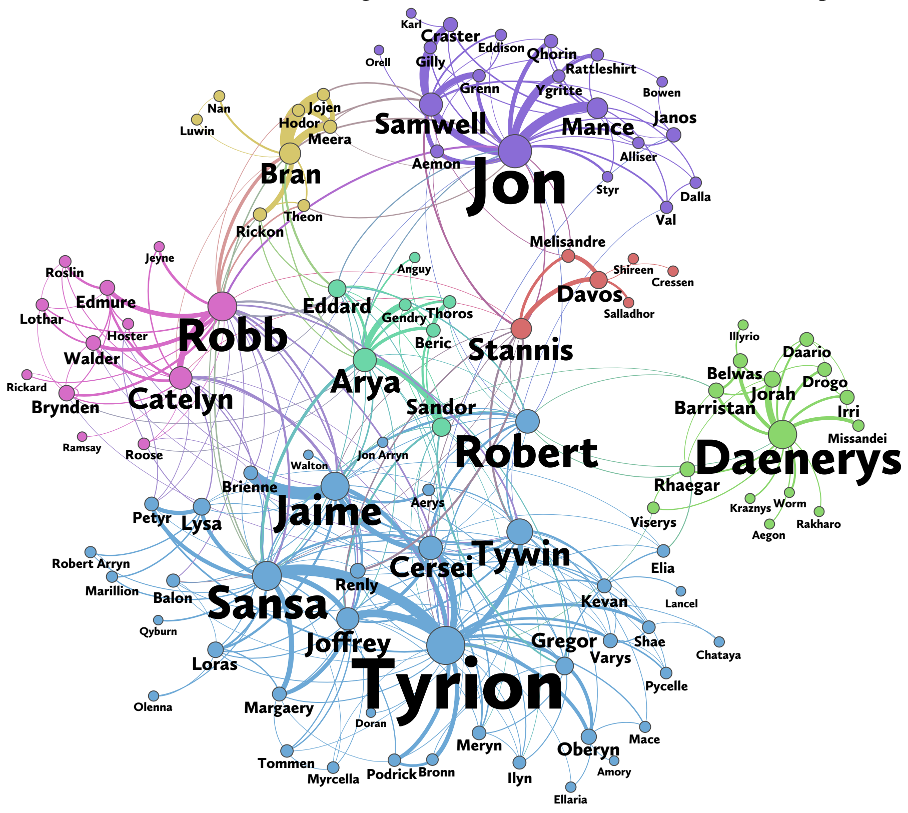
Constructing and drawing networks in *qgraph*

Network Visualization

In these exercises, we will analyze a weighted social network of book series *A Song of Ice and Fire*, on which the popular TV show *Game of Thrones* is based. This network was published in Math Horizons Magazine, and is based on the third book, A Storm of Swords, on which the third and fourth season of the TV series are based. More information, including the data, is available online. The network published is the following:



Exercise 1 Look through the paper to get an idea of what this network represents. What do the nodes and edges represent? ■

The data as published online can be loaded in R as follows (you can find the data file on the *Companion Website*:

```r
Data <- read.csv("stormofswords.csv")
```

Exercise 2 Look at the data in RStudio using the "View" function. This matrix encodes a network. Can you figure out how? What do the rows stand for and what do the columns stand for? ■

This structure is known as an *edgelist* encoding a network, which can also be used as input for qgraph:

```r
library("qgraph")
qgraph(Data, directed = FALSE)
```

Exercise 3 When plotting an undirected graph using an adjacency or weights matrix as input we normally do not have to set the `directed` argument. Now that we use an edgelist, however, I do. Why? ■

