Drawing Networks

Using qgraph

PNAWS 2019

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

```
install.packages("qgraph",dep=TRUE)
```

Make sure you can load qgraph:

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

And that you have version 1.6 or higher:

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

```
## [1] "1.6"
```

If this fails, make sure you have the latest (3.5.2) version of R and that all depended/imported/suggested packages are installed (see CRAN).

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

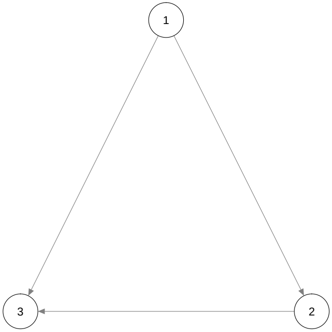
- The `input` argument is the input. This can be a *weights matrix*
- A weights matrix is a square *n* by *n* matrix in which each element indicates the relationship between two variables
- Any relationship can be used as long as:
  - A 0 indicates no relationship
  - Absolute negative values are similar 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 symetrical weights matrix is interpreted as an unweighted graph

Weights matrices

```r
qgraph(input)
```

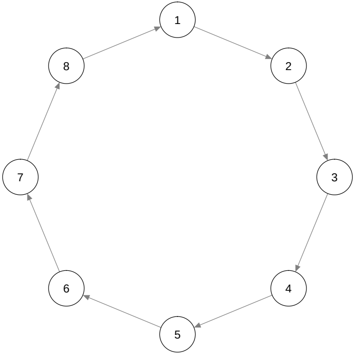
```r
input <- matrix(c(
         0,1,1,
         0,0,1,
         0,0,0),3,3,byrow=TRUE)
print(input)
```

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



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:

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [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 matrices become quite large, so manually defining the matrix is not effective. So some tricks are needed to make the matrix:

```
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

```
print(input)

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [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

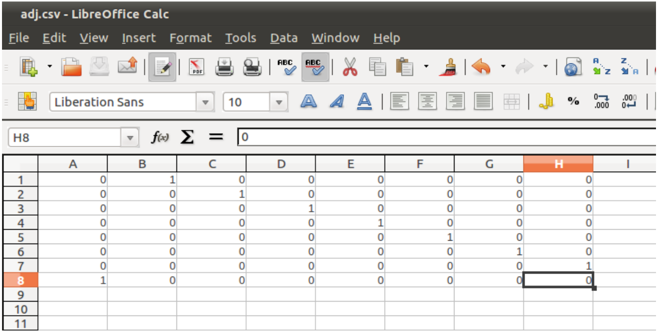
You can also change matrices manually (doesn't work in RStudio):

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

Or read the matrix from a text file!

Weights matrices

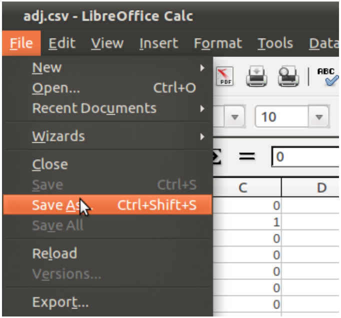
First make the matrix in a spreadsheet program (here LibreOffice)



	A	B	C	D	E	F	G	H	I
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	
9									
10									
11									

