DATA SOCIETY:

Interactive visualization with R - Network Graphs - 2

One should look for what is and not what he thinks should be. (Albert Einstein)

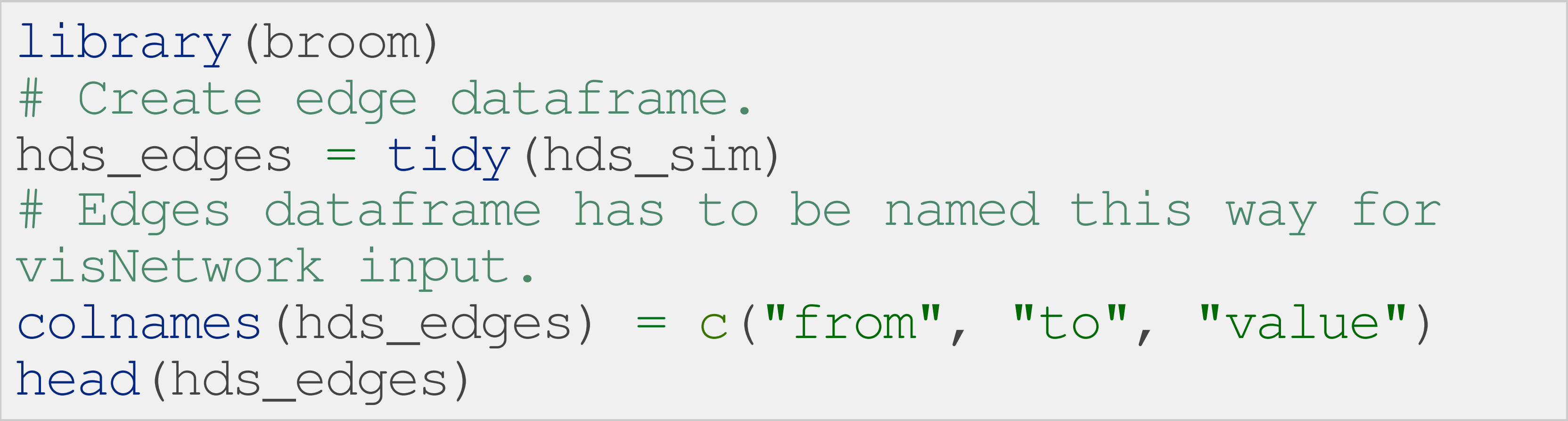
Module completion checklist

Objective	Complete
Create nodes and edges dataframes	
Build and customize a network HTMLwidget	

