithm* (Pearl 2009). This DAG is created by means of the *pcalg* package (Kalisch et al. 2012). This function estimates the equivalence class of a DAG, which means that multiple networks can describe the same set of conditional independencies. More specifically, a *Completed Partially Directed Acyclic Graph* (CPDAG) is estimated, which is a network that may contain both directed and undirected edges. A directed edge is drawn when this edge is present in all DAGs in the equivalence class, and an undirected edge is drawn when at least one DAG contains the edge with direction  $a \rightarrow b$ , and at least one DAG that contains the same edge, but with direction  $b \rightarrow a$ .

When the user wants to estimate a DAG, one also has to specify which method is to be used to test the conditional independencies:

## Method for Conditional Independence

### IC-Algorithm:



None

Binary

Discrete

D-separation

Gaussian

None

The choice of method mainly depends on the data: If the data contains Gaussian data, the user would choose Gaussian. The same holds for Binary data and Discrete (multinomial) data. D-separation is a special method that states that two variables are conditionally independent controlling for a third variable if that third variable blocks ALL paths between the initial two variables.

### IC-Algorithm: Skeleton

The *IC-Algorithm: Skeleton* is an estimation method that estimates the skeleton of a Directed Acyclic Graph (DAG) using the PC-algorithm. The result is an undirected network. As with the DAG estimation

method, the user has to specify the function that is used for testing conditional independence. All conditional independencies are tested with an alpha of 0.05.

### Multilevel VAR: 2-level

The *Multilevel VAR: 2-level* is an estimation method that estimates a multivariate vector autoregression parameters between variables. The 2-level version is appropriate for time-series data that contains multiple participants and multiple measurements (either on the day-level or within the day-level). The result of this type of network is a directed network that contains the relation between variables over time, and the effect of a variable on itself over time (Möttus, Epskamp, and Francis 2015).

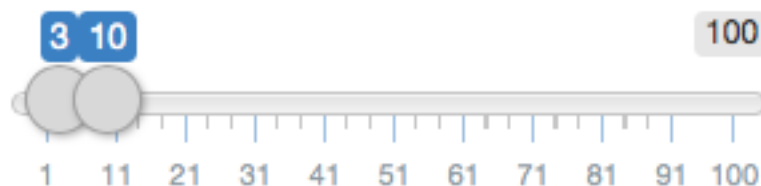
In order to properly estimate a multilevel VAR network, you must select “all columns” when uploading your data.

When a user wants to estimate a multilevel VAR network with two levels, one has to first specify which variable holds the participant numbers:

#### Subject Identification Variable

Also, because the datafile holds several columns that are only used to identify the individual measurement, the user must specify which columns contain the actual data that you want to use in your estimation.

#### Select Variable Columns

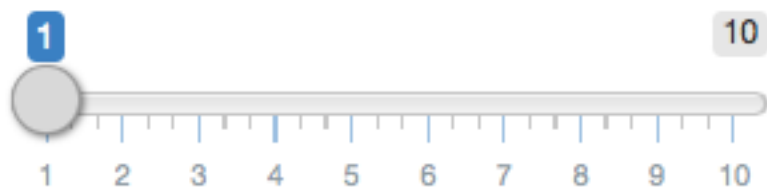


An information criterion is used to select the best fitting model, which is in turn visualized as a network. The default is the *Akaike Information Criterion* (AIC), but one can also choose the *Bayesian Information Criterion* (BIC):

#### Information Criterion

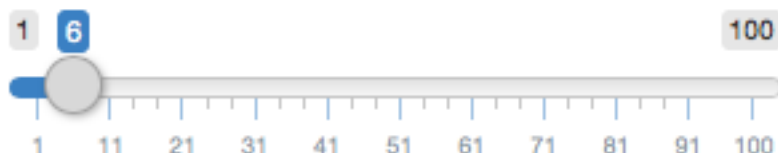
In VAR-models, one typically looks at the effect of a variable on all other variables and on itself at the next time point. This means that the time lag used to find the proper effect is 1. In some cases, it may be interesting to see what the effect of a variable is when this time lag is different. It is therefore possible to set the time lag. Note that it may return an error when a time lag is chosen that is larger than the amount of measurements per person.

## Time Lag



Some users may want to fix how many edges may end at a variable. If you do not have a preference for this, you can set the maximum incoming edges to the equal to the amount of variables in your network.

## Maximum Incoming edges per node



By default, the multilevel VAR network uses a stepwise procedure to find the best fitting network. By unchecking this option, the application will run a nodewise procedure in order to find the best fitting combination of edges and edge weights.