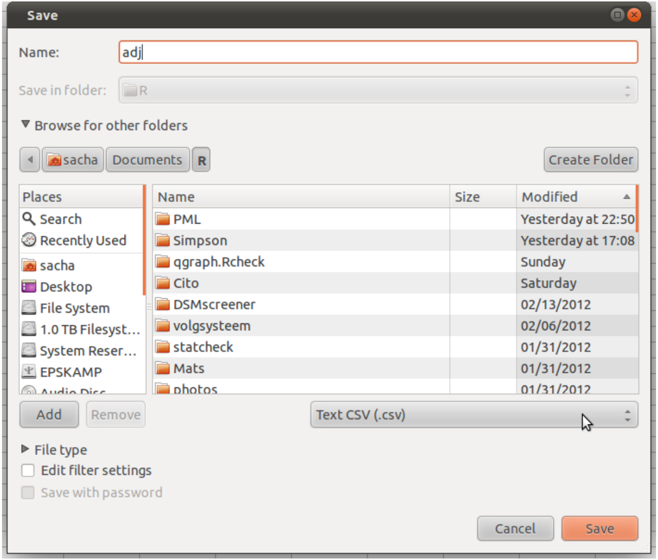
Weights matrices

Next save as or export



Weights matrices

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



Weights matrices

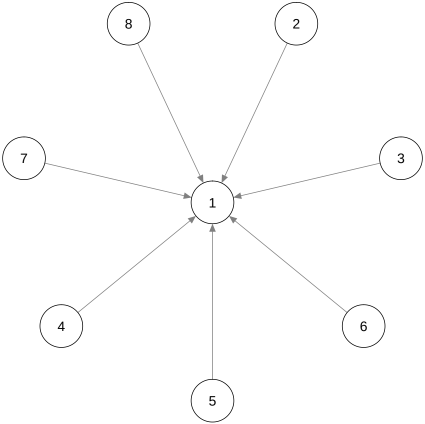
Read in R (for tab delimited use `read.table()`):

```
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

Exercise: Create this graph



Weights matrices

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

Weighted graphs

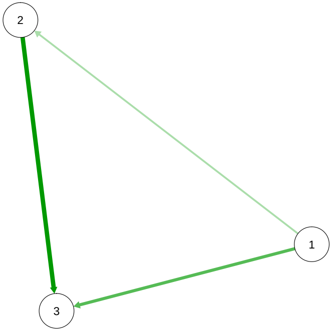
- Specify edge weights to make a graph weighted
  - In a weights matrix: simply specify other values than only zeros and ones
- An edge weigth of 0 indicates no connection
- Positive and negative edge weights must be comparable in strength

Weighted graphs

```
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
```

```
qgraph(input)
```

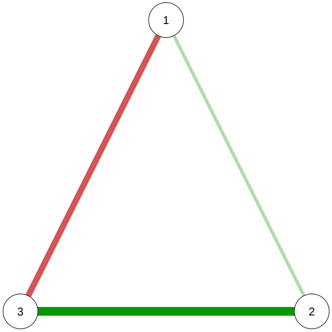


Weighted graphs

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

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

```
qgraph(input)
```

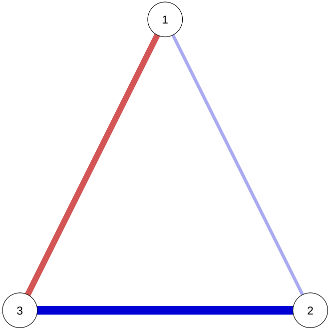


Weighted graphs

