Network psychometrics

Epskamp, S.

Publication date
2017
Document Version
Other version
License
Other

Chapter 9

Network Visualizations of Relationships in Psychometric Data

Abstract

We present the *qgraph* package for R, which provides an interface to visualize data through network modeling techniques. For instance, a correlation matrix can be represented as a network in which each variable is a node and each correlation an edge; by varying the width of the edges according to the magnitude of the correlation, the structure of the correlation matrix can be visualized. A wide variety of matrices that are used in statistics can be represented in this fashion, for example matrices that contain (implied) covariances, factor loadings, regression parameters and p values. *qgraph* can also be used as a psychometric tool, as it performs exploratory and confirmatory factor analysis, using *sem* (Fox, 2006) and *lavaan* (Rosseel, 2012); the output of these packages is automatically visualized in *qgraph*, which may aid the interpretation of results. In this article, we introduce *qgraph* by applying the package functions to data from the NEO-PI-R, a widely used personality questionnaire.

9.1 Introduction

The human visual system is capable of processing highly dimensional information naturally. For instance, we can immediately spot suggestive patterns in a scatterplot, while these same patterns are invisible when the data is numerically represented in a matrix.

We present *qgraph*[1], an R package that accommodates this capacity for spotting patterns by visualizing data in a novel way: through networks. Networks consist

This chapter has been adapted from: Epskamp, S., Cramer, A.O.J., Waldorp, L.J., Schmittmann, V.D., and Borsboom, D. (2012). qgraph: Network Visualizations of Relationships in Psychometric Data. *Journal of Statistical Software, 48* (1), 1–18.

[1] http://cran.r-project.org/web/packages/qgraph/index.html

of nodes (also called 'vertices') that are connected by edges (Harary, 1969). Each edge has a certain weight, indicating the strength of the relevant connection, and in addition edges may or may not be directed. In most applications of network modeling, nodes represent entities (e.g., people in social networks, or genes in gene networks). However, in statistical analysis it is natural to represent variables as nodes. This representation has a longstanding tradition in econometrics and psychometrics (e.g., see Bollen & Lennox, 1991; Edwards & Bagozzi, 2000), and was a driving force behind the development of graphical models for causal analysis (Spirtes, Glymour, & Scheines, 2000; Pearl, 2000). By representing relationships between variables (e.g., correlations) as weighted edges important structures can be detected that are hard to extract by other means. In general, *qgraph* enables the researcher to represent complex statistical patterns in clear pictures, without the need for data reduction methods.

qgraph was developed in the context of network approaches to psychometrics (Cramer et al., 2010; Borsboom, 2008; Schmittmann et al., 2013), in which theoretical constructs in psychology are hypothesized to be networks of causally coupled variables. In particular, *qgraph* automates the production of graphs such as those proposed in Cramer et al. (2010). However, the techniques in the package have also proved useful as a more general tool for visualizing data, and include methods to visualize output from several psychometric packages like *sem* (Fox, 2006) and *lavaan* (Rosseel, 2012).

A number of R packages can be used for the visualization and analysis of networks (e.g., *network*, Butts, Handcock, & Hunter, 2011; *statnet* Handcock, Hunter, Butts, Goodreau, & Morris, 2008; *igraph* Csardi & Nepusz, 2006). In visualizing graphs *qgraph* distinguishes itself by being specifically aimed at the visualization of statistical information. This usually leads to a special type of graph: a non-sparse weighted graph. Such graphs typically contain many edges (e.g., a fully connected network with 50 nodes has 2450 edges) thereby making it hard to interpret the graph; as well as inflating the file size of vector type image files (e.g., PDF, SVG, EPS). *qgraph* is specifically aimed at presenting such graphs in a meaningful way (e.g., by using automatic scaling of color and width, cutoff scores and ordered plotting of edges) and to minimize the file size of the output in vector type image files (e.g., by minimizing the amount of polygons needed). Furthermore, *qgraph* is designed to be usable by researchers new to R, while at the same time offering more advanced customization options for experienced R users.

qgraph is not designed for numerical analysis of graphs (Boccaletti, Latora, Moreno, Chavez, & Hwang, 2006), but can be used to compute the node centrality measures of weighted graphs proposed by Opsahl et al. (2010). Other R packages as well as external software can be used for more detailed analyses. *qgraph* facilitates these methods by using commonly used methods as input. In particular, the input is the same as used in the *igraph* package for R, which can be used for many different analyses.

In this article we introduce *qgraph* using an included dataset on personality traits. We describe how to visualize correlational structures, factor loadings and structural equation models and how these visualizations should be interpreted. Finally we will show how *qgraph* can be used as a simple unified interface to perform several exploratory and confirmatory factor analysis routines available in

R.

