Preliminary Network Visualization with R

Learning Outcomes

- Understanding Quarto and RMarkdown documents
- Understanding functions and arguments
- Loading a data set: `read_csv()`
- Peeking at a data set: `head()`, `tail()`, `glimpse()`
- Wrangling data frames with {tidyverse} commands: `filter()`, `rename()`
- Creating a data frame from scratch with field names: `data.frame()`
- Running functions from {visNetwork} and {igraph} and adjusting argument values: `visNetwork()`, `graph_from_data_frame()`, `E()`, `V()`
- Writing out as a .csv or .graphml file

Introduction

Today, we are going to do some preliminary exercises for working with network data using R and RStudio. You should use a "Quarto" or "RMarkdown" document for organizing your work and notes to get practice using that approach to doing "reproducible research". Both of these are simple text documents that allow you to mix together narrative text and code to generate formatted, dynamic output. It also allows you to easily manage notes and run code.

Quarto/RMarkdown and R Basics

- Use the **File > New File > Quarto Document...** or **File > New File > R Markdown...** command to create new Quarto or R Markdown documents.
- Use hashtags (**#**) to format headers for section titles. The more **#**'s, the smaller the header.
- Use syntax to format text
    - Wrap text in asterisks (e.g., *italics*) to create *italicized* text
    - Wrap text in double asterisks (e.g., **bold**) to create **bold** text
    - Wrap text in carats (e.g., ^superscript^) to create ^{superscript} text
    - Wrap text in tildes (e.g., ~subscript~^) to create _{subscript} text
    - Wrap text in double tildes (e.g., ~~strikethrough~~) to create ~~strikethrough~~ text
    - Wrap text in back ticks (e.g., `inline code`) to create `inline code`
- Chunks of R code are included in code blocks that start and end with three back ticks. The opening line of the code block should also include the designator {r}, which tells the rendering engine to run the code using R (as opposed to Python, etc.)

```
        print("this is a code block")
```

```
[1] "this is a code block"
```

- Run code by highlighting it in your document and hitting `command-RETURN` or `command-ENTER`. This sends the code to the R console and executes it.
- A whole block of code can be run by clicking the green right arrow at the top of the code block
- Comments can be included within R code blocks by prefacing them with #
- Use the **Render** (for Quarto documents) or **Knit** (for RMarkdown documents) to create nicely formatted reports based on your notes and code. These commands read through your text file and create an HTML or PDF file that incorporates your syntax, runs the code in your code blocks, and inserts the output of running that code into the report.

Today's Exercise

Load in a dataset

We will use a slightly different version of the dataset we were working with last time: social connections among characters in the *Game of Thrones* saga, but only from book one.

```
# read in edges and vertices as data frames/tibbles from .csv
# files using the {tidyverse} package
library(tidyverse)
```

```
f <- file.choose()
```

```
e <- read_csv(f, col_names = TRUE)
head(e)
```

```
# A tibble: 6 × 5
  Source                          Target             Type       weight  book
  <chr>                           <chr>              <chr>       <dbl> <dbl>
1 Addam-Marbrand                  Jaime-Lannister    Undirected      3     1
2 Addam-Marbrand                  Tywin-Lannister    Undirected      6     1
3 Aegon-I-Targaryen               Daenerys-Targaryen Undirected      5     1
4 Aegon-I-Targaryen               Eddard-Stark       Undirected      4     1
5 Aemon-Targaryen-(Maester-Aemon) Alliser-Thorne     Undirected      4     1
6 Aemon-Targaryen-(Maester-Aemon) Bowen-Marsh        Undirected      4     1
```

```
tail(e)
```

```
# A tibble: 6 × 5
  Source           Target          Type       weight  book
  <chr>            <chr>           <chr>       <dbl> <dbl>
1 Tyrion-Lannister Varys           Undirected      3     1
```

```
2 Tyrion-Lannister Willis-Wode     Undirected      4     1
3 Tyrion-Lannister Yoren           Undirected     10     1
4 Tywin-Lannister  Varys           Undirected      4     1
5 Tywin-Lannister  Walder-Frey     Undirected      8     1
6 Waymar-Royce     Will-(prologue) Undirected     18     1
```

```
        glimpse(e)
```

```
Rows: 684
Columns: 5
$ Source <chr> "Addam-Marbrand", "Addam-Marbrand", "Aegon-I-Targaryen", "Aegon…
$ Target <chr> "Jaime-Lannister", "Tywin-Lannister", "Daenerys-Targaryen", "Ed…
$ Type   <chr> "Undirected", "Undirected", "Undirected", "Undirected", "Undire…
$ weight <dbl> 3, 6, 5, 4, 4, 4, 9, 5, 13, 34, 5, 4, 10, 3, 5, 3, 12, 11, 6, 4…
$ book   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
```

Create a graph using the {visNetwork} package...

We will first use the visNetwork package to do a quick visualization of the network. Start by loading in the package...

```
        library(visNetwork)
```

{visNetwork} requires separate data frames for nodes and edges, so we create those from our dataset...

```
        # we first create a data frame with edges and weights as a simple
        # copy of the dataset we have loaded in
        edges <- e
```

Because this dataset is so large, we're going to limit it to only those edges with a high weight...

```
        # pick out only edges with high weight
        edges <- filter(e, weight >= 50)
```

From this, we then create a data frame of vertices by selecting all the unique names in either the "Source" or "Target" columns...

```
        vertices <- data.frame(
          id = unique(c(edges$Source,edges$Target)),
          label = unique(c(edges$Source, edges$Target))
          # the label argument allows us to display node names
        )
```

We then change some column names to meet some requirements for the `visNetwork()` function, e.g., we change "Source" and "Target" to "from" and "to" and we make the "Type" column start with a lowercase letter...

