Making graphs in R
Using the qgraph package

Sacha Epskamp

University of Amsterdam
Department of Psychological Methods

15-10-2013
All codes in these slides were run using R version 3.1.0 (2014-04-10) and qgraph version 1.2.5 and were made on Windows 7 x64 x86-64 build 9200.

Get the latest version of R from www.r-project.org and the latest version of qgraph from CRAN:

```r
install.packages("qgraph", dep=TRUE)
```
Make sure you can load qgraph:

```r
library("qgraph")
```

And that you have version 1.2.3 or higher:

```r
packageDescription('qgraph')$Version
```

## [1] "1.2.5"
If this fails, make sure you have the latest (2.15) version of R and that all depended/imported/suggested packages are installed (see CRAN).
Note that the following defaults are set for this presentation:

```r
options(
  qgraph = list(
    border.width = 2,
    asize = 8,
    unCol = "black",
    vsize = 10,
    esize = 3)
)
```

So the codes can create different looking graphs on your screen!
Help on R

Do it yourself . . .

- For basic understanding of R: Read through a R manual!
  - How do I make a matrix?
  - How do I index an object?
  - What is a list?
  - Try a short one first! (R for beginners)

- Help on how to use a function: Use the ? function (e.g. ?matrix)
  - How do I define a matrix by row?
  - How do I set mean() to omit NA’s?

- Find a certain function: Use the ?? function
  - What is a function to reduce a string to a certain amount of characters?
    - ??trim

- Or use google!
Help on R

... or ask for help!

▶ Stackoverflow websites (see next slide)
▶ For problems concerning specific packages: Mail the maintainer
▶ For short questions, you can use Twitter or Google+ with hashtag #rstats
Stackexchange

Stackexchange is a series of free question and answer websites on many different topics. Two are very useful for whenever you get stuck in R:

For programming technical questions regarding R see:
http://stackoverflow.com/

For statistical questions regarding R see:
http://crossvalidated.com/

In both of these make sure you use the tag `r` and include a reproducible example:
http://stackoverflow.com/q/5963269/567015
Graphs

- A graph is a *network* that consists of *n* nodes (or vertices) that are connected with *m* edges.
- Each edge can have a *weight* indicating the strength of that connection.
- An edge can be directed (have an arrow) or undirected.
The `qgraph()` function

- The main function in `qgraph` is `qgraph()`
  - Most other functions are either wrapping functions using `qgraph()` or functions used in `qgraph()`
- The `qgraph()` function requires only one argument (`input`)
- A lot of other arguments can be specified, but these are all optional

**Usage:**

```
qgraph( input, ... )
```
Weights matrices

- The input argument is the input. This can be a weights matrix.

- A weights matrix is a square $n$ by $n$ matrix in which each element indicates the relationship between two variables.

- Any relationship can be used as long as:
  - A 0 indicates no relationship
  - Absolute negative values are similar in strength to positive values

- We will first look at unweighted graphs, in which case the weights matrix is the same as an adjacency matrix.
  - A 1 indicates a connection
  - A 0 indicates a connection
  - Rows indicate the node of origin
  - Columns indicate the node of destination
  - By default the diagonal is omitted
  - By default, a symmetrical weights matrix is interpreted as an unweighted graph
Weights matrices

```r
input <- matrix(c(
  0,1,1,
  0,0,1,
  0,0,0),3,3,byrow=TRUE)
print(input)
#>
#>     [,1] [,2] [,3]
#> [1,]   0   1   1
#> [2,]   0   0   1
#> [3,]   0   0   0
```

`qgraph(input)`
Weights matrices

Exercise: Create this graph

The layout should be right automatically, only use one argument in `qgraph()`
Weights matrices

To make this graph, we need this matrix:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[2,]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[3,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[4,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[5,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[6,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>[7,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>[8,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Weights matrices

These matrices become quite large, so manually defining the matrix is not effective. So some tricks are needed to make the matrix:

```r
input <- matrix(0,8,8)
input[1,2] <- 1
input[2,3] <- 1
input[3,4] <- 1
input[4,5] <- 1
input[5,6] <- 1
input[6,7] <- 1
input[7,8] <- 1
input[8,1] <- 1
```
Weights matrices

```r
print(input)

# [1,] 0 1 0 0 0 0 0 0       # [2,] 0 0 1 0 0 0 0 0       # [3,] 0 0 0 1 0 0 0 0       # [4,] 0 0 0 0 1 0 0 0       # [5,] 0 0 0 0 0 1 0 0       # [6,] 0 0 0 0 0 0 1 0       # [7,] 0 0 0 0 0 0 0 1       # [8,] 1 0 0 0 0 0 0 0
```
Weights matrices

You can also change matrices manually (doesn’t work in RStudio):

```r
input <- matrix(0, 8, 8)
fix(input)
```

Or read the matrix from a text file!
Weights matrices

First make the matrix in a spreadsheet program (here LibreOffice)

```
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Weights matrices

Next save as or export
Weights matrices

Save as CSV (comma delimited text file) or tab delimited:
Weights matrices

Read in R (for tab delimited use `read.table()`):