9.2 Creating Graphs

Throughout this article we will be working with a dataset concerning the five factor model of personality (Benet-Martinez & John, 1998; Digman, 1989; Goldberg, 1990a, 1993; McCrae & Costa, 1997). This is a model in which correlations between responses to personality items (i.e., questions of the type 'do you like parties?', 'do you enjoy working hard?') are explained by individual differences in five personality traits: *neuroticism*, *extraversion*, *agreeableness*, *openness to experience* and *conscientiousness*. These traits are also known as the 'Big Five'. We use an existing dataset in which the Dutch translation of a commonly used personality test, the NEO-PI-R (Costa & McCrae, 1992; Hoekstra, de Fruyt, & Ormel, 2003), was administered to 500 first year psychology students (Dolan, Oort, Stoel, & Wicherts, 2009). The NEO-PI-R consists of 240 items designed to measure the five personality factors with items that cover six facets per factor[2]. The scores of each subject on each item are included in *qgraph*, as well as information on the factor each item is designed to measure (this information is in the column names). All graphs in this chapter were made using R version 2.14.1 (2011-12-22) and *qgraph* version 1.0.0.

First, we load *qgraph* and the NEO-PI-R dataset:

```
library("qgraph")
data("big5")
```

Input Modes

The main function of *qgraph* is called `qgraph()`, and its first argument is used as input for making the graph. This is the only mandatory argument and can either be a weights matrix, an edge-list or an object of class `"qgraph"`, `"loadings"` and `"factanal"` (*stats*; R Core Team, 2016), `"principal"` (*psych*; Revelle, 2010), `"sem"` and `"semmod"` (*sem*; Fox, 2006), `"lavaan"` (*lavaan*; Rosseel, 2012), `"graphNEL"` (*Rgraphviz*; Gentry et al., 2011) or `"pcAlgo"` (*pcalg*; Kalisch, Maechler, & Colombo, 2010). In this chapter we focus mainly on *weights matrices*, information on other input modes can be found in the documentation.

A weights matrix codes the connectivity structure between nodes in a network in matrix form. For a graph with n nodes its weights matrix $\boldsymbol{A}$ is a square n by n matrix in which element a_{ij} represents the strength of the connection, or weight, from node i to node j. Any value can be used as weight as long as (a) the value zero represents the absence of a connection, and (b) the strength of connections is symmetric around zero (so that equal positive and negative values are comparable in strength). By default, if $\boldsymbol{A}$ is symmetric an undirected graph is plotted and otherwise a directed graph is plotted. In the special case where all edge weights are either 0 or 1 the weights matrix is interpreted as an adjacency matrix and an unweighted graph is made.

[2] A facet is a subdomain of the personality factor; e.g., the factor neuroticism has depression and anxiety among its subdomains.

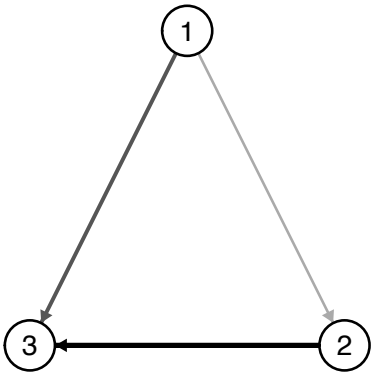


Figure 9.1: A directed graph based on a 3 by 3 weights matrix with three edges of different strengths.

For example, consider the following weights matrix:

$$\begin{bmatrix} 0 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

This matrix represents a graph with 3 nodes with weighted edges from node 1 to nodes 2 and 3, and from node 2 to node 3. The resulting graph is presented in Figure 9.1.

Many statistics follow these rules and can be used as edge weights (e.g., correlations, covariances, regression parameters, factor loadings, log odds). Weights matrices themselves also occur naturally (e.g., as a correlation matrix) or can easily be computed. Taking a correlation matrix as the argument of the function `qgraph()` is a good start to get acquainted with the package.

With the NEO-PI-R dataset, the correlation matrix can be plotted with:

```
qgraph(cor(big5))
```

This returns the most basic graph, in which the nodes are placed in a circle. The edges between nodes are colored according to the sign of the correlation (green for positive correlations, and red for negative correlations), and the thickness of the edges represents the absolute magnitude of the correlation (i.e., thicker edges represent higher correlations).

Visualizations that aid data interpretation (e.g., are items that supposedly measure the same construct closely connected?) can be obtained either by using the `groups` argument, which groups nodes according to a criterion (e.g., being in the same psychometric subtest) or by using a layout that is sensitive to the correlation structure. First, the `groups` argument can be used to specify which nodes belong together (e.g., are designed to measure the same factor). Nodes belonging together have the same color, and are placed together in smaller circles. The `groups` argument can be used in two ways. First, it can be a list in which each