## ☒ Stepwise Estimation

### Multilevel VAR: 3-level

The *Multilevel VAR: 3-level* is an estimation method that estimates a multivariate vector autoregression parameters between variables. The 3-level version is appropriate for time-series data that contains multiple participants, measurements over multiple days and multiple measurements within a day.

Next to all the options that are available in the 2-level network, the user also has to specify the column name that contains the day variable, as we are now constructing a 3-level network.

## Day Identification Variable

### Partial Correlation

The *Partial Correlation* estimation method estimates the correlation between two variables, controlling for all other variables in the network. No further action is required of the user to construct a partial correlation network: by selection Partial Correlation, the application will automatically construct this type of network.

## Pearson Correlation

The *Pearson Correlation* method is a method that estimates the correlation between two variables. The result is a network in which all correlations are displayed. No further action is required of the user to construct a correlation network: by selection Pearson Correlation, the application will automatically construct this type of network.

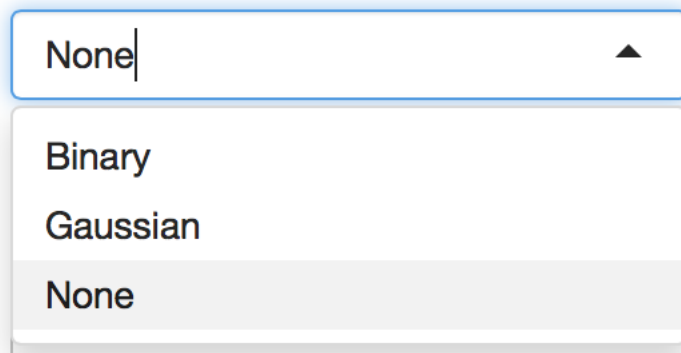
## VAR-model

The *VAR-model* estimation method estimates a *vector autoregressive lag-1* (VAR) model using General Linear Models (GLM). By using GLM, each node is predicted based on all other nodes at the previous time point. The edges in the resulting network represent the lag-1 association between two variables.

This method is useful when the user wants to estimate a network based on time series data. In order to estimate a proper VAR-network, one has to have all the time points on the rows, and the variables on the columns.

In order to run the VAR-model estimation method, the user has to specify the distribution family of the data:

### Distribution family for VAR-model:



None

Binary

Gaussian

None

There are currently two options: the data can either be binary or continuously measured. In case of the first, the user selects binary, and in case of the latter, the user chooses Gaussian.

## Small World Index

The *Small World Index* (SWI) is an index that tells you whether a network is in fact a small world or not. A Small World Network is a network in which the average distance between two nodes resembles that of a Random Network, and where the average ratio between the amount of existing edges and possible edges (*clustering*) is high in comparison to a Random Network (Watts and Strogatz 1998).

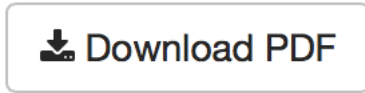
### The Small-World Index of this network is

[1] 1

A network is considered a Small World Network when the SWI is higher or equal to 3. The SWI is automatically calculated for each network that is constructed, and will be re-calculated when parameters are changed.

## Download Network

Each network that is constructed can be download as a .pdf file by clicking on the corresponding button:



## Network Esthetics

After you have selected the estimation method that you want to use to construct a network, a network is plotted. The user can then go on and change different esthetics.

### General

#### Title

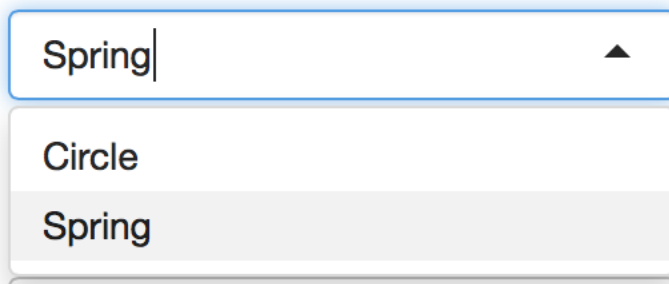
The user can enter a title, which is then displayed above the network, both in the app itself and in the pdf file that can be downloaded.

**Title:**

### Network Layout

There are two options for node placement in the NetworkApp:

#### Network Layout:

A dropdown menu with a blue border. The selected option is "Spring", which is displayed in the top bar. Below the bar, a list of options is shown: "Circle" and "Spring". The "Spring" option is highlighted with a grey background.