```r
qgraph(input,
  theme = "colorblind")
```

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

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



Interpreting **qgraph**

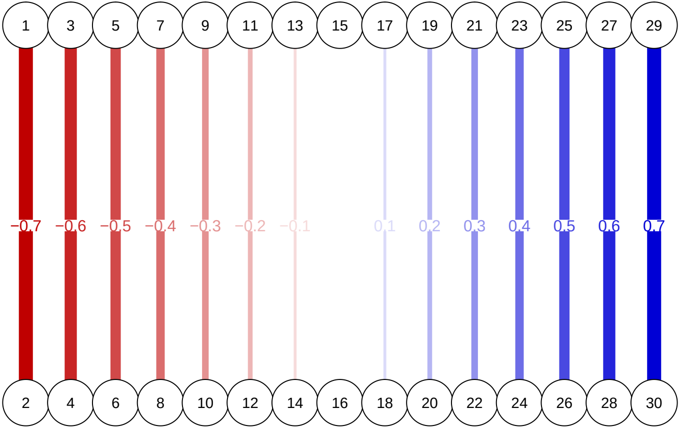
- Under the default coloring scheme, positive edge weights (here correlations) are shown as green (colorblind theme: blue) edges and negative edge weights as red.
- An edge weight of 0 is omitted. The wider and more colorful an edge the stronger the edge weight.

To interpret **qgraph** networks, three values need to be known:

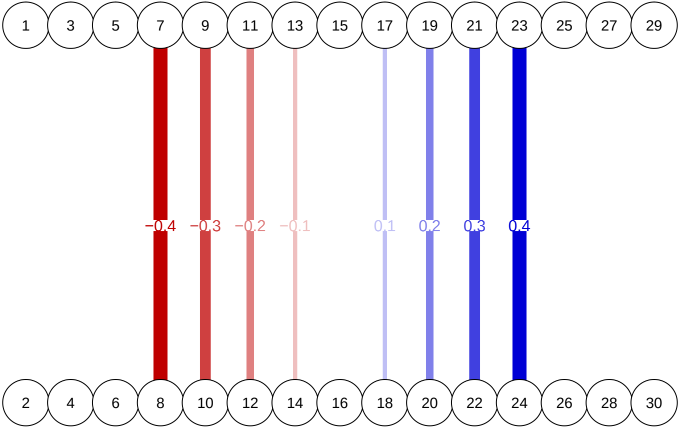
Minimum
: Edges with absolute weights under this value are omitted

Cut
: If specified, splits scaling of width and color

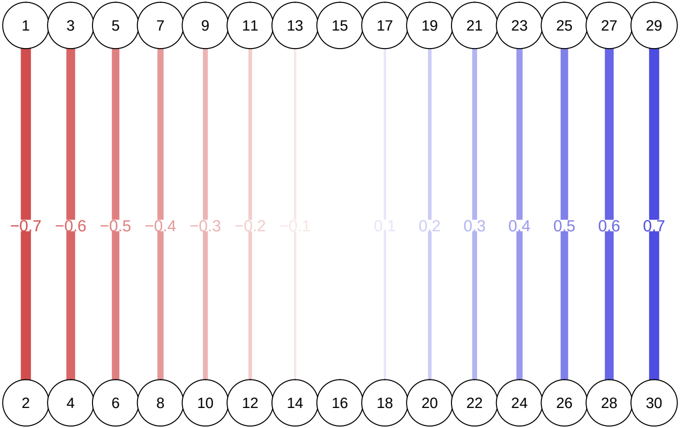
Maximum
: If set, edge width and color scale such that an edge with this value would be the widest and most colorful



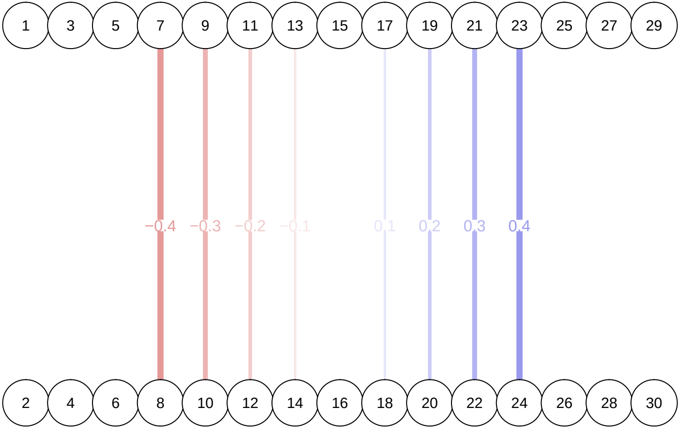
No minimum, maximum or cut



No minimum, maximum or cut

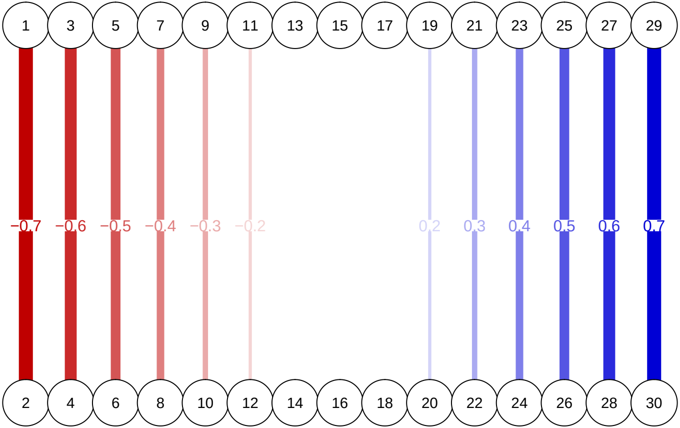

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
−0.7 −0.6 −0.5 −0.4 −0.3 −0.2 −0.1 0.1 0.2 0.3 0.4 0.5 0.6 0.7
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

Maximum 1

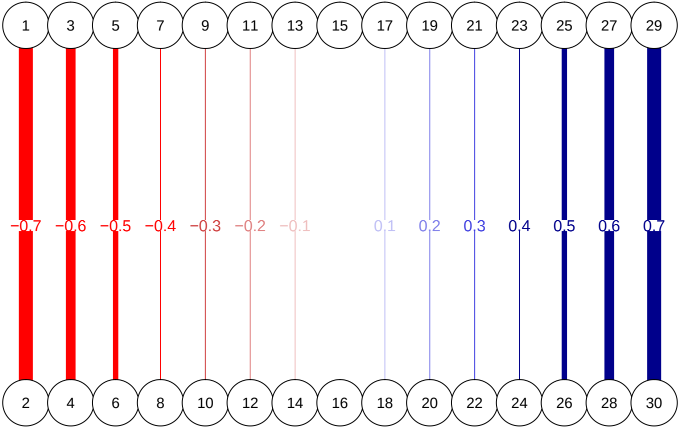

1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
−0.4
−0.3
−0.2
−0.1
0.1
0.2
0.3
0.4
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30

Maximum 1

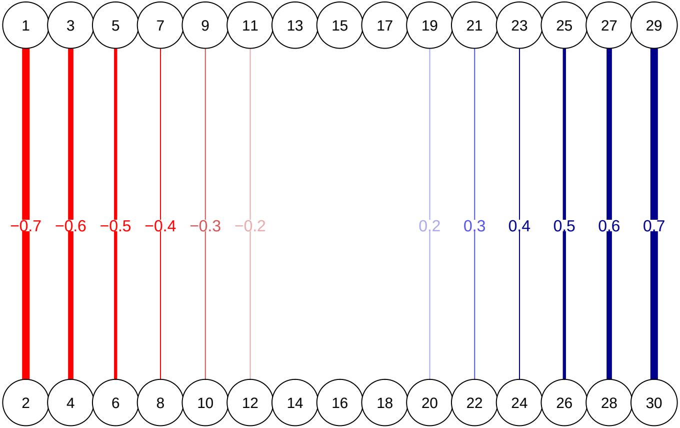
Maximum must be set to make graphs comparable!


1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
−0.7
−0.6
−0.5
−0.4
−0.3
−0.2
0.2
0.3
0.4
0.5
0.6
0.7
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30

Minimum 0.1


1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
−0.7
−0.6
−0.5
−0.4
−0.3
−0.2
−0.1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30

Cut 0.4



Minimum	0.1
Cut	0.4
Maximum	1

By default, above 20 nodes the cut value is automatically chosen to be equal to the maximum of the 75th quantile of absolute edge strengths or the edge strength corresponding to 2n-th edge strength (n being the number of nodes).

Layout modes

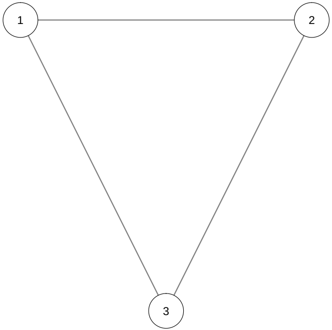
- The placement of the nodes is specified with the `layout` argument in `qgraph()`
- This can be a *n* by 2 matrix indicating the *x* and *y* position of each node
- `layout` can also be given a character indicating one of the two default layouts
  - If `layout="circular"` the nodes are placed in circles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

Layout matrix