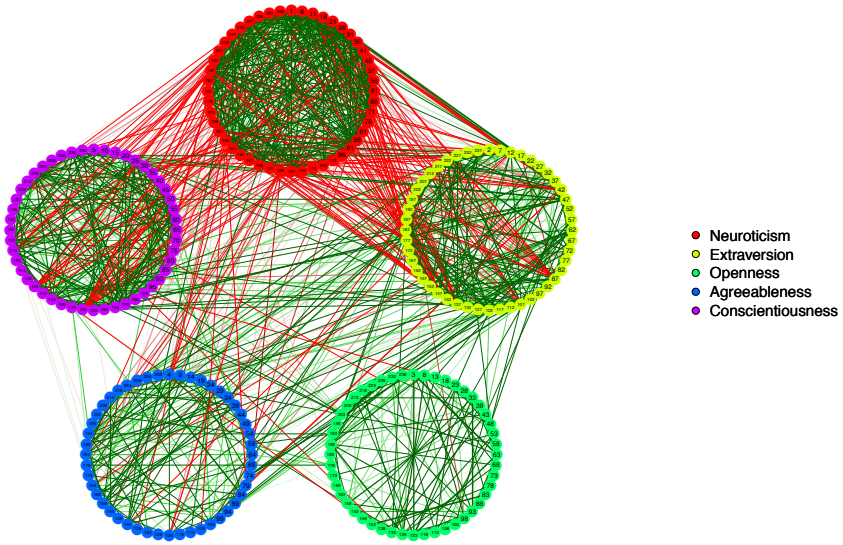


Figure 9.2: A visualization of the correlation matrix of the NEO-PI-R dataset. Each node represents an item and each edge represents a correlation between two items. Green edges indicate positive correlations, red edges indicate negative correlations, and the width and color of the edges correspond to the absolute value of the correlations: the higher the correlation, the thicker and more saturated is the edge.

element is a vector containing the numbers of nodes belonging together. Secondly, it can be a factor in which the levels belong together. The names of the elements in the list or the levels in the factor are used in a legend of requested.

For the Big 5 dataset, the grouping of the variables according to the NEO-PI-R manual is included in the package. The result of using the `groups` argument is a network representation that readily facilitates interpretation in terms of the five personality factors:

```
data("big5groups")
Q <- qgraph(cor(big5), groups = big5groups)
```

Note that we saved the graph in the object `Q`, to avoid specifying these arguments again in future runs. It is easy to subsequently add other arguments: for instance, we may further optimize the representation by using the `minimum` argument to omit correlations we are not interested in (e.g., very weak correlations), `borders` to omit borders around the nodes, and `vsize` to make the nodes smaller:

```
Q <- qgraph(Q, minimum = 0.25, borders = FALSE, vsize = 2)
```

The resulting graph is represented in Figure 9.2.

Layout Modes

Instead of predefined circles (as was used in Figure 9.2), an alternative way of facilitating interpretations of correlation matrices is to let the placement of the nodes be a function of the pattern of correlations. Placement of the nodes can be controlled with the `layout` argument. If `layout` is assigned `"circular"`, then the nodes are placed clockwise in a circle, or in smaller circles if the `groups` argument is specified (as in Figure 9.2). If the nodes are placed such that the length of the edges depends on the strength of the edge weights (i.e., shorter edges for stronger weights), then a picture can be generated that shows how variables cluster. This is a powerful exploratory tool, that may be used as a visual analogue to factor analysis. To make the length of edges directly correspond to the edge weights an high dimensional space would be needed, but a good alternative is the use of force-embedded algorithms (Di Battista, Eades, Tamassia, & Tollis, 1994) that iteratively compute the layout in two-dimensional space.

A modified version of the Fruchterman and Reingold (1991) algorithm is included in *qgraph*. This is a `C` function that was ported from the *sna* package (Butts, 2010). A modification of this algorithm for weighted graphs was taken from *igraph* (Csardi & Nepusz, 2006). This algorithm uses an iterative process to compute a layout in which the length of edges depends on the absolute weight of the edges. To use the Fruchterman-Reingold algorithm in `qgraph()` the `layout` argument needs to be set to `"spring"`. We can do this for the NEO-PI-R dataset, using the graph object `Q` that we defined earlier, and omitting the legend:

```
qgraph(Q, layout = "spring", legend = FALSE)
```

Figure 9.3 shows the correlation matrix of the Big Five dataset with the nodes placed according to the Fruchterman-Reingold algorithm. This allows us inspect the clustering of the variables. The figure shows interesting structures that are far harder to detect with conventional analyses. For instance, neuroticism items (i.e., red nodes) cluster to a greater extent when compared to other traits; especially openness is less strongly organized than the other factors. In addition, agreeableness and extraversion items are literally intertwined, which offers a suggestive way of thinking about the well known correlation between these traits.

The placement of the nodes can also be specified manually by assigning the `layout` argument a matrix containing the coordinates of each node. For a graph of n nodes this would be a n by 2 matrix in which the first column contains the x coordinates and the second column contains the y coordinates. These coordinates can be on any scale and will be rescaled to fit the graph by default. For example, the following matrix describes the coordinates of the graph in Figure 9.1:

$$\begin{bmatrix} 2 & 2 \\ 3 & 1 \\ 1 & 1 \end{bmatrix}$$

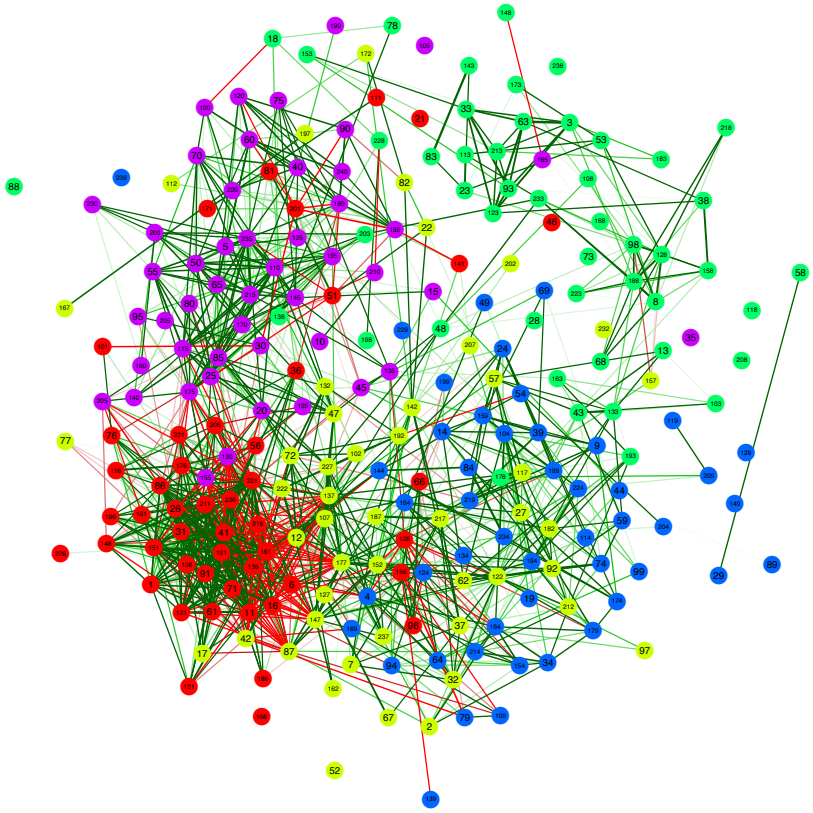


Figure 9.3: A graph of the correlation matrix of the NEO-PI-R dataset in which the nodes are placed by the Fruchterman-Reingold algorithm. The specification of the nodes and edges are identical to Figure 9.2

This method of specifying the layout of a graph is identical to the one used in the *igraph* (Csardi & Nepusz, 2006) package, and thus any layout obtained through *igraph* can be used[3].

One might be interested in creating not one graph but an animation of several graphs that are similar to each other. Such animations can, for example, illustrate the growth of a network over time or show the change of correlational structures in repeated measures. For such similar but not equal graphs the Fruchterman-Reingold algorithm might return completely different layouts, which will make the animation unnecessary hard to interpret. This problem can be solved by limiting the amount of space a node may move in each iteration. The function `qgraph.animate()` automates this process and can be used for various types of animations.

Output Modes

To save the graphs, any output device in R can be used to obtain high resolution, publication-ready image files. Some devices can be called directly by `qgraph()` through the `filetype` argument, which must be assigned a string indicating what device should be used. Currently `filetype` can be `"R"` or `"x11"`[4] to open a new plot in R, raster types `"tiff"`, `"png"` and `"jpg"`, vector types `"eps"`, `"pdf"` and `"svg"` and `"tex"`. A *PDF* file is advised, and this can thus be created with `qgraph(\ldots, filetype = "pdf")`.

Often, the number of nodes makes it potentially hard to track which variables are represented by which nodes. To address this problem, one can define mouseover tooltips for each node, so that the name of the corresponding variable (e.g., the item content in the Big Five graph) is shown when one places the cursor over the relevant node. In *qgraph*, mouseover tooltips can be placed on the nodes in two ways. The `"svg" filetype` creates a *SVG* image using the *RSVGTipsDevice* package (Plate, 2009)[5]. This filetype can be opened using most browsers (best viewed in Firefox) and can be used to include mouseover tooltips on the node labels. The `tooltips` argument can be given a vector containing the tooltip for each node. Another option for mouseover tooltips is to use the `"tex" filetype`. This uses the *tikzDevice* package (Sharpsteen & Bracken, 2010) to create a `.tex` file that contains the graph[6], which can then be built using pdfLaTeX. The `tooltips` argument can also be used here to create mouseover tool tips in a PDF file[7].

[3] To do this, first create an `"igraph"` object by calling `graph.adjacency()` on the weights matrix with the arguments= `weighted=TRUE`. Then, use one of the layout functions (e.g., `layout.spring()`) on the `"igraph"` object. This returns the matrix with coordinates which can be used in `qgraph()`

[4] *RStudio* users are advised to use `filetype="x11"` to plot in R

[5] *RSVGTipsDevice* is only available for 32bit versions of R

[6] Note that this will load the *tikzdevice* package which upon loading checks for a LaTeX compiler. If this is not available the package might fail to load

[7] We would like to thank Charlie Sharpsteen for supplying the tikz codes for these tooltips

Standard visual parameters

In weighted graphs green edges indicate positive weights and red edges indicate negative weights[8]. The color saturation and the width of the edges corresponds to the absolute weight and scale relative to the strongest weight in the graph (i.e., the edge with the highest absolute weight will have full color saturation and be the widest). It is possible to control this behavior by using the `maximum` argument: when `maximum` is set to a value above any absolute weight in the graph then the color and width will scale to the value of `maximum` instead[9]. Edges with an absolute value under the `minimum` argument are omitted, which is useful to keep filesizes from inflating in very large graphs.