The *Circle* layout will place all the node in a circle. The *Spring* layout uses the Fruchterman-Reingold Algorithm (Fruchterman and Reingold 1991) that places more strongly connected nodes closer together. The Spring layout is used by default.

## Graph Details

If the option *Graph Details* is checked, the cut-off value and the maximum edge weight is displayed below the network. The graph details will also be displayed on the pdf version, if the option is checked before downloading the figure.

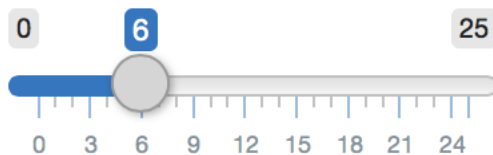
☐ **Graph Details**

## Node Esthetics

### Node Size

The size of the nodes can be adjusted by changing the slider that is displayed below:

#### Node Size:



Depending on the value, the size of the nodes will either be increased or decreased to one's taste.

### Node Shape

The user can change the shape of the node to one of the following shapes that speak for themselves:

#### Node shape:

Circle ▲

Circle

Diamond

Heart

Square

Triangle

### Node Labels

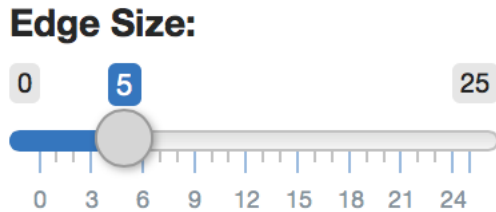
The labels that are displayed within the nodes are subtracted from the header if present in the datafile. If needed, the user can uncheck this option, which will ensure that the nodes are displayed without any text.

☒ **Node Labels**

## Edge Esthetics

### Edge Size

As with the size of the nodes, the size of the edges can be adjusted as well, by changing the value of the following slider:



When the network contains directed edges, the size of the arrow will be adjusted relative to the size of the edges.

### Edge Weights

If edge weights exist in the datafile that is uploaded, these will automatically be displayed. However, it is possible that you want to construct a network without taking the edge weights into account. By unchecking this box, the network will become unweighted.

☒ **Edge Weights**

### Directed Edges

As is the case for the edge weights, it is possible to force the edges to have a direction, or to change a directed network into an undirected one. When the user constructs a PDC-network, one has to check this option in order to see the appropriate network.

☐ **Directed Edges**

### Plot Self-Loops

It may be interesting or useful to plot edges that originate and end in the same node, so-called *self-loops*. By checking this box, self-loops will be automatically drawn and visualized.

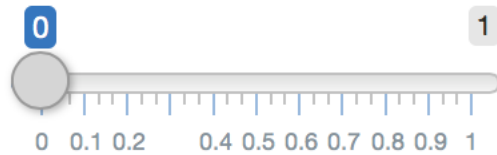
☐ **Plot Self-Loops**



## Edge Threshold

By adjusting this slider, the user can control the value below which the edges are removed from the network. If the edge threshold is changed, the spring layout will also change, next to the results from centrality analyses.

### Edge Threshold:



## Minimum Edge Weight

In order to make a network more interpretable, it is possible to not display edges below a certain value. By adjusting the slider below, the user can control the value above which edges are displayed.

### Minimum Edge Weight:

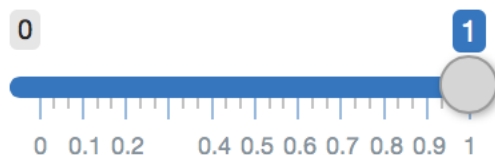


This option differs from the edge threshold, in that the edge threshold option will remove the edge, whereas the minimum edge weight option will not remove the edges, but will not display them either.

## Maximum Edge Weight

The user can change this slider to make various networks more comparable. By setting the maximum edge weight, the application will adjust the edge thickness and saturation accordingly.

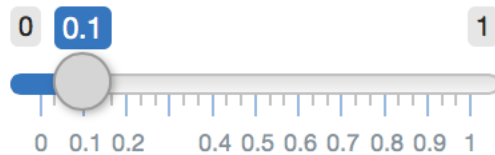
### Maximum Edge Weight:



## Edge Cut-Off Value

The *Cut-Off* option can be used to change the scaling of the edges' thickness and saturation. Edges with an absolute above the set value will have the most saturated colour, and will vary in thickness as the edges become stronger. Edges with an absolute weight below the set value will have the thickness, but will vary in saturation according to the edge weight.