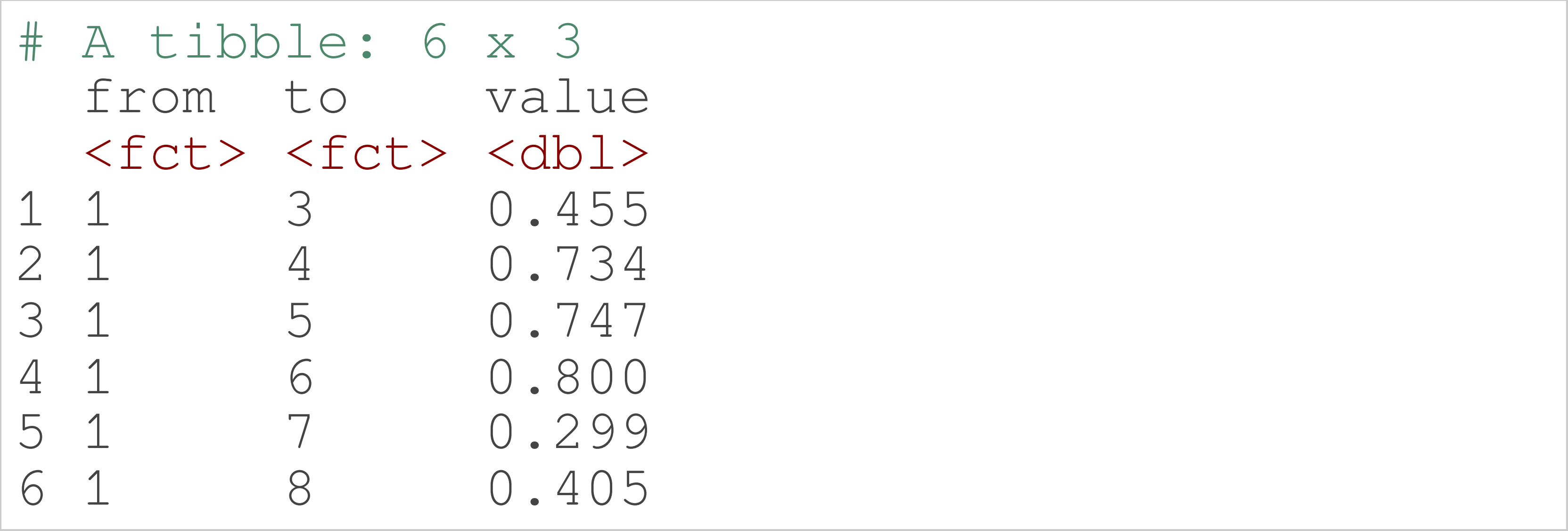
Creating the network: edges

- Now that the dataset is prepped for visualization, we must generate the **edge dataframe** and the **node dataframe** that will comprise our network graph
- We can start by transforming our similarity matrix into an edge dataframe
- The edge dataframe informs `visNetwork`:
  - to draw an edge `from` which node `to` which node
  - the `value` (the thickness) of the edge

- We can do this using the `tidy` function from the broom package, which turns the messy output of built-in R functions into tidy dataframes



```
library(broom)
# Create edge dataframe.
hds_edges = tidy(hds_sim)
# Edges dataframe has to be named this way for
visNetwork input.
colnames(hds_edges) = c("from", "to", "value")
head(hds_edges)
```

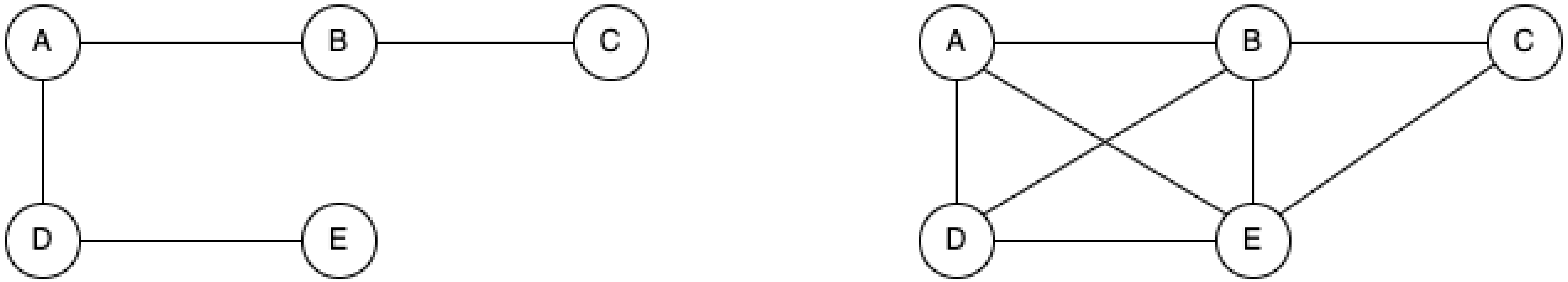


```
# A tibble: 6 x 3
  from  to    value
  <fct> <fct> <dbl>
1 1     3     0.455
2 1     4     0.734
3 1     5     0.747
4 1     6     0.800
5 1     7     0.299
6 1     8     0.405
```

Setting a similarity threshold

- We need to choose a similarity threshold to create edges for the network
- A similarity threshold is a measure of the strength of the relationship between nodes in the network
- The wrong similarity threshold will result in a very sparse network graph
    - A sparse network graph is a type of network graph in which only a fraction of all possible connections between nodes (vertices) exists.
- The threshold value can be chosen via trial and error based on what works best for your network
- We generally assign the threshold as **0.5** or as the **mean/median** of the similarity matrix