In larger graphs the above edge settings can become hard to interpret. With the `cut` argument a cutoff value can be set which splits scaling of color and width. This makes the graphs much easier to interpret as you can see important edges and general trends in the same picture. Edges with absolute weights under the cutoff score will have the smallest width and become more colorful as they approach the cutoff score, and edges with absolute weights over the cutoff score will be full red or green and become wider the stronger they are.

In addition to these standard arguments there are several arguments that can be used to graphically enhance the graphs to, for example, change the size and shape of nodes, add a background or Venn diagram like overlay and visualize test scores of a subject on the graph. The documentation of the `qgraph()` function has detailed instructions and examples on how these can be used.

9.3 Visualizing Statistics as Graphs

Correlation Matrices

In addition to the representations in Figures 9.2 and 9.3, *qgraph* offers various other possibilities for visualizing association structures. If a correlation matrix is used as input, the `graph` argument of `qgraph()` can be used to indicate what *type* of graph should be made. By default this is `"association"` in which correlations are used as edge weights (as in Figures 9.2 and 9.3).

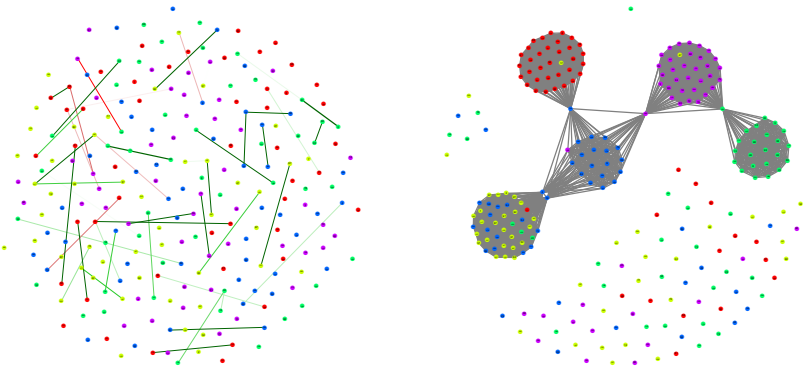
Another option is to assign `"concentration"` to `graph`, which will create a graph in which each edge represents the partial correlation between two nodes: partialling out all other variables. For normally distributed continuous variables, the partial correlation can be obtained from the inverse of the correlation (or covariance) matrix. If $\boldsymbol{P}$ is the inverse of the correlation matrix, then the partial correlation ω_{ij} of variables i and j is given by:

$$\omega_{ij} = \frac{-p_{ij}}{\sqrt{p_{ii}p_{jj}}}$$

Strong edges in a resulting concentration graph indicate correlations between variables that cannot be explained by other variables in the network, and are therefore indicative of causal relationships (e.g., a real relationship between smoking

[8] The edge colors can currently not be changed except to grayscale colors using `gray=TRUE`

[9] This must be done to compare different graphs; a good value for correlations is 1



(a) (b)

Figure 9.4: Additional visualizations based on the correlations of the NEO-PI-R dataset. Panel (a) shows a concentration graph with partial correlations and panel (b) shows a graph in which connections are based on an exploratory factor analysis.

and lung cancer that cannot be explained by other factors, for example gender), provided that all relevant variables are included in the network.

The left panel of Figure 9.4 shows a concentration graph for the NEO-PI-R dataset. This graph shows quite a few partial correlations above 0.3. Within the factor model, these typically indicate violations of local independence and/or simple structure.

A third option for the `graph` argument is `"factorial"`, which creates an unweighted graph based on an exploratory factor analysis on the correlation (or covariance) matrix with promax rotation (using `factanal()` from *stats*). By default the number of factors that are extracted in the analysis is equal to the number of eigenvalues over 1 or the number of groups in the `groups` argument (if supplied). Variables are connected with an edge if they both load higher on the same factor than the cutoff score that is set with the `cut` argument. As such, the `"factorial"` option can be used to investigate the structure as detected with exploratory factor analysis.

The right panel of Figure 9.4 shows the factorial graph of the NEO-PI-R dataset. This graph shows five clusters, as expected, but also displays some overlap between the extraversion and agreeableness items.

qgraph has two functions that are designed to make all of the above graphs in a single run. The first option is to use the `qgraph.panel()` function to create a four-panel plot that contains the association graph with circular and spring layouts, a concentration graph with the spring layout, and a factorial graph with the spring layout. We can apply this function to the Big Five data to obtain all graphs at once:

```
qgraph.panel(cor(big5), groups = big5groups, minimum = 0.25,
     borders = FALSE, vsize = 1, cut = 0.3)
```

A second option to represent multiple graphs at once is to use the `qgraph.svg()` function to produce an interactive graph. This function uses *RSVGTipsDevice* (only available for 32bit versions of R; Plate, 2009) to create a host of SVG files for all three types of graphs, using circular and spring layouts and different cutoff scores. These files contain hyperlinks to each other (which can also be used to show the current graph in the layout of another graph) and can contain mouseover tool tips as well. This can be a useful interface to quickly explore the data. A function that does the same in `tex` format will be included in a later version of *qgraph*, which can then be used to create a multi-page pdf file containing the same graphs as `qgraph.panel()`.