## Edge Cut-Off Value:

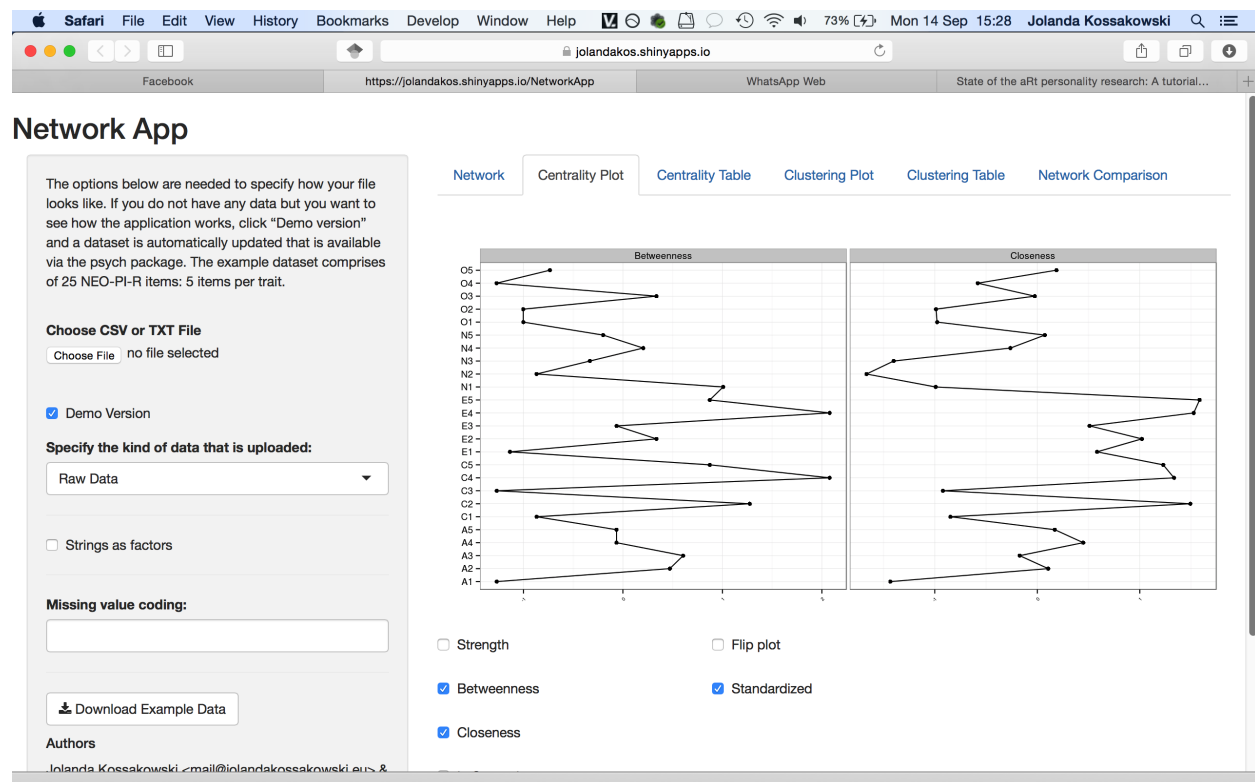


## Centrality Analyses

In network analysis, centrality analyses are used to determine a node's importance in the network (Costantini et al. 2015). A node's importance can be assessed in different ways: the most used measures are the *strength*, the *closeness* and the *betweenness*.

### Centrality plot

When you click on the *centrality plot* tab, the default centrality measures (*closeness* and *betweenness*) are visualized:



### Standardized Centrality Measures

Centrality measures are by default standardized, but you can uncheck this option in order to obtain the raw centrality measures.

☒ **Standardized**

### Centrality Plot Layout

The centrality measures are plotted side by side, where the rows exemplify the various nodes in the network. If wanted, the user can check the *flip plot* in order to turn the plot 45 degrees.

☐ **Flip plot**

### Strength

The *strength* is calculated by summing up the edge weights that are connected to a specific node. By taking the edge weights into account, the strength shows us not which node has the most connections, but which node has the strongest connections. A node with a high strength centrality is a node that influences many other nodes.

☐ **Strength**

### Betweenness

The *betweenness* is the amount of shortest paths between any two nodes (the shortest route from node A to node B) that pass a specific node. A node with a high betweenness centrality is considered a “hub”, a node through which the most shortest paths go through.

☒ **Betweenness**

### Closeness

The *closeness* is the inverse of the sum of the distance from one node to all other nodes in the network. As such, the closeness shows the predictive quality that a specific node may have. A node with a high closeness centrality is a node from which it is easy to travel to all other nodes. In other words, a node with a high closeness centrality is a node that can predict other nodes well.