Sparse Graph vs Dense Graph



Setting the number of edges

- To simplify our network viz, we are going to subset the **first 200 edges**, ranked according to their values

```
# We choose the median as the threshold since this gives us the best visualization.
hds_edges = subset(hds_edges,value>median(hds_edges$value))

# Arrange by order of edge thickness.
hds_edges = arrange(hds_edges, desc(value))

# Subset only top 200.
hds_edges = hds_edges[1:200,]
```

Creating the network: nodes

- Now that we have the edges set, we can focus on the nodes
- The nodes input to `visNetwork` must have an `id` column
- We can get these nodes by extracting the unique nodes from the **`from`** and **`to`** columns of the edges dataframe

```
# Get unique nodes from edges dataframe and combine them
hds_nodes_from = data.frame(id = unique(hds_edges$from))
hds_nodes_to = data.frame(id = unique(hds_edges$to))
hds_nodes = rbind(hds_nodes_from,hds_nodes_to)

# Retain unique nodes in case nodes are repeated in `from` and `to` columns
hds_nodes = unique(hds_nodes)
```

Creating the network: nodes

- It can also have additional attributes like color, shape, labels, etc.
  - **Color** : Used to visually distinguish nodes based on categories, attributes, or characteristics.
  - **Shape** : Determines the geometric form of nodes, emphasizing groupings or types within the network.
  - **Label** : Determines the geometric form of nodes, emphasizing groupings or types within the network.

```r
# Add color to the nodes dataframe based on stroke value from original dataframe
hds_small = select(hds_small, stroke)               #<- we only need the target info
hds_small$id = rownames(hds_small)

# Merge nodes dataframe with the dataframe with Target value
hds_nodes = merge(hds_nodes, hds_small,
                    by = "id", all.x = TRUE)            #<- merge() needs the `id` column to
                                                        #   join the two dataframes
# Assign color to nodes based on the stroke value
hds_nodes$color = factor(hds_nodes$stroke,          #<- create a factor
labels = c("orange", "darkblue"), #<- assign color
levels = c(1, 0))                                 #   based on Target value

# Because we aligned colors based on the stroke column, we can drop it
hds_nodes = select(hds_nodes, c(id, color))
```

Creating the network: nodes (cont’d)

```
head(hds_nodes)
```

```
    id    color
1 1002 darkblue
2 1019 darkblue
3 1035 darkblue
4 1036 darkblue
5 1038 darkblue
6 1044 darkblue
```

Module completion checklist

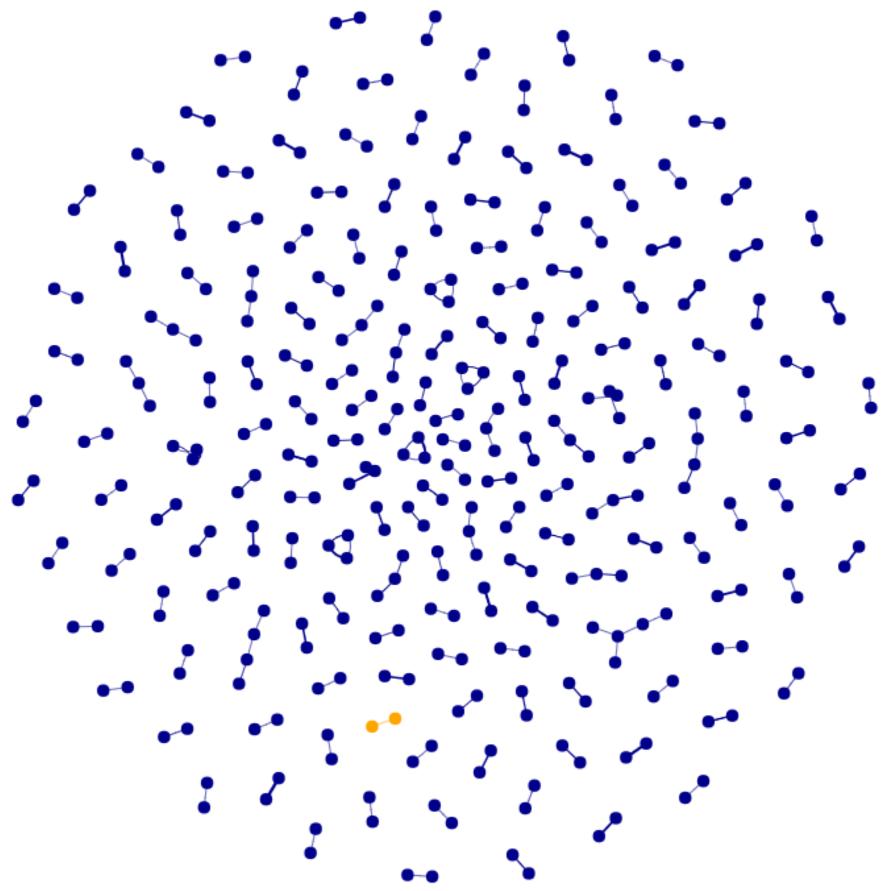
Objective	Complete
Create nodes and edges dataframes	✔
Build and customize a network HTMLwidget	