```r
input <- read.csv("adj.csv", header=FALSE)
print(input)
```

```
## V1 V2 V3 V4 V5 V6 V7 V8
## 1 0 1 0 0 0 0 0 0
## 2 0 0 1 0 0 0 0 0
## 3 0 0 0 1 0 0 0 0
## 4 0 0 0 0 1 0 0 0
## 5 0 0 0 0 0 1 0 0
## 6 0 0 0 0 0 0 1 0
## 7 0 0 0 0 0 0 0 1
## 8 1 0 0 0 0 0 0 0
```
Weights matrices

These methods are not **reproducible**. Scripts should not depend on manual input. An easy way to change this is to first define a matrix, then run `dput()` on the object and use that result in your script:

```r
dput(input)
## structure(c(0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0), .Dim = c(8L, 8L))
```
Weights matrices

```r
input2 <- structure(c(0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1), .Dim = c(8L, 8L))

print(input2)
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[2,]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[3,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[4,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[5,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[6,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>[7,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>[8,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Weights matrices

Exercise: Create this graph
Weights matrices

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[2,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[3,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[4,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[5,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[6,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[7,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[8,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The directed argument

- The directed argument can be used to force a directed (TRUE) or undirected (FALSE) graph
- This can also be specified per edge in a matrix
The **directed argument**

```r
input <- matrix(1, 3, 3)
print(input)
```

```
[,1] [,2] [,3]
[1,] 1 1 1
[2,] 1 1 1
[3,] 1 1 1
```

```r
qgraph(input)
```
The directed argument

```r
print(input)
```

```r
[,1] [,2] [,3]
[1,] 1 1 1
[2,] 1 1 1
[3,] 1 1 1
```

```r
qgraph(input, directed=TRUE)
```
The directed argument

```r
print(input)

## [,1] [,2] [,3]
## [1,] 1 1 1
## [2,] 1 1 1
## [3,] 1 1 1

dir <- matrix(c(
  FALSE, TRUE, FALSE,
  TRUE, FALSE, FALSE,
  FALSE, FALSE, FALSE),
  3, 3, byrow=TRUE)
print(dir)

## [,1] [,2] [,3]
## [1,] FALSE TRUE FALSE
## [2,] TRUE FALSE FALSE
## [3,] FALSE FALSE FALSE

qgraph(input, directed=dir)
```

1
2
3
1
2
3
1
2
3
The **bidirectional** argument

- Multiple directed edges between two nodes are curved
- To change this behavior, `bidirectional` can be set to `TRUE`
- Can also be a matrix
The bidirectional argument

```
print(input)

## [,1] [,2] [,3]
## [1,] 1 1 1
## [2,] 1 1 1
## [3,] 1 1 1
```

```
qgraph(input, directed=TRUE, bidirectional=TRUE)
```
The bidirectional argument

```
print(input)
```

```
## [,1] [,2] [,3]
## [1,] 1 1 1
## [2,] 1 1 1
## [3,] 1 1 1
```

```
bidir <- matrix(c(  FALSE,TRUE,FALSE,  TRUE,FALSE,FALSE,  FALSE,FALSE,FALSE),3,3,byrow=TRUE)
print(bidir)
```

```
## [,1] [,2] [,3]
## [1,] FALSE TRUE FALSE
## [2,] TRUE FALSE FALSE
## [3,] FALSE FALSE FALSE
```

```
qgraph(input,  
directed=TRUE,  
bidirectional=bidir)
```
Arguments for directed graphs

- Two other arguments can be used this way:
  - `curve` to curve each edge
  - `lty` to create dashed lines (not yet in matrix form)

- And finally a few other arguments:
Arguments for directed graphs

```
input <- matrix(c(0,1,1, 1,0,1, 0,0,0),3,3,byrow=TRUE)
print(input)
```

```
# [1,] 0 1 1
# [2,] 1 0 1
# [3,] 0 0 0
```

```
qgraph(input, asize=10, arrows=2, open=TRUE, curvePivot=TRUE)
```
Weighted graphs

- Specify edge weights to make a graph weighted
  - In a weights matrix: simply specify other values than only zeros and ones
- An edge weight of 0 indicates no connection
- Positive and negative edge weights must be comparable in strength
- The “length” of an edge is defined as the inverse of the weight.
  - Stronger connected nodes are closer together
  - An edge weight of 0 indicates infinite length
Weighted graphs

```r
input <- matrix(c(0,1,2,
                  0,0,3,
                  0,0,0),3,3,byrow=TRUE)

print(input)
```

```
## [,1] [,2] [,3]
## [1,] 0  1  2
## [2,] 0  0  3
## [3,] 0  0  0
```
Layout modes