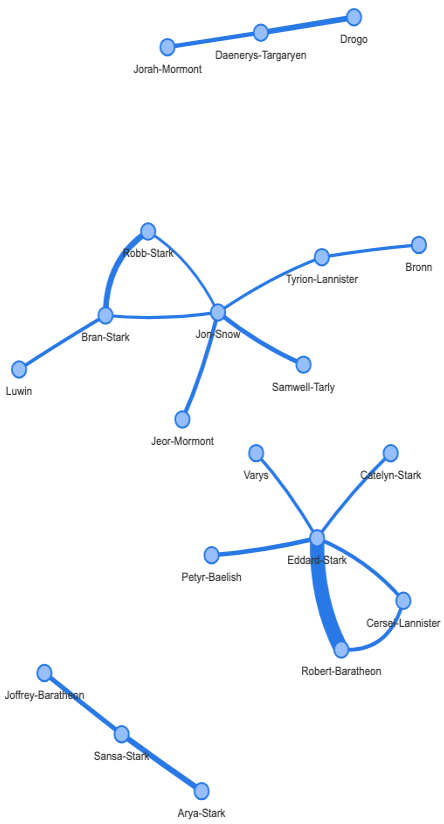
```
edges <- rename(edges, from = Source, to = Target, type = Type)
```

We also add new columns for “width”, “label”, and “title” to improve our visualization…

```
edges <- mutate(edges,
  width = weight / max(weight) * 20, # scale the width
  label = weight, # label the edges with their weights
  title = paste("Weight:", weight) # tooltip for the edges
)
```

Now, we use the `visNetwork()` function to create and visualize the network… Note that this creates an interactive graph where we can select and move vertices around!

```
# create the visNetwork graph
visNetwork(nodes = vertices, edges = edges) %>%
  visEdges(smooth = TRUE) %>%
  visNodes(size = 10) %>%
  # set a seed for reproducible layout
  visLayout(randomSeed = 42)
```



Questions

Looking at this graph, answer the following questions…

- How many **components** are in the graph?
- What is the **diameter** of the largest component?
- Which node(s) has the **highest degree**? What is that degree?
- If we initially filter our dataset by a higher edge weight (e.g., 60), how do these answers change? What about using a lower edge weight (e.g., 40)?

Create the same graph using the {igraph} package...

```
# Load the igraph package
library(igraph)

# use the `graph_from_data_frame()` function to create a graph from
# our data frames of edges and vertices
graph <- graph_from_data_frame(
  d = edges,
  vertices = vertices,
  directed = FALSE)
graph
```

```
IGRAPH fb6a42b UNW- 20 18 --
+ attr: name (v/c), label (v/c), type (e/c), weight (e/n), book (e/n),
| width (e/n), label (e/n), title (e/c)
+ edges from fb6a42b (vertex names):
 [1] Arya-Stark        --Sansa-Stark      Bran-Stark        --Jon-Snow
 [3] Bran-Stark        --Luwin            Bran-Stark        --Robb-Stark
 [5] Bronn             --Tyrion-Lannister Catelyn-Stark     --Eddard-Stark
 [7] Cersei-Lannister  --Eddard-Stark     Cersei-Lannister  --Robert-Baratheon
 [9] Daenerys-Targaryen--Drogo            Daenerys-Targaryen--Jorah-Mormont
[11] Eddard-Stark      --Petyr-Baelish    Eddard-Stark      --Robert-Baratheon
[13] Eddard-Stark      --Varys            Jeor-Mormont      --Jon-Snow
+ ... omitted several edges
```

We can use the {igraph} functions `E()` and `V()` to get information about the edges and vertices of a graph...

```
# show edge information
E(graph)
```

```
+ 18/18 edges from fb6a42b (vertex names):
 [1] Arya-Stark        --Sansa-Stark      Bran-Stark        --Jon-Snow
 [3] Bran-Stark        --Luwin            Bran-Stark        --Robb-Stark
 [5] Bronn             --Tyrion-Lannister Catelyn-Stark     --Eddard-Stark
 [7] Cersei-Lannister  --Eddard-Stark     Cersei-Lannister  --Robert-Baratheon
 [9] Daenerys-Targaryen--Drogo            Daenerys-Targaryen--Jorah-Mormont
[11] Eddard-Stark      --Petyr-Baelish    Eddard-Stark      --Robert-Baratheon
[13] Eddard-Stark      --Varys            Jeor-Mormont      --Jon-Snow
```

```
[15] Joffrey-Baratheon --Sansa-Stark      Jon-Snow          --Robb-Stark
[17] Jon-Snow          --Samwell-Tarly    Jon-Snow          --Tyrion-Lannister
```

```
E(graph)$type
```

```
 [1] "Undirected" "Undirected" "Undirected" "Undirected" "Undirected"
 [6] "Undirected" "Undirected" "Undirected" "Undirected" "Undirected"
[11] "Undirected" "Undirected" "Undirected" "Undirected" "Undirected"
[16] "Undirected" "Undirected" "Undirected"
```

```
E(graph)$weight
```

```
 [1] 104  56  65 112  61  64  69  72 101  75  81 291  61  81  87  53  81  56
```

```
E(graph)$width
```

```
 [1]  7.147766  3.848797  4.467354  7.697595  4.192440  4.398625  4.742268
 [8]  4.948454  6.941581  5.154639  5.567010 20.000000  4.192440  5.567010
[15]  5.979381  3.642612  5.567010  3.848797
```

```
# show vertex information
V(graph)
```