☒ **Closeness**

### In Strength

When the user has uploaded either an adjacency matrix or an edge list, it is possible to calculate the so-called *In Strength*. The in strength is calculated by summing up the weights of all the incoming edges of a specific node. A node with a high in strength is a node that is highly influential, as many paths end to that specific node.

☐ **In Strength**

## Out Strength

The *Out Strength* is the exact opposite of the in strength. So, instead of summing up the weights of all incoming edges, the out strength is calculated by summing up the weights of all outgoing edges. This means that a node with a high out strength is one that influences a lot of other nodes, as many edges originate in that specific node.

### ☐ Out Strength

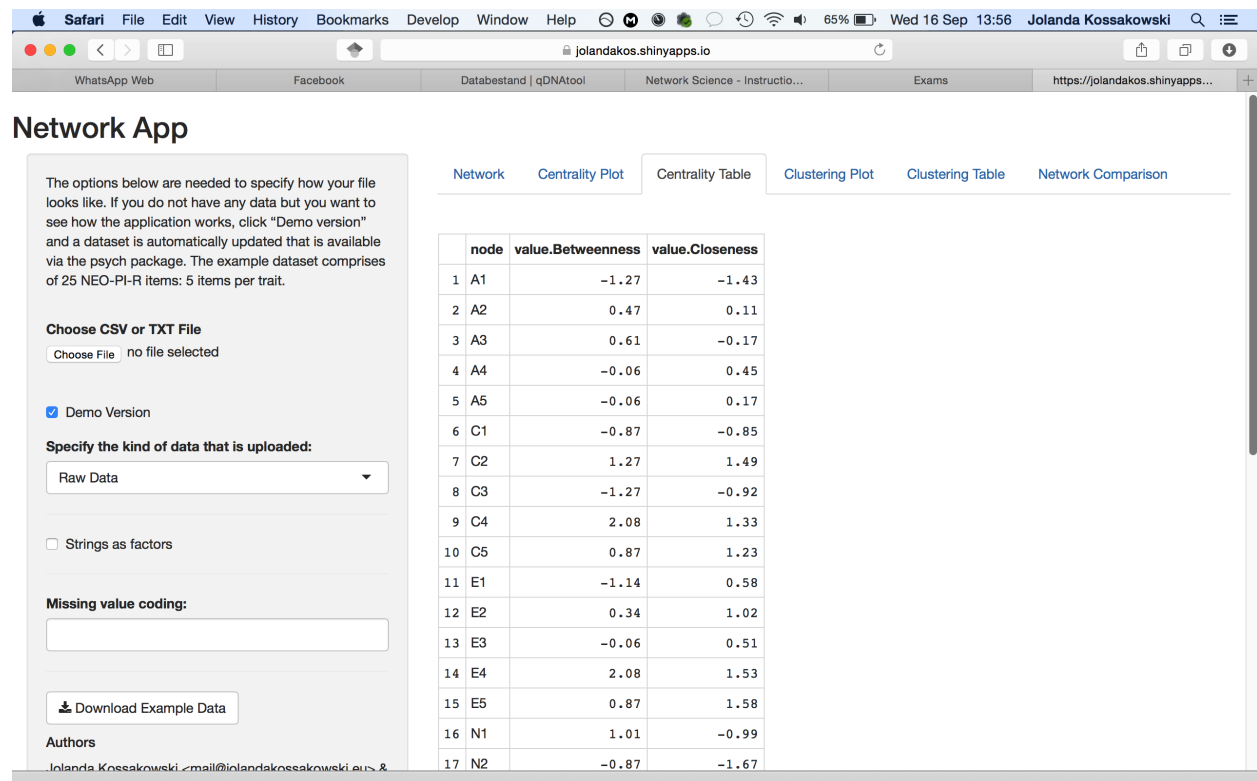
## Download Centrality Plot

After choosing the centrality measures and deciding on the plot layout, the centrality plot can be downloaded as a .pdf file.