- The placement of the nodes is specified with the `layout` argument in `qgraph()`
- This can be a \( n \) by 2 matrix indicating the \( x \) and \( y \) position of each node
- `layout` can also be given a character indicating one of the two default layouts
  - If `layout="circular"` the nodes are placed in circles per group (if the `groups` list is specified)
  - If `layout="spring"` the Fruchterman Reingold algorithm is used for the placement
- And a final option is to specify a grid-like layout
Layout matrix

```r
input <- matrix(c(1,3,3)
L <- matrix(c( 0,1, 1,1, 0.5,0),
ncol=2,byrow=TRUE)
print(L)
```

```
## [,1] [,2]
## [1,] 0.0 1
## [2,] 1.0 1
## [3,] 0.5 0
```

```r
qgraph(input, layout=L)
```
Layout matrix

```r
L <- matrix(c(0,1, 1,1, 0,0), ncol=2, byrow=TRUE)
print(L)
```

```
## [,1] [,2]
## [1,] 0 1
## [2,] 1 1
## [3,] 0 0
```

```r
cgraph(input, layout=L)
```

![Graph diagram](image)
Layout matrix

- With the layout matrix the actual layout can be specified
- The scale is not relevant
- `qgraph()` returns a list containing everything needed to make the graph
- This can be used to force another graph based on the layout of the first

```
Q <- qgraph(input)
qgraph(input2, layout=Q$layout)
```
Grid layout

```r
input <- matrix(c(1, 3, 3)
L <- matrix(c(1, 0, 2,
0, 0, 0,
0, 3, 0),
nrow=3, byrow=TRUE)
print(L)
```

```
## [,1] [,2] [,3]
## [1,] 1 0 2
## [2,] 0 0 0
## [3,] 0 3 0
```

```r
qgraph(input, layout=L)
```
Grid layout

```
input <- matrix(1, 3, 3)
L <- matrix(c(1, 0, 2,
              0, 0, 0,
              3, 0, 0), nrow=3, byrow=TRUE)
print(L)
```

```
# [1,] 1 0 2
# [2,] 0 0 0
# [3,] 3 0 0
```

`qgraph(input, layout=L)`
Fruchterman-Reingold layout

- `layout="spring"` uses a force-embedded algorithm (the Fruchterman-Reingold algorithm)
- This is an iterative algorithm.
- The initial layout is a circle
- Then in each iteration:
  - Each node is repulsed by all other nodes
  - Connected nodes are also attracted to each other
  - The maximum displacement weakens each iteration
- After this process the layout is rescaled to fit the \(-1\) to \(1\) \(xy\)-plane
- The unscaled layout is returned as `layout.orig`
Big 5

Load the big 5 dataset:

```r
data(big5)
str(big5)
```

```r
## num [1:500, 1:240] 2 3 4 4 5 2 2 1 4 2 ...
## - attr(*, "dimnames")=List of 2
## ..$ : NULL
## ..$ : chr [1:240] "N1" "E2" "O3" "A4" ...
```
Big 5

\texttt{qgraph(corr(big5), minimum=0.25)}
The groups argument

- The groups indicates which nodes belong together
- Nodes belonging together are...
  - placed in smaller circles (with circular layout)
  - colored in the same color (either rainbow or defined with color)
- Names in the groups can be used as legend
- groups can even be used to perform a oneline CFA with qgraph.cfa()

Either use a factor (a vector with characters) or a list in which each element is a vector containing the number of nodes that belong together
The `groups` argument

# List:
```r
groups <- list(A = c(1,2,3,4,5),
               B = c(6,7,8,9,10))
```

# Factor:
```r
groups <- c("A","A","A","A","A",
             "B","B","B","B","B")
```

# Result:
```r
qgraph(matrix(1,10,10), groups=groups)
```
data(big5groups)
big5graph <- qgraph(cor(big5), minimum=0.25, groups=big5groups)
Big 5

qgraph (big5graph, layout="spring")

- Neuroticism
- Extraversion
- Openness
- Agreeableness
- Conscientiousness
qgraph graphs can not be manually rescaled, and hence the RStudio Export function can not be used to save qgraph graphs.

For the best result, save graphs in a PDF device!
Export to PDF

```r
# Open a pdf device:
pdf("nameoffile.pdf")

# Plot stuff:
qgraph(1)

# Close pdf device:
dev.off()

## pdf
## 2

(If you get faulty output, make sure to run dev.off() enough times until R returns Null Device)
Export to PNG

```r
# Open a pdf device:
png("nameoffile.png")

# Plot stuff:
qgraph(1)

# Close pdf device:
dev.off()

## pdf
## 2

(If you get faulty output, make sure to run dev.off() enough times until R returns Null Device)
Important qgraph arguments

- **minimum**: Omits edge weights with absolute values under this argument
- **maximum**: Sets the strongest edge to scale to
  - **cut**: Splits scaling of color and width
- **vsize**: Sets the size of nodes
- **esize**: Sets the size of edges
- **asize**: Sets the size of arrows
- **filetype**: Type of file to save the plot to
- **filename**: Name of the file to save the plot to
Contribute to qgraph

The developmental version of qgraph can be found on GitHub (https://github.com/SachaEpskamp/qgraph) and can be installed using devtools

```r
library("devtools")
install_github("qgraph","sachaepskamp")
```

If you have any ideas on concepts to implement in qgraph or encounter any bugs please post them on GitHub!