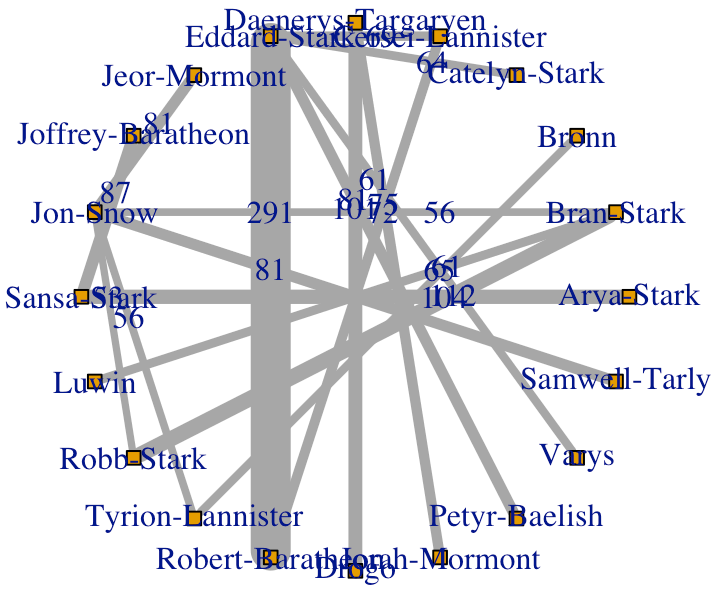
```
+ 20/20 vertices, named, from fb6a42b:
 [1] Arya-Stark         Bran-Stark         Bronn              Catelyn-Stark
 [5] Cersei-Lannister   Daenerys-Targaryen Eddard-Stark       Jeor-Mormont
 [9] Joffrey-Baratheon  Jon-Snow           Sansa-Stark        Luwin
[13] Robb-Stark         Tyrion-Lannister   Robert-Baratheon   Drogo
[17] Jorah-Mormont      Petyr-Baelish      Varys              Samwell-Tarly
```

```
V(graph)$label
```

```
 [1] "Arya-Stark"         "Bran-Stark"         "Bronn"
 [4] "Catelyn-Stark"      "Cersei-Lannister"   "Daenerys-Targaryen"
 [7] "Eddard-Stark"       "Jeor-Mormont"       "Joffrey-Baratheon"
[10] "Jon-Snow"           "Sansa-Stark"        "Luwin"
[13] "Robb-Stark"         "Tyrion-Lannister"   "Robert-Baratheon"
[16] "Drogo"              "Jorah-Mormont"      "Petyr-Baelish"
[19] "Varys"              "Samwell-Tarly"
```

Use the `plot.igraph()` function to visualize a graph...

```
l <- layout_in_circle(graph)
plot.igraph(graph,
            vertex.shape="square",
            vertex.size = 5,
            layout = l)
```

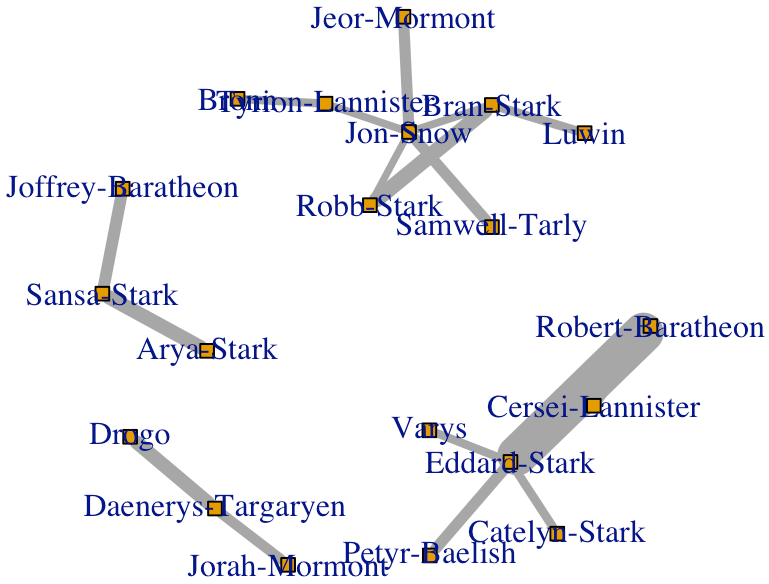



```
        # circle layouts are common, but not always visually informative!
```

{igraph} includes the option to use various “force-directed” layout algorithms for constructing nicer-looking graphs, where the edges are similar in length and cross each other as little as possible

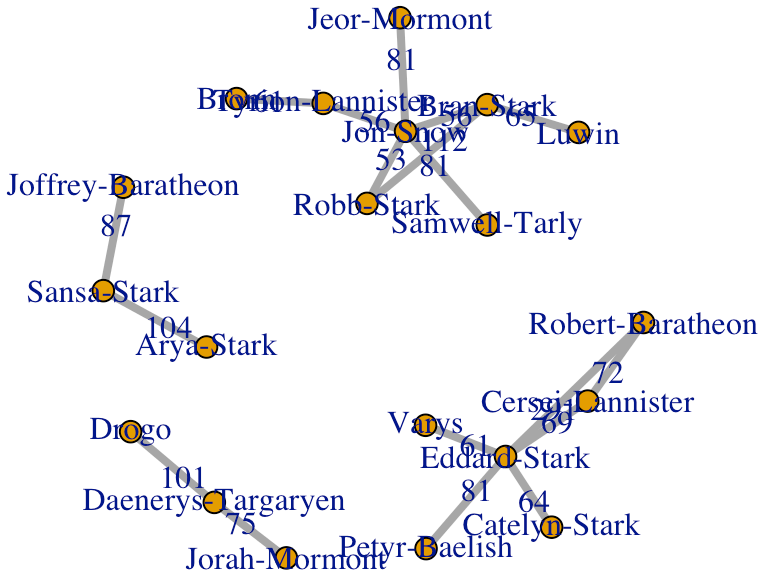
They work by simulating the graph as a physical system

```
        l <- layout_with_kk(graph)
        plot.igraph(graph,
                    vertex.shape="square",
                    vertex.size = 5,
                    edge.label = "",
                    layout = l)
```




```
# here, edge widths are determined by the "width" variable we defined above,
# and edge labels are set to be blank

plot.igraph(graph,
            vertex.shape="circle",
            vertex.size = 8,
            edge.width = 4,
            layout = l)
```




```
        # here, we set a single edge widths, and edge labels come from the "weight" varia
```