## Centrality table

When you click on the *centrality plot* tab, the default centrality measures (closeness and betweenness) are represented in a table:

The screenshot shows a web browser window with the URL "jolandakos.shinyapps.io". The page title is "Network App". On the left, there is a sidebar with options to "Choose CSV or TXT File", "Choose File" (no file selected), "Demo Version" (checked), "Specify the kind of data that is uploaded:" (Raw Data), "Strings as factors" (unchecked), "Missing value coding:" (empty field), and a "Download Example Data" button. The main content area has tabs for "Network", "Centrality Plot", "Centrality Table", "Clustering Plot", "Clustering Table", and "Network Comparison". The "Centrality Table" tab is active, displaying a table with 17 rows and 3 columns: "node", "value.Betweenness", and "value.Closeness".

	node	value.Betweenness	value.Closeness
1	A1	-1.27	-1.43
2	A2	0.47	0.11
3	A3	0.61	-0.17
4	A4	-0.06	0.45
5	A5	-0.06	0.17
6	C1	-0.87	-0.85
7	C2	1.27	1.49
8	C3	-1.27	-0.92
9	C4	2.08	1.33
10	C5	0.87	1.23
11	E1	-1.14	0.58
12	E2	0.34	1.02
13	E3	-0.06	0.51
14	E4	2.08	1.53
15	E5	0.87	1.58
16	N1	1.01	-0.99
17	N2	-0.87	-1.67

The centrality table tab has exactly the same functions as the centrality plot tab. For more information on the different centrality measures and functions, you can read the [centrality plot section](#)

## Download Centrality Table

After choosing the centrality measures, the centrality table can be downloaded as a .txt file.

 **Download Centrality Table**

## Clustering Analyses

Besides centrality measures, *Clustering coefficients* can be interesting for the assessment of a network. The local clustering coefficient of a node is the ratio of amount of connections of a specific node, and the maximum possible amount of connections (Costantini et al. 2015). The clustering coefficient can give information on the importance of a node in a network: if a node's neighbours are already connected to each other, removing that initial node will not result in drastic changes.

Often, clustering coefficients are calculated by looking at the amount of *triangles* that a specific node is part of: three nodes that are connected to each other. Note that clustering coefficients can only be calculated for undirected networks. In the application, if the user constructed a directed network, clustering coefficients are automatically computed for the undirected version.

## Clustering Plot

When you click on the *clustering plot* tab, the default clustering measures (WS, Zhang, Onnela and Barrat) are visualized:

### Standardized Clustering Coefficients

Clustering measures are by default standardized, but you can uncheck this option in order to obtain the raw clustering measures.

### Clustering Plot Layout

The clustering measures are plotted side by side, where the rows exemplify the various nodes in the network. If wanted, the user can check the *flip plot* in order to turn the plot 45 degrees.

### WS

The *WS* (Watts & Strogatz) is the unweighted clustering coefficient of a specific node. As its definition states, the edge weights are not taken into account in the calculation of the clustering coefficients.

☒ **WS**

## Zhang

The *Zhang* clustering coefficient is a clustering coefficient for weighted networks. It is similar to the WS coefficient, but instead of looking at the maximum amount of connections that a node may have, the Zhang coefficient looks at the maximum possible value that can be obtained with the edge weights.

✓ Zhang

## Onnela

The *Onnela* clustering coefficient calculates a clustering coefficient for each node by taking into account the edge weights that exist in the triangles, and the amount of triangles that a specific node is part of. The Onnela coefficient differs from the Barrat coefficient in that the Onnela coefficient scales the edge weights into the *average intensity*.

✓ Onnela

## Barrat

The *Barrat* clustering coefficient calculates a coefficient for each node by taking into account the edge weights that exist in the triangles, and the amount of triangles that a specific node is part of.

✓ Barrat

## Download Clustering Plot

After choosing the clustering coefficients and deciding on the plot layout, the clustering plot can be downloaded as a .pdf file.

 Download Clustering Plot

## Clustering Table

The clustering table tab has exactly the same functions as the clustering plot tab. For more information on the different clustering coefficients and functions, you can read the clustering plot section

## Download Clustering Table

After choosing the clustering coefficients, the clustering table can be downloaded as a .txt file.

 Download Clustering Table

# Network Comparison

It may be interesting for some researchers to investigate the difference between two network structures. This has now been made possible in the application (Van Borkulo et al. 2015). The *Network Comparison Test* works independently of the rest of the application. Note that the grey left panel is not needed for the Network Comparison test.

## Network App

The options below are needed to specify how your file looks like. If you do not have any data but you want to see how the application works, click "Demo version" and a dataset is automatically updated that is available via the psych package. The example dataset comprises of 25 NEO-PI-R items: 5 items per trait.