Matrices that are similar to correlation matrices, like covariance matrices and lag-1 correlations in time series, can also be represented in *qgraph*. If the matrix is not symmetric (as is for instance the case for lag-1 correlations) then a directed graph is produced. If the matrix has values on the diagonal (e.g., a covariance matrix) these will be omitted by default. To show the diagonal values the `diag` argument can be used. This can be set to `TRUE` to include edges from and to the same node, or `"col"` to color the nodes according to the strength of diagonal entries. Note that it is advisable to only use standardized statistics (e.g., correlations instead of covariances) because otherwise the graphs can become hard to interpret.

Significance

Often a researcher is interested in the significance of a statistic (p value) rather than the value of the statistic itself. Due to the strict cutoff nature of significance levels, the usual representation then might not be adequate because small differences (e.g., the difference between edges based on t statistics of 1.9 and 2) are hard to see.

In *qgraph* statistical significance can be visualized with a weights matrix that contains p values and assigning `"sig"` to the `mode` argument. Because these values are structurally different from the edge weights we have used so far, they are first transformed according to the following function:

$$w_i = 0.7(1 - p_i)^{\log_{0.95} \frac{0.4}{0.7}}$$

where w_i is the edge weight of edge i and p_i the corresponding significance level. The resulting graph shows different levels of significance in different shades of blue, and omits any insignificant value. The levels of significance can be specified with the `alpha` argument. For a black and white representations, the `gray` argument can be set to `TRUE`.

For correlation matrices the *fdrtool* package (Strimmer., 2011) can be used to compute p values of a given set of correlations. Using a correlation matrix as input the `graph` argument should be set to `"sig"`, in which case the p values are computed and a graph is created as if `mode="sig"` was used. For the Big 5 data, a significance graph can be produced through the following code:

```
qgraph(cor(big5), groups = big5groups, vsize = 2,
       graph = "sig", alpha = c(1e-04, 0.001, 0.01))
```

Factor Loadings

A factor-loadings matrix contains the loadings of each item on a set of factors obtained through factor analysis. Typical ways of visualizing such a matrix is to boldface factor loadings that exceed, or omit factor loadings below, a given cutoff score. With such a method smaller, but interesting, loadings might easily be overlooked. In *qgraph*, factor-loading matrices can be visualized in a similar way as correlation matrices: by using the factor loadings as edge weights in a network. The function for this is |qgraph.loadings()— which uses the factor-loadings matrix to create a weights matrix and a proper layout and sends that information to `qgraph()`.

There are two functions in *qgraph* that perform an exploratory analysis based on a supplied correlation (or covariance) matrix and send the results to |qgraph-.loadings()—. The first is `qgraph.efa()` which performs an exploratory factor analysis (EFA; Stevens, 1996) using `factanal()` (*stats*; R Core Team, 2016). This function requires three arguments plus any additional argument that will be sent to |qgraph.loadings()— and `qgraph()`. The first argument must be a correlation or covariance matrix, the second the number of factors to be extracted and the third the desired rotation method.

To perform an EFA on the Big 5 dataset we can use the following code:

```
qgraph.efa(big5, 5, groups = big5groups, rotation = "promax",
     minimum = 0.2, cut = 0.4, vsize = c(1, 15),
     borders = FALSE, asize = 0.07, esize = 4, vTrans = 200)
```

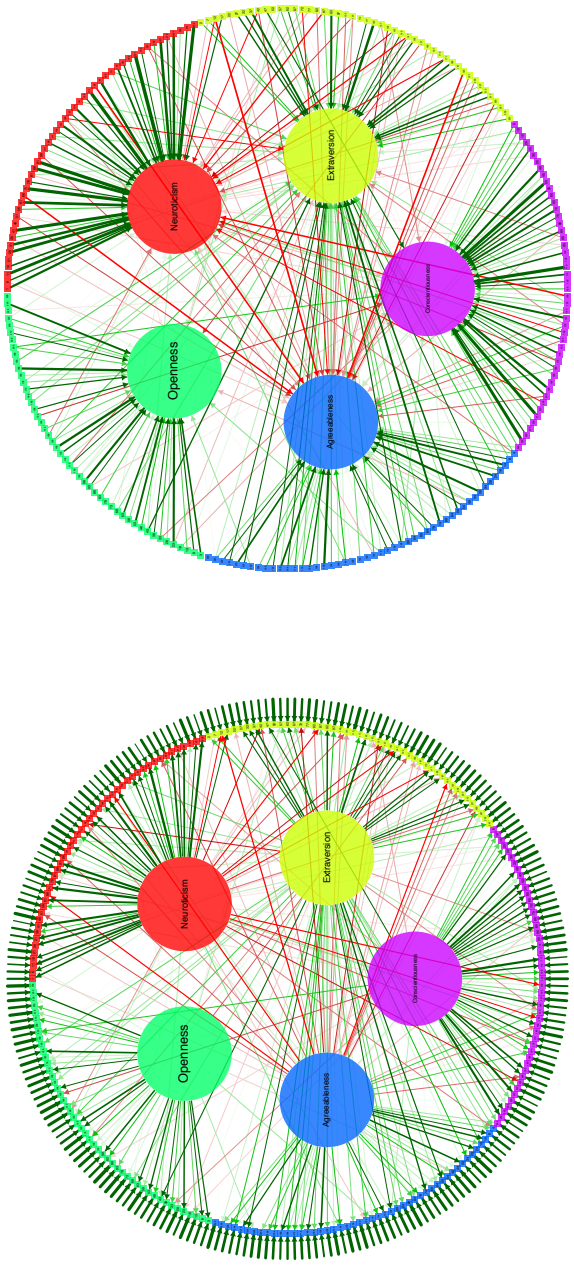
Note that we supplied the `groups` list and that we specified a *promax* rotation allowing the factors to be correlated.