Creating the network

- The `nodes` and `edges` dataframes are all we need to create the visNetwork
- Let’s take a look at the resulting network
- You can zoom in to see the individual nodes

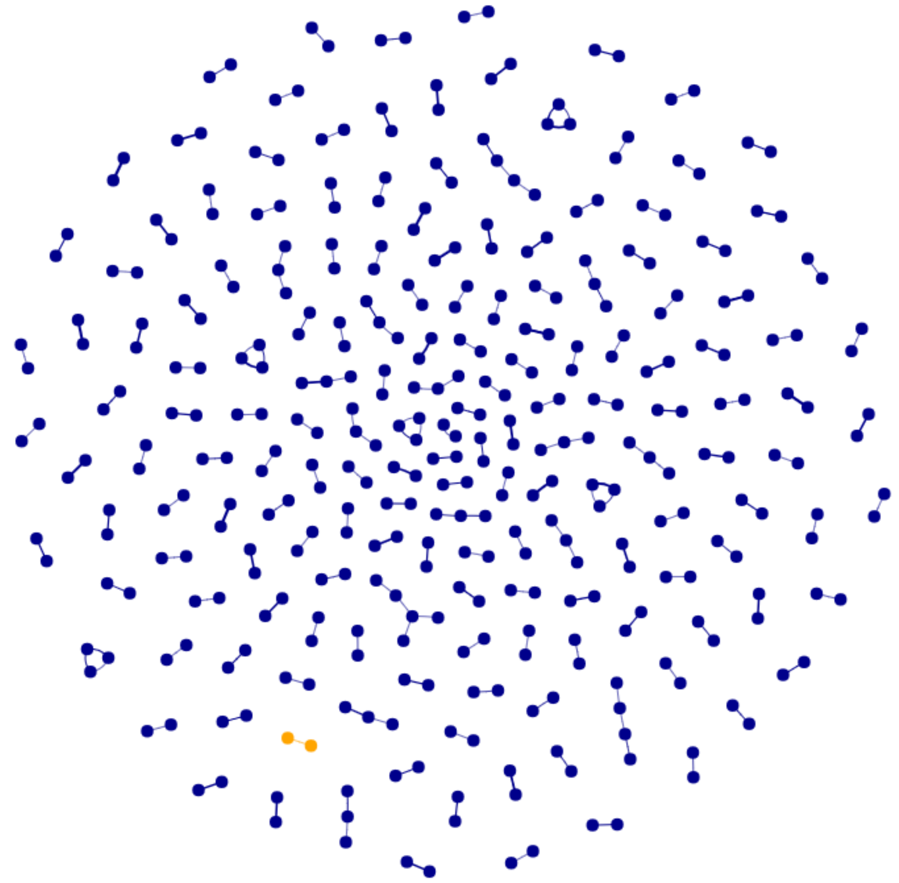
```
# Create network.
hds_network = visNetwork(hds_nodes,
#<- set nodes
                           hds_edges)           #
<- set edges
```

```
hds_network
```