```r
input <- matrix(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)
```

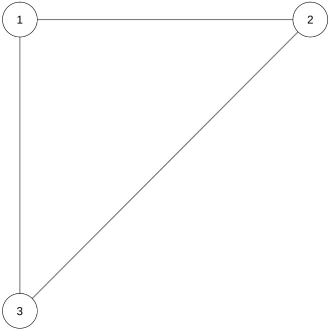


Layout matrix

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

```
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
```



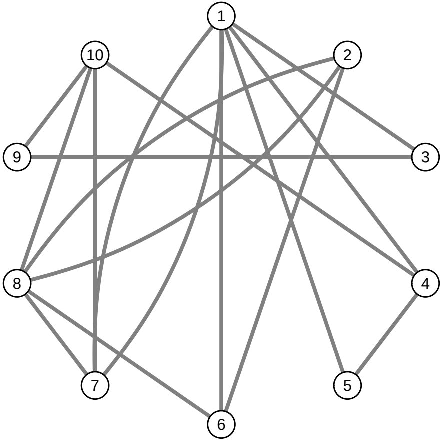
Layout matrix

- 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)
```

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


1
2
3
4
5
6
7
8
9
10

output

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!

Note that if a legend is used, the plot is made square by making the width 1.4 times the height of a plot

Export to PDF

```
# 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 untill R returns `Null Device`)

Export to PNG

```
# 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 untill 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
groups	Colors nodes and adds a legend
nodeNames	Adds a legend with specific names per node

Contribute to **qgraph**

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

```
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!

Thank you for your attention!