Some other useful {igraph} functions

Other functions from {igraph} can be run to show alternative representations of the graph properties or to return graph properties...

```
        # show adjacency matrix representation
        am <- as_adjacency_matrix(graph)
        head(am)
```

```
6 x 20 sparse Matrix of class "dgCMatrix"

Arya-Stark         . . . . . . . . . . 1 . . . . . . . . .
Bran-Stark         . . . . . . . . . . 1 . 1 1 . . . . . .
Bronn              . . . . . . . . . . . . . . . . 1 . . . .
Catelyn-Stark      . . . . . . 1 . . . . . . . . . . . . . .
Cersei-Lannister   . . . . . . 1 . . . . . . . 1 . . . . .
Daenerys-Targaryen . . . . . . . . . . . . . . . . 1 1 . . .
```

```
        # show adjacency matrix with weights
        am <- as_adjacency_matrix(graph, attr = "weight")
        head(am)
```

```
6 x 20 sparse Matrix of class "dgCMatrix"

Arya-Stark         . . . . . . . . . . . 104  .   .  .  .   .  . . . .
Bran-Stark         . . . . . . . . . . 56   . 65 112  .  .   .  . . . .
Bronn              . . . . . . . . . .  .   .  .   . 61  .   .  . . . .
Catelyn-Stark      . . . . . . 64 . .  .   .  .   .  .  .   .  . . . .
Cersei-Lannister   . . . . . . 69 . .  .   .  .   .  . 72   .  . . . .
Daenerys-Targaryen . . . . . . . . . .  .   .  .   .  .  . 101 75 . . .
```

```
        # show edge list
        el <- as_edgelist(graph, names = TRUE)
        head(el)
```

```
     [,1]            [,2]
[1,] "Arya-Stark"    "Sansa-Stark"
[2,] "Bran-Stark"    "Jon-Snow"
[3,] "Bran-Stark"    "Luwin"
[4,] "Bran-Stark"    "Robb-Stark"
[5,] "Bronn"         "Tyrion-Lannister"
[6,] "Catelyn-Stark" "Eddard-Stark"
```

```
        # list vertices adjacent to each vertex
        al <- as_adj_list(graph)
        head(al)
```

```
$`Arya-Stark`
+ 1/20 vertex, named, from fb6a42b:
[1] Sansa-Stark

$`Bran-Stark`
+ 3/20 vertices, named, from fb6a42b:
[1] Jon-Snow   Luwin      Robb-Stark

$Bronn
+ 1/20 vertex, named, from fb6a42b:
[1] Tyrion-Lannister

$`Catelyn-Stark`
+ 1/20 vertex, named, from fb6a42b:
[1] Eddard-Stark

$`Cersei-Lannister`
+ 2/20 vertices, named, from fb6a42b:
[1] Eddard-Stark     Robert-Baratheon
```

```
$`Daenerys-Targaryen`
+ 2/20 vertices, named, from fb6a42b:
[1] Drogo         Jorah-Mormont
```

```
# list edges connected to each vertex
ael <- as_adj_edge_list(graph)
head(ael)
```

```
$`Arya-Stark`
+ 1/18 edge from fb6a42b (vertex names):
[1] Arya-Stark--Sansa-Stark

$`Bran-Stark`
+ 3/18 edges from fb6a42b (vertex names):
[1] Bran-Stark--Jon-Snow   Bran-Stark--Luwin      Bran-Stark--Robb-Stark

$Bronn
+ 1/18 edge from fb6a42b (vertex names):
[1] Bronn--Tyrion-Lannister

$`Catelyn-Stark`
+ 1/18 edge from fb6a42b (vertex names):
[1] Catelyn-Stark--Eddard-Stark

$`Cersei-Lannister`
+ 2/18 edges from fb6a42b (vertex names):
[1] Cersei-Lannister--Eddard-Stark     Cersei-Lannister--Robert-Baratheon

$`Daenerys-Targaryen`
+ 2/18 edges from fb6a42b (vertex names):
[1] Daenerys-Targaryen--Drogo         Daenerys-Targaryen--Jorah-Mormont
```

```
# calculate the degree of each vertex
degree(graph)
```

```
       Arya-Stark         Bran-Stark              Bronn      Catelyn-Stark
                1                  3                  1                  1
  Cersei-Lannister Daenerys-Targaryen       Eddard-Stark       Jeor-Mormont
                2                  2                  5                  1
 Joffrey-Baratheon           Jon-Snow        Sansa-Stark              Luwin
                1                  5                  2                  1
       Robb-Stark   Tyrion-Lannister   Robert-Baratheon              Drogo
                2                  2                  2                  1
    Jorah-Mormont      Petyr-Baelish              Varys      Samwell-Tarly
                1                  1                  1                  1
```

```
mean(degree(graph)) # average degree
```

```
[1] 1.8
```

```
        # calculate geodesic distance between all pairs ofvertices using weights
        geo_d <- distances(graph)
        head(geo_d)
```

```
                  Arya-Stark Bran-Stark Bronn Catelyn-Stark Cersei-Lannister
Arya-Stark                 0        Inf   Inf           Inf              Inf
Bran-Stark               Inf          0   173           Inf              Inf
Bronn                    Inf        173     0           Inf              Inf
Catelyn-Stark            Inf        Inf   Inf             0              133
Cersei-Lannister         Inf        Inf   Inf           133                0
Daenerys-Targaryen       Inf        Inf   Inf           Inf              Inf
                  Daenerys-Targaryen Eddard-Stark Jeor-Mormont
Arya-Stark                       Inf          Inf          Inf
Bran-Stark                       Inf          Inf          137
Bronn                            Inf          Inf          198
Catelyn-Stark                    Inf           64          Inf
Cersei-Lannister                 Inf           69          Inf
Daenerys-Targaryen                 0          Inf          Inf
                  Joffrey-Baratheon Jon-Snow Sansa-Stark Luwin Robb-Stark
Arya-Stark                      191      Inf         104   Inf        Inf
Bran-Stark                      Inf       56         Inf    65        109
Bronn                           Inf      117         Inf   238        170
Catelyn-Stark                   Inf      Inf         Inf   Inf        Inf
Cersei-Lannister                Inf      Inf         Inf   Inf        Inf
Daenerys-Targaryen              Inf      Inf         Inf   Inf        Inf
                  Tyrion-Lannister Robert-Baratheon Drogo Jorah-Mormont
Arya-Stark                     Inf              Inf   Inf          Inf
Bran-Stark                     112              Inf   Inf          Inf
Bronn                           61              Inf   Inf          Inf
Catelyn-Stark                  Inf              205   Inf          Inf
Cersei-Lannister               Inf               72   Inf          Inf
Daenerys-Targaryen             Inf              Inf   101           75
                  Petyr-Baelish Varys Samwell-Tarly
Arya-Stark                  Inf   Inf          Inf
Bran-Stark                  Inf   Inf          137
Bronn                       Inf   Inf          198
Catelyn-Stark               145   125          Inf
Cersei-Lannister            150   130          Inf
Daenerys-Targaryen          Inf   Inf          Inf
```

```
        # calculate unweighted geodesic distance between pairs of
        # vertices, ignoring weights
        geo_d <- distances(graph, algorithm = "unweighted")
```

```
Warning in distances(graph, algorithm = "unweighted"): Unweighted algorithm
chosen, weights ignored
```

```
        head(geo_d)
```

```
                  Arya-Stark Bran-Stark Bronn Catelyn-Stark Cersei-Lannister
Arya-Stark                 0        Inf   Inf           Inf              Inf
Bran-Stark               Inf          0     3           Inf              Inf
Bronn                    Inf          3     0           Inf              Inf
Catelyn-Stark            Inf        Inf   Inf             0                2
Cersei-Lannister         Inf        Inf   Inf             2                0
Daenerys-Targaryen       Inf        Inf   Inf           Inf              Inf
                  Daenerys-Targaryen Eddard-Stark Jeor-Mormont
Arya-Stark                       Inf          Inf          Inf
Bran-Stark                       Inf          Inf            2
Bronn                            Inf          Inf            3
Catelyn-Stark                    Inf            1          Inf
Cersei-Lannister                 Inf            1          Inf
Daenerys-Targaryen                 0          Inf          Inf
                  Joffrey-Baratheon Jon-Snow Sansa-Stark Luwin Robb-Stark
Arya-Stark                        2      Inf           1   Inf        Inf
Bran-Stark                      Inf        1         Inf     1          1
Bronn                           Inf        2         Inf     4          3
Catelyn-Stark                   Inf      Inf         Inf   Inf        Inf
Cersei-Lannister                Inf      Inf         Inf   Inf        Inf
Daenerys-Targaryen              Inf      Inf         Inf   Inf        Inf
                  Tyrion-Lannister Robert-Baratheon Drogo Jorah-Mormont
Arya-Stark                     Inf              Inf   Inf           Inf
Bran-Stark                       2              Inf   Inf           Inf
Bronn                            1              Inf   Inf           Inf
Catelyn-Stark                  Inf                2   Inf           Inf
Cersei-Lannister               Inf                1   Inf           Inf
Daenerys-Targaryen             Inf              Inf     1             1
                  Petyr-Baelish Varys Samwell-Tarly
Arya-Stark                  Inf   Inf           Inf
Bran-Stark                  Inf   Inf             2
Bronn                       Inf   Inf             3
Catelyn-Stark                 2     2           Inf
Cersei-Lannister              2     2           Inf
Daenerys-Targaryen          Inf   Inf           Inf
```