☐ Demo Version

Specify your type of data here:  
[.txt]

Specify the kind of data that is uploaded:  
[Raw Data]

Choose file  
[Choose File] no file selected

From column: [1] To column: [1] ☐ All columns

☐ Strings as factors

Missing value coding:  
[ ]

Network Estimation Method:  
[Multilevel VAR: 2-level]

[Download Example Data](#)

Authors  
Jolanda Kossakowski <mail@jolandakossakowski.eu> & Sacha Epskamp & Claudia van Borkulo

If you want to run the application in R, please run the following R-code:  
[R-code](#)

Network Centrality Plot Centrality Table Clustering Plot Clustering Table Network Comparison Help

### Dataset 1

Specify your type of data here:  
[.txt]

Choose file  
[Choose File] no file selected

☐ Strings as factors

Missing value coding:  
[ ]

### Dataset 2

Specify your type of data here:  
[.txt]

Choose file  
[Choose File] no file selected

☐ Strings as factors

Missing value coding:  
[ ]

### Network Comparison Test Specifications

Amount of iterations: [100]

Rule to define edges: [AND-rule]

Gamma: [0]

☐ Weighted

☐ Binary data

Error: argument of length 0

The difference between network structures is defined as the deviation in absolute weighted sum scores of the connections (Barrat et al. 2004). This permutation-based test randomly regroups participants from both samples repeatedly and calculates the differences between these subsamples. The resulting distribution under the null hypothesis (both subsamples are equal) is used to test the observed difference of the original subsamples (Kossakowski et al. 2015).

The result of the comparison test is expressed as a  $p$ -value, which can then be set off against an alpha of your choosing

```
[1] "p-value"  
[1] 0.14
```

## Uploading Data

As with the construction of a network, the two samples need to be uploaded. Note that ONLY raw data can be used in this comparison test. Please read the [data uploading section](#) for more information.

## Dataset 1

Choose CSV or TXT File

no file selected

☐ Strings as factors

Missing value coding:

## Dataset 2

Choose CSV or TXT File

no file selected

☐ Strings as factors

Missing value coding:

## Specifications

After uploading the data, the user can set a few specifications before running the comparison test.

### Iterations

The user can specify how many iterations are appropriate for the comparison. The default is 100 iterations, but the standard is 1000 iterations. Note that, the higher the amount of iterations, the longer it takes for the test to be finished.

### Amount of iterations

### Gamma

The *gamma* is the tuning parameter that is needed for the Extended Bayesian Information Criterion (EBIC) that is used within the function. The default is 0, but 0.5 is also often chosen.

### Gamma

### Edge defining rule

The Network Comparison Test fit an Ising network to the data before calculating the difference between the networks. The result of this fitting procedure is an asymmetrical adjacency matrix, where a nonzero value stands for the presence of an edge.



## Rule to define edges

AND-rule ▲

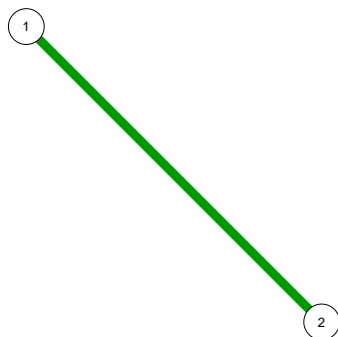
AND-rule

OR-rule

In order to fit an Ising network, the user must specify what kind of rule must be used to define the absence or presence of an edge. The *AND-rule* states that in the adjacency matrix, both the edge from node 1 to node 2 and the edge from node 2 to node 1 must be nonzero in order for that edge to be drawn. The resulting edge is the average of the two edge weights:

```
##      [,1] [,2]
## [1,]  0.0  0.3
## [2,]  0.6  0.0

## [1] 0.45
```



In this example, an edge weight from node 1 to node 2 and an edge weight from node 2 to node 1 is returned by the `IsingFit` function, and thus in the resulting network, an edge is drawn with its weight being the average of the two edge weights.

```
##      [,1] [,2]
## [1,]  0.0  0
## [2,]  0.6  0

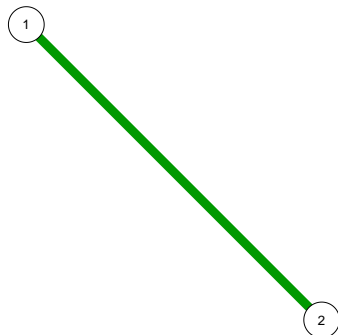
## [1] 0
```



Here, only an edge from node 2 to node 1 is returned by the IsingFit function, and as a result, no edge is drawn between the two nodes.