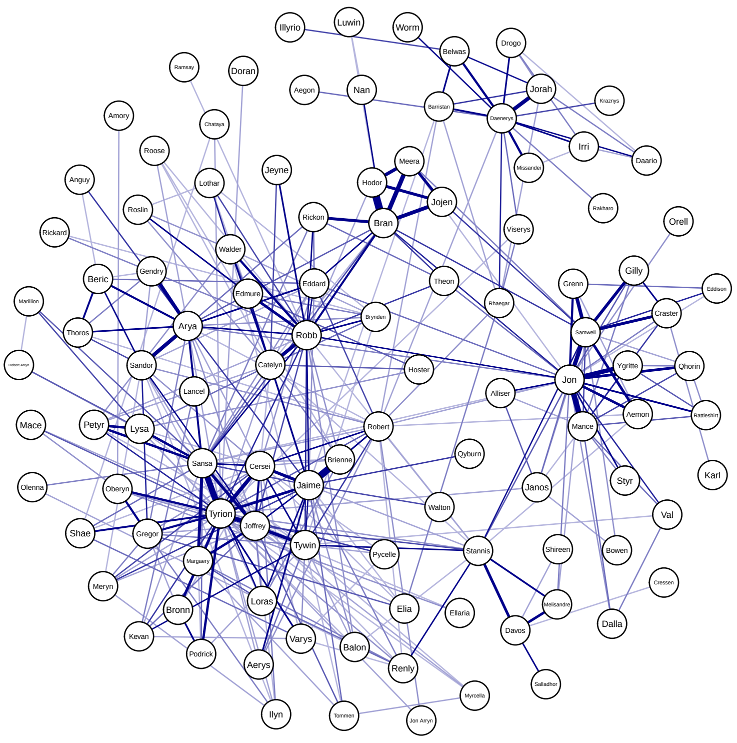
Now load the following dataset also available on the *Companion Website*:

```
Data2 <- read.csv("stormofswords_wmat.csv")
```

This dataset represents the same network, but in a different way. This representation is called a *weights matrix*.

Exercise 4 Investigate the new data using `View(Data2)`. Can you figure out how this weights matrix encodes a network. Use `Data2` as input to `qgraph()`. Do you obtain the same network? Do you still need to use the `directed` argument?

The plotted network is plotted using a circular layout. This circular layout, however, is hard to interpret and read in such networks with many nodes. In addition, we may prefer plotting edges with a different color than green when they represent values that can only be positive. Finally, the nodes are very large and we might want to make them smaller:



Exercise 5 Recreate the plot above, changing the layout to a spring layout, the edge color to "darkblue" and the node size to 3. Look at the qgraph help page to figure out the commands needed (`?qgraph`). Note: your computer might generate a different spring layout (nodes placed on different locations).

qgraph uses three arguments that need to be known to interpret a network: `minimum`, `cut`, and `maximum`. These are set automatically, and can be shown using the argument `details = TRUE`.

Exercise 6 What values did qgraph set to `cut` and `maximum`? Note: `minimum` is always set to 0 by default and not shown with `details = TRUE` unless it differs from 0.

The `minimum` argument can be used to *hide* edges with an absolute (negative edges are treated as positive) weight below some value. Note that these edges are only visually hidden, not removed in further analyses (which can be done using the `threshold` argument). This argument is useful when plotting dense graphs (e.g., correlation networks) but *not* recommended in the networks estimated in this textbook.

Exercise 7 Set the `minimum` argument to 1, 10 and 20 while using a spring layout. How does the network change? Do the same using the `threshold` argument. Can you explain why the layout remains the same using `minimum` but changes using `threshold`? ■

Edges drawn in qgraph are drawn more wider and more saturated the stronger the absolute edge weight is. In large networks, it might be useful to split the scaling of width and color. This is what the `cut` argument does: edges with an absolute edge weight under the cutoff will be drawn of the smallest width and vary only in color. Edges with an absolute edge weight over the cutoff value will always be drawn fully saturated (green or red by default), and will be drawn thicker the stronger they are. You can disable this behavior by setting `cut = 0`, which will ensure that edges all scale in color and width. qgraph automatically sets a cutoff value when there are 20 or more nodes in the network. We recommend setting `cut = 0` unless you specifically want this behavior.

Exercise 8 Set the `cut` argument to 50, 10 and 1 while using a spring layout. How does the network change? Now set the `cut` argument to 0. What happened? ■

The scaling of edges is chosen based on the strongest absolute edge in the network. This is because networks highly differ in their weights. This social network is based on the number of interactions. The characters "Bran" and "Hodor" interacted the most (96 times), leading to a strongest edge weight of 96. A network based on (partial) correlations, however, can never have a stronger edge weight than 1 (and usually features weaker edges). The `maximum` argument can be used to overwrite the "largest edge" to which the color and width of edges scale to. Its value is treated as the weight of an invisible edge, and is automatically set to the largest edge weight in the network. Setting `maximum` higher will make edges scale to that value instead of the strongest edge.

Exercise 9 Set the `maximum` argument to 200. What happened? Now set `maximum` back to its default and subsequently to 10. Why doesn't `maximum = 10` change the network from its default value, but `maximum = 200` does? ■

Exercise 10 What would be a good setting for `maximum` when drawing networks based on (partial) correlations? ■