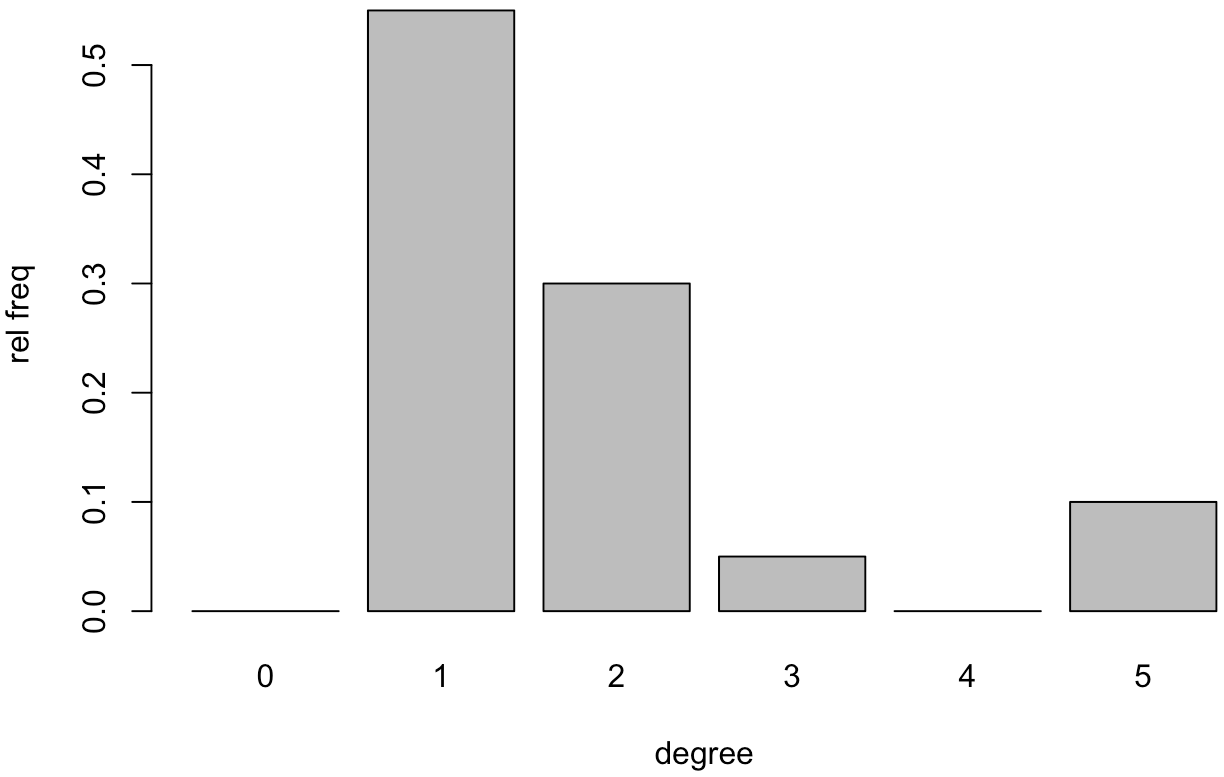
```r
# average distance between pairs of vertices
mean_distance( # uses weights
  graph,
  unconnected = TRUE,
  details = FALSE
)
```

```
[1] 130.7551
```