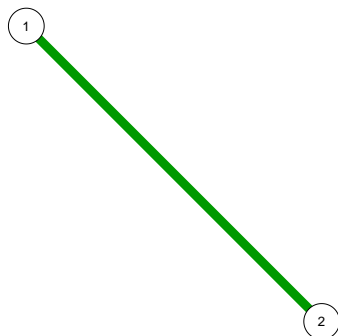
When the *OR-rule* is used, at least of the two possible edges between nodes 1 and 2 (from node 1 to node 2 or from node 2 to node 1) needs to be nonzero in order for it to be drawn. When only one of the two edges is nonzero, that value is used as the edge weight. In the case that both entries are nonzero, the average is used as the edge weight:

```
##      [,1] [,2]
## [1,]  0.0  0
## [2,]  0.6  0
```



```
##      [,1] [,2]
## [1,]  0.0  0.3
## [2,]  0.6  0.0
```

```
## [1] 0.45
```



In these two cases, at least one edge is nonzero, and thus in the resulting network, an edge is drawn.



In this example, both the edge from node 1 to node 2 and the edge from node 2 to node 1 are zero, and thus no edge is drawn between the two nodes.

### **Weighted Networks**

Depending on the research question, the user can specify whether weighted networks are to be compared, or whether unweighted networks are used. Weighted networks take the edge weights into account, whereas unweighted networks only look at the absence/presence of an edge.

☐ **Weighted**

### **Binary Data**

The user can specify whether the data samples consist of binary data by checking the box below.

☐ **Binary data**

## References

- Barrat, Alain, Marc Barthélemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. 2004. “The Architecture of Complex Weighted Networks.” *Proceedings of the National Academy of Sciences of the United States of America* 101. National Acad Sciences: 3747–52.
- Costantini, Giulio, Sacha Epskamp, Denny Borsboom, Marco Perugini, René Möttus, Lourens J Waldorp, and Angélique OJ Cramer. 2015. “State of the ART Personality Research: A Tutorial on Network Analysis of Personality Data in R.” *Journal of Research in Personality* 54. Elsevier: 13–29.
- Csardi, Gabor, and Tamas Nepusz. 2006. “The Igraph Software Package for Complex Network Research.” *InterJournal, Complex Systems* 1695: 1–9. <http://igraph.org>.
- Epskamp, Sacha, Angélique O. J. Cramer, Lourens J. Waldorp, Verena D. Schmittmann, and Denny Borsboom. 2012. “qgraph: Network Visualizations of Relationships in Psychometric Data.” *Journal of Statistical Software* 48: 1–18. <http://www.jstatsoft.org/v48/i04/>.
- Fruchterman, Thomas M J, and Edward M Reingold. 1991. “Graph drawing by force-directed placement.” *Software: Practice and Experience* 21: 1129–64.
- Kalisch, Markus, Martin Mächler, Diego Colombo, Marloes H Maathuis, and Peter Bühlmann. 2012. “Causal Inference Using Graphical Models with the R Package Pcalg.” *Journal of Statistical Software* 47 (11): 1–26.
- Kossakowski, Jolanda J, Sacha Epskamp, Jacobien M Kieffer, Claudia D van Borkulo, Mijke Rhemtulla, and Denny Borsboom. 2015. “The Application of a Network Approach to Health-Related Quality of Life (HRQoL): Introducing a New Method for Assessing HRQoL in Healthy Adults and Cancer Patient.”
- Möttus, R, S Epskamp, and A Francis. 2015. “Within-Individual Variability of Personality Characteristics and Physical Exercise: Contemporaneous and Lagged Associations.” *Journal of Research in Personality*, Submitted for Publication.
- Pearl, Judea. 2009. *Causality*. Cambridge university press.
- Van Borkulo, C D, L J Waldorp, L Boschloo, R A Schoevers, and D Borsboom. 2015. “Tatistical Comparison of Two Net- Works with Respect to Global Strength.” [https:// github.com/cvborkulo/NetworkComparisonTest](https://github.com/cvborkulo/NetworkComparisonTest).
- Watts, Duncan J, and Steven H Strogatz. 1998. “Collective Dynamics of ‘Small-World’ Networks.” *Nature* 393. Nature Publishing Group: 440–42.
- Wild, Beate, Michael Eichler, Hans-Christoph Friederich, Mechthild Hartmann, Stephan Zipfel, and Wolfgang Herzog. 2010. “A Graphical Vector Autoregressive Modelling Approach to the Analysis of Electronic Diary Data.” *BMC Medical Research Methodology* 10: 1–13.
- Zhao, Tuo, Han Liu, Kathryn Roeder, John Lafferty, and Larry Wasserman. 2014. *Huge: High-Dimensional Undirected Graph Estimation*. <http://CRAN.R-project.org/package=huge>.