The resulting graph is shown in the left panel of Figure 9.5. The factors are placed in an inner circle with the variables in an outer circle around the factors[10]. The factors are placed clockwise in the same order as the columns in the loadings matrix, and the variables are placed near the factor they load the highest on. Because an EFA is a *reflective measurement model*, the arrows point towards the variables and the graph has residuals (Bollen & Lennox, 1991; Edwards & Bagozzi, 2000).

The left panel of Figure 9.5 shows that the Big 5 dataset roughly conforms to the 5 factor model. That is, most variables in each group of items tend to load on the same factor. However, we also see many crossloadings, indicating departures from simple structure. Neuroticism seems to be a strong factor, and most crossloadings are between extraversion and agreeableness.

The second function that performs an analysis and sends the results to |qgraph-.loadings()— is `qgraph.pca()`. This function performs a principal component analysis (PCA; Jolliffe, 2002) using `princomp()` of the *psych* package (Revelle, 2010). A PCA differs from an EFA in that it uses a *formative measurement model*

[10]For a more traditional layout we could set `layout="tree"`



(a) (b)

Figure 9.5: Visualization of an exploratory factor analysis (a) and a principal component analysis (b) in the NEO-PI-R dataset.

(i.e., it does not assume a common cause structure). It is used in the same way as `qgraph.efa()`; we can perform a PCA on the big 5 dataset using the same code as with the EFA:

```
qgraph.pca(big5, 5, groups = big5groups, rotation = "promax",
     minimum = 0.2, cut = 0.4, vsize = c(1, 15),
     borders = FALSE, asize = 0.07, esize = 4, vTrans = 200)
```

The right panel of Figure 9.5 shows the results. Notice that the arrows now point towards the factors, and that there are no residuals, as is the case in a formative measurement model[11]. Note that the correlations between items, which are not modeled in a formative model, are omitted from the graphical representation to avoid clutter.

Confirmatory Factor Analysis

Confirmatory factor models and regression models involving latent variables can be tested using structural equation modeling (SEM; Bollen, 1989; Pearl, 2000). SEM can be executed in R with three packages: *sem* (Fox, 2006), *OpenMx* (Boker et al., 2011) and *lavaan* (Rosseel, 2012). *qgraph* currently supports *sem* and *lavaan*, with support for *OpenMx* expected in a future version. The output of *sem* (a `"sem"` object) can be sent to (1) `qgraph()` for a representation of the standardized parameter estimates, (2) `qgraph.semModel()` for a path diagram of the specified model, and (3) `qgraph.sem()` for a 12 page pdf containing fit indices and several graphical representations of the model, including path diagrams and comparisons of implied and observed correlations. Similarly, the output of *lavaan* (a `"lavaan"` object) can be sent to `qgraph()` or `qgraph.lavaan()`.

SEM is often used to perform a confirmatory factory analysis (CFA; Stevens, 1996) in which variables each load on only one of several correlated factors. Often this model is identified by either fixing the first factor loading of each factor to 1, or by fixing the variance of each factor to 1. Because this model is so common, it should not be necessary to fully specify this model for each and every run. However, this is currently still the case. Especially using *sem* the model specification can be quite long.

The `qgraph.cfa()` function can be used to generate a CFA model for either the *sem* package or the *lavaan* package and return the output of these packages for further inspection. This function uses the `groups` argument as a measurement model and performs a CFA accordingly. The results can be sent to another function, and are also returned. This is either a `"sem"` or `"lavaan"` object which can be sent to `qgraph()`, `qgraph.sem()`, `qgraph.lavaan()` or any function that can handle the object. We can perform the CFA on our dataset using *lavaan* with the following code:

```
fit <- qgraph.cfa(cov(big5), N = nrow(big5),
     groups = big5groups, pkg = "lavaan",
     opts = list(se = "none"), fun = print)
```

[11] In `qgraph.loadings` there are no arrows by default, but these can be set by setting the `model` argument to `"reflective"` or `"formative"`.