Calculate and plot the degree distribution...

```r
dd <- degree_distribution(graph)

# the following two lines plot the degree distribution
x <- seq(0, length(dd)-1, by = 1)
barplot(dd, names = x, xlab = "degree", ylab = "rel freq")
```



Write out an {igraph} data to a file...

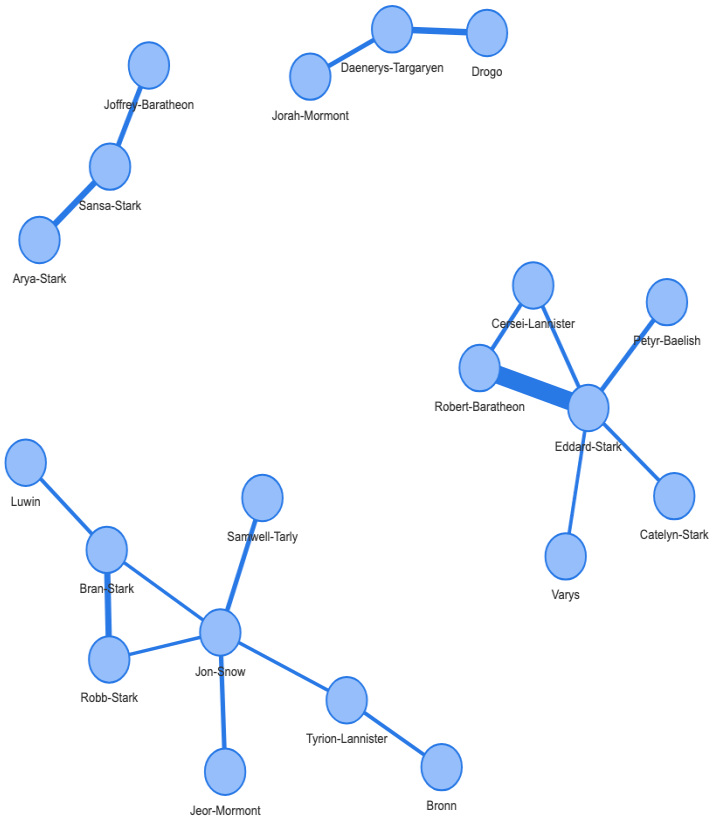
```r
# in .csv format
write_csv(as_long_data_frame(graph), file = "~/Desktop/got.csv")

# in .graphml format
write_graph(graph, file = "~/Desktop/got.graphml", format = c("graphml"))
```

Other cool things to explore

Plot an {igraph} object with {visNetwork} directly

```
visIgraph(graph)
```




```
# here, graph is an {igraph} object, and
# visIgraph() is a function from {visNetwork}
```

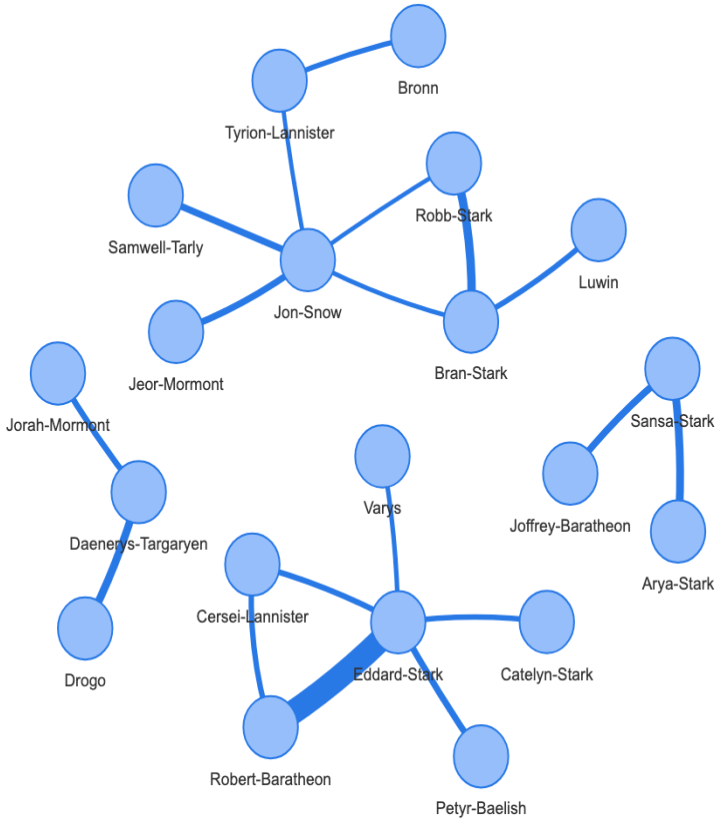
Convert an {igraph} object to a {visNetwork} object

```
graph <- toVisNetworkData(graph, idToLabel = TRUE)
```

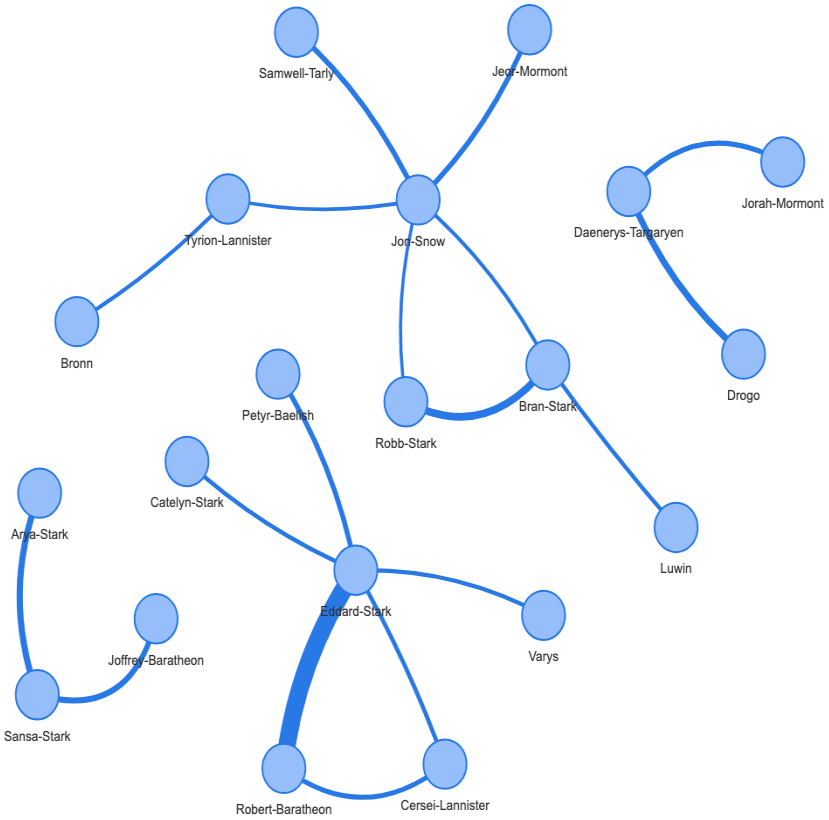
{visNetwork} options

Below are some examples of interesting {visNetwork} options for tweaking graph layouts (e.g., by editing the *physics* of layout positioning) and for interacting with a graph...

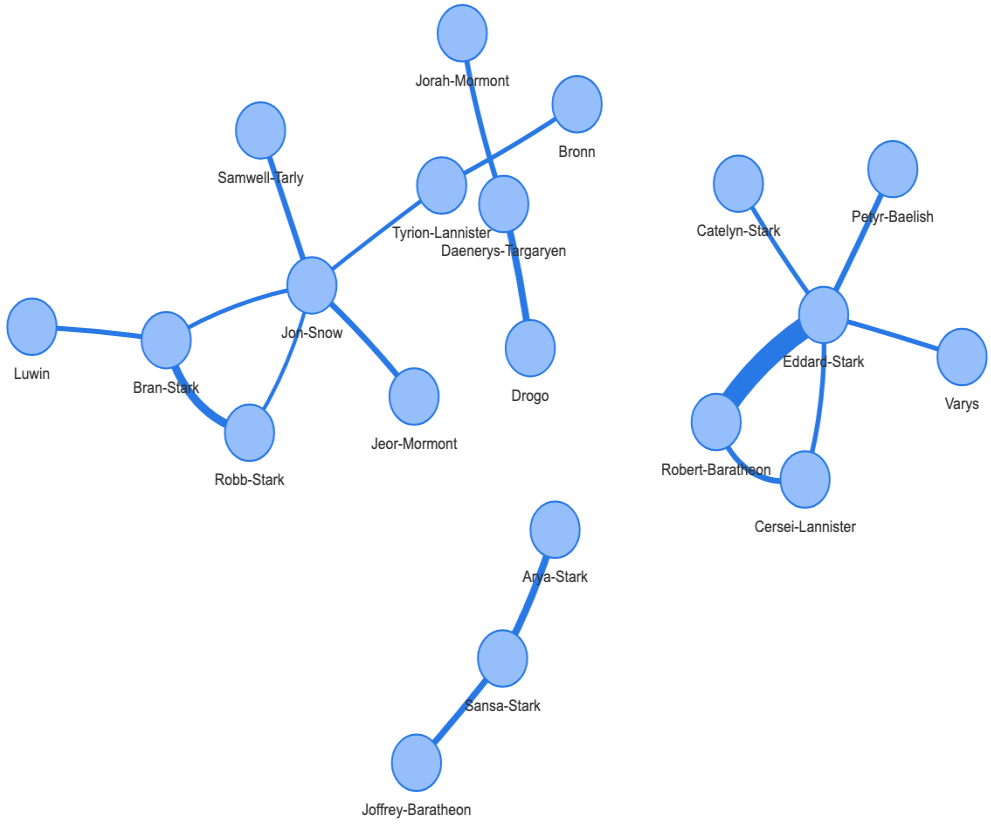
```
visNetwork(graph$nodes, graph$edges) %>%
  visPhysics(
    solver = "forceAtlas2Based",
    forceAtlas2Based = list(gravitationalConstant = -30)
    )
```




```
visNetwork(graph$nodes, graph$edges) %>%
  visPhysics(solver = "repulsion")
```