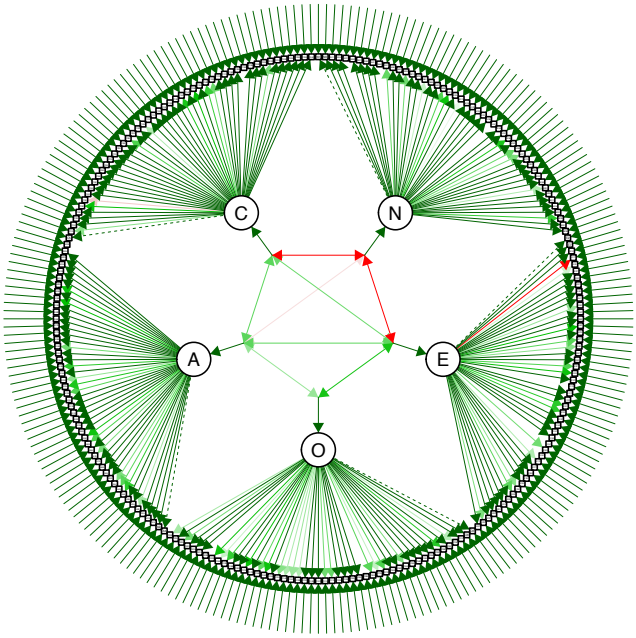


Figure 9.6: Standardized parameter estimations of a confirmatory factor analysis performed on the NEO-PI-R dataset.

```
lavaan (0.4-11) converged normally after 128 iterations

  Number of observations                           500

  Estimator                                         ML
  Minimum Function Chi-square                60838.192
  Degrees of freedom                             28430
  P-value                                        0.000
```

Note that we did not estimate standard errors to save some computing time. We can send the results of this to `qgraph.lavaan` to get an output document.

Figure 9.6 shows part of this output: a visualization of the standardized parameter estimates. We see that the first loading of each factor is fixed to 1 (dashed lines) and that the factors are correlated (bidirectional arrows between the factors). This is the default setup of `qgraph.cfa()` for both *sem* and *lavaan*[12]. From

[12]Using *lavaan* allows to easily change some options by passing arguments to `cfa()` using the `opts` argument. For example, we could fix the variance of the factors to 1 by specifying

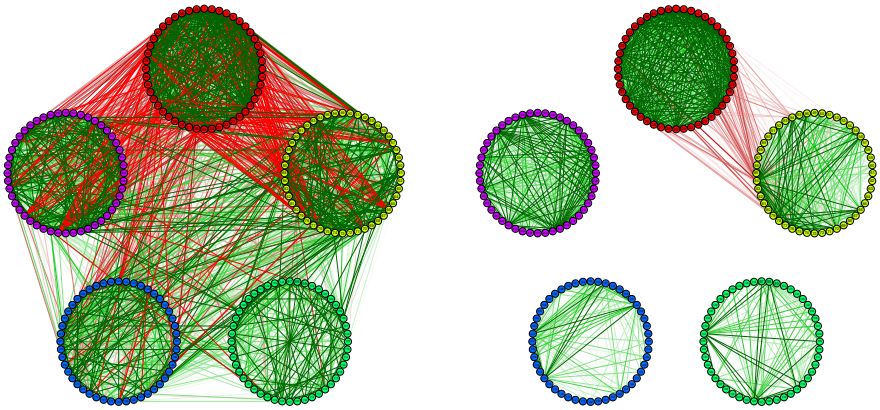


Figure 9.7: The observed correlations in the NEO-PI-R dataset (left) and the correlations that are implied by the model of Figure 9.6 (right).

the output above, we see that this model does not fit very well, and inspection of another part of the output document shows why this is so: Figure 9.7 shows a comparison of the correlations that are implied by the CFA model and the observed correlations, which indicates the source of the misfit. The model fails to explain the high correlations between items that load on different factors; this is especially true for extraversion and agreeableness items. The overlap between these items was already evident in the previous figures, and this result shows that this overlap cannot be explained by correlations among the latent factors in the current model.

9.4 Conclusion

The network approach offers novel opportunities for the visualization and analysis of vast datasets in virtually all realms of science. The *qgraph* package exploits these opportunities by representing the results of well-known statistical models graphically, and by applying network analysis techniques to high-dimensional variable spaces. In doing so, *qgraph* enables researchers to approach their data from a new perspective.

qgraph is optimized to accommodate both inexperienced and experienced R users: The former can get a long way by simply applying `qgraph.panel()` to a correlation matrix of their data, while the latter may utilize the more complex functions in *qgraph* to represent the results of time series modeling. Overall, however, the package is quite accessible and works with carefully chosen defaults,

`qgraph.cfa(\ldots,opts=list(std.lv=TRUE))`.

so that it almost always produces reasonable graphs. Hopefully, this will allow the network approach to become a valuable tool in data visualization and analysis.

Since *qgraph* is developed in a psychometric context, its applications are most suitable for this particular field. In this chapter we have seen that *qgraph* can be used to explore several thousands of correlations with only a few lines of code. This resulted in figures that not only showed the structure of these correlations but also suggested where exactly the five-factor model did not fit the data. Another example is the manual of a test battery for intelligence (IST; Liepmann, Beauducel, Brocke, & Amthauer, 2010) in which such graphs were used to argue for the validity of the test. Instead of examining full datasets *qgraph* can also be used to check for statistical assumptions. For example, these methods can be used to examine multicollinearity in a set of predictors or the local independence of the items of a subtest.

Clearly, we are only beginning to scratch the surface of what is possible in the use of networks for analyzing data, and the coming years will see considerable developments in this area. Especially in psychometrics, there are ample possibilities for using network concepts (such as centrality, clustering, and path lengths) to gain insight in the functioning of items in psychological tests.