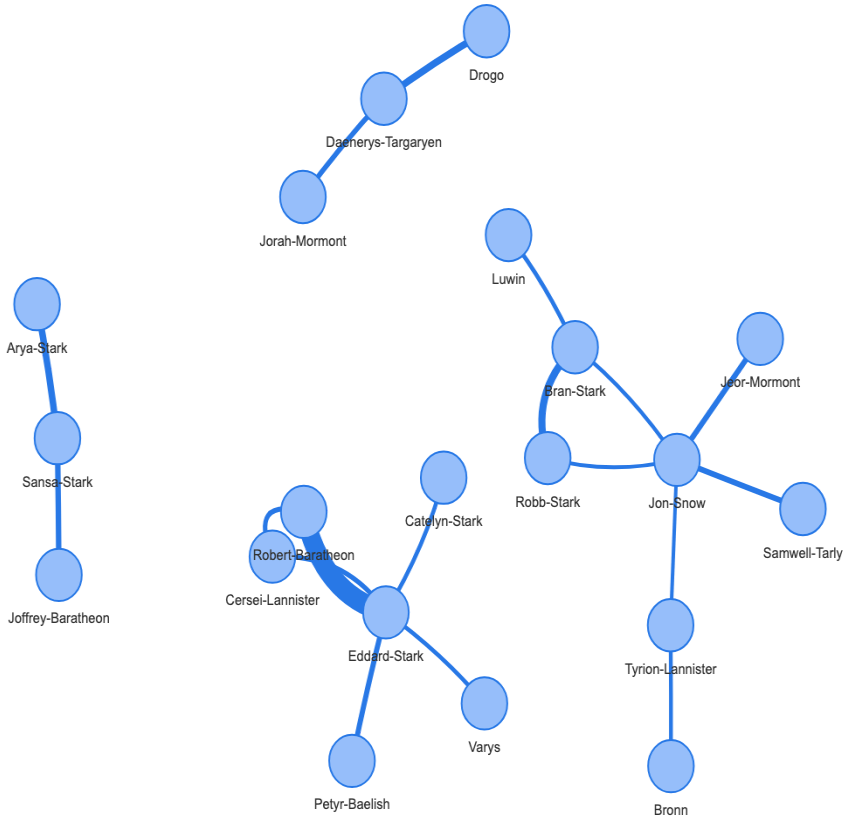



```
visNetwork(graph$nodes, graph$edges) %>%
  visOptions(
    highlightNearest = list(
      enabled = TRUE,
      hover = TRUE,
      degree = 1,
      hideColor = "rgba(0,0,0,0.05)")
    ) %>%
  visEdges(color = list(inherit = "to"))
```



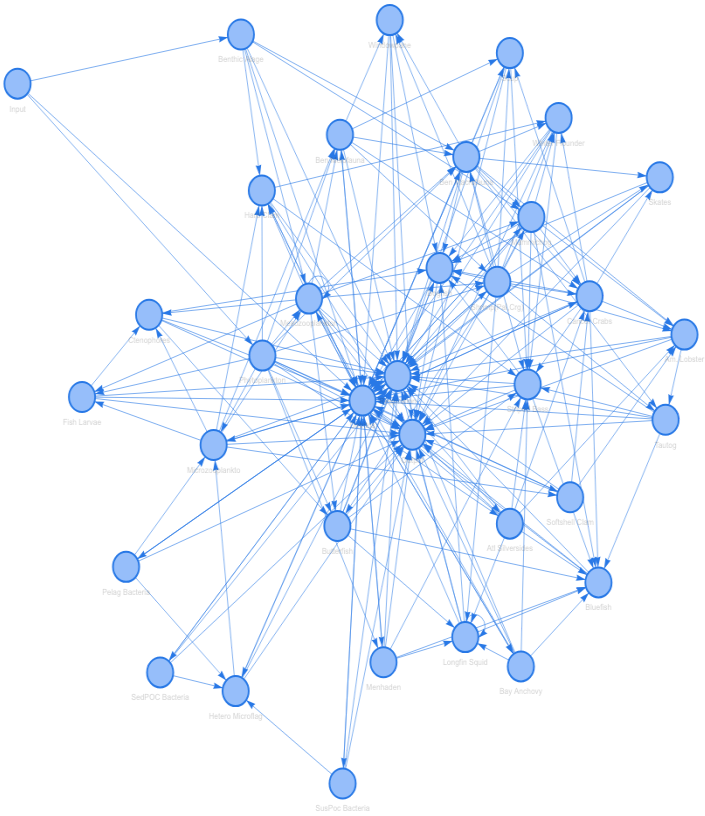

```
visNetwork(graph$nodes, graph$edges) %>%
  visOptions(manipulation = TRUE) %>%
  visLayout(randomSeed = 123)
```

Edit

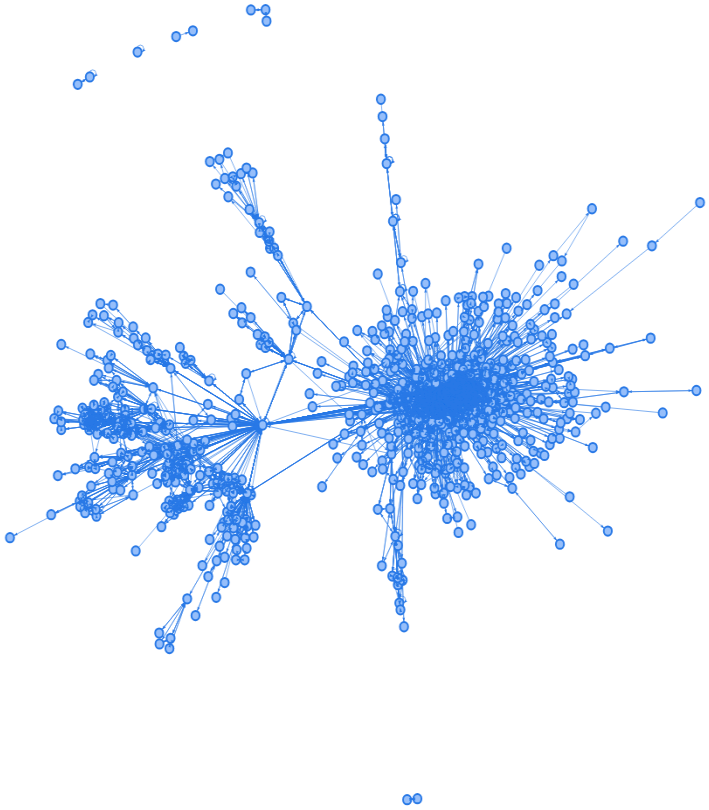


Visualizing and working with other datasets from the package {igraphdata}

```
library(igraphdata)
data("foodwebs") # list containing multiple {igraph} objects
graph <- foodwebs$Narragan
visIgraph(graph)
```



```
data("USairports") # a single {igraph} object
graph <- USairports
visIgraph(USairports)
```



Questions

- Calculate the mean degree and plot the degree distribution for each of the new networks graphed above