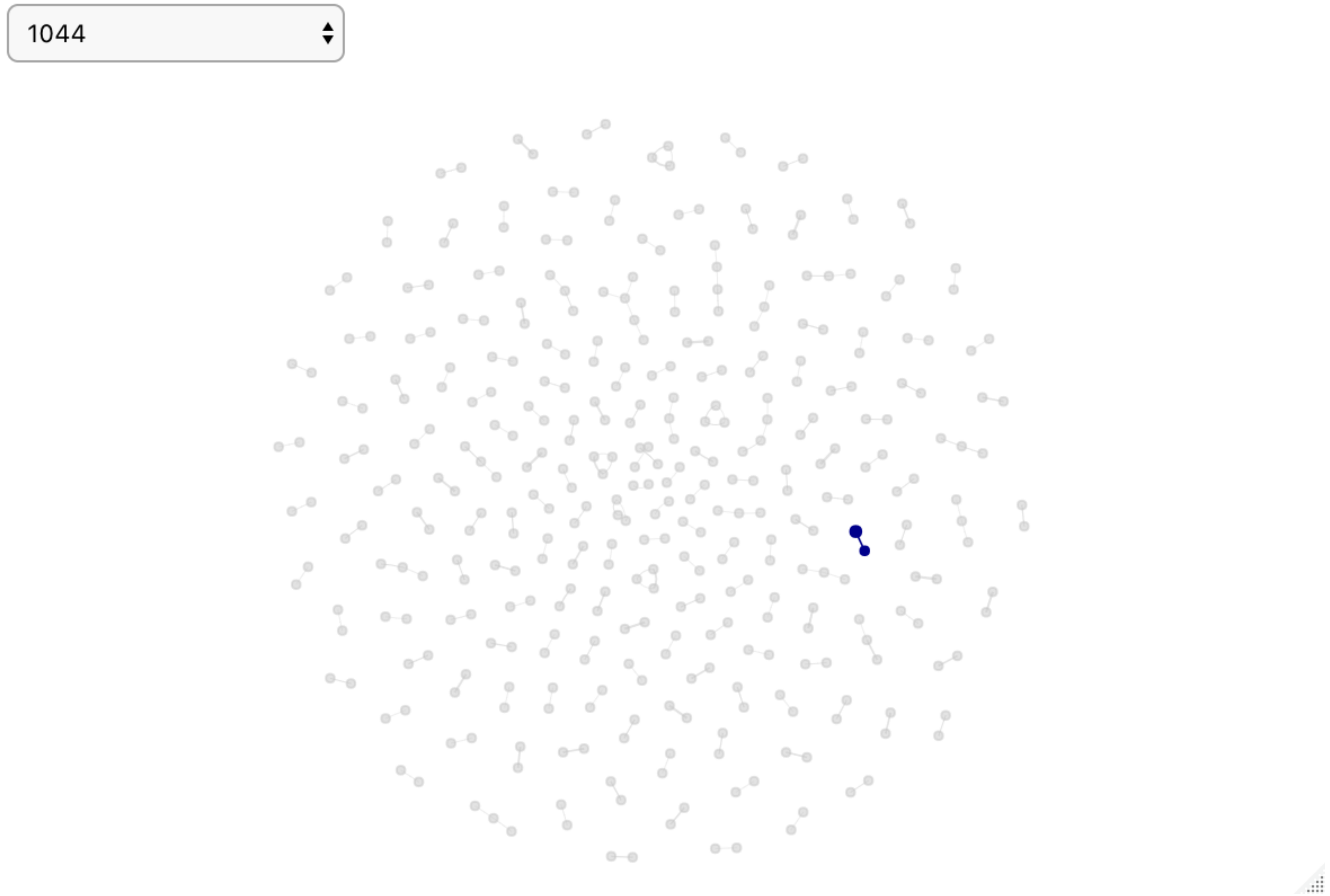
Analyzing the network

- Share your thoughts and findings in the chat:
    - Which target has **the most** nodes?
    - Are the nodes for vulnerable people **connected** to the nodes for non-vulnerable people?
    - What could this tell us about the **similarity** between the attributes of people in these two categories?



Analyzing the network

- We notice that there are:
  - more nodes with **stroke** 0 (non-vulnerable people)
  - fewer nodes with **stroke** = 1 (vulnerable people)
- The nodes for vulnerable people do not appear very connected to those for non-vulnerable people
- This means that the attributes of people from different categories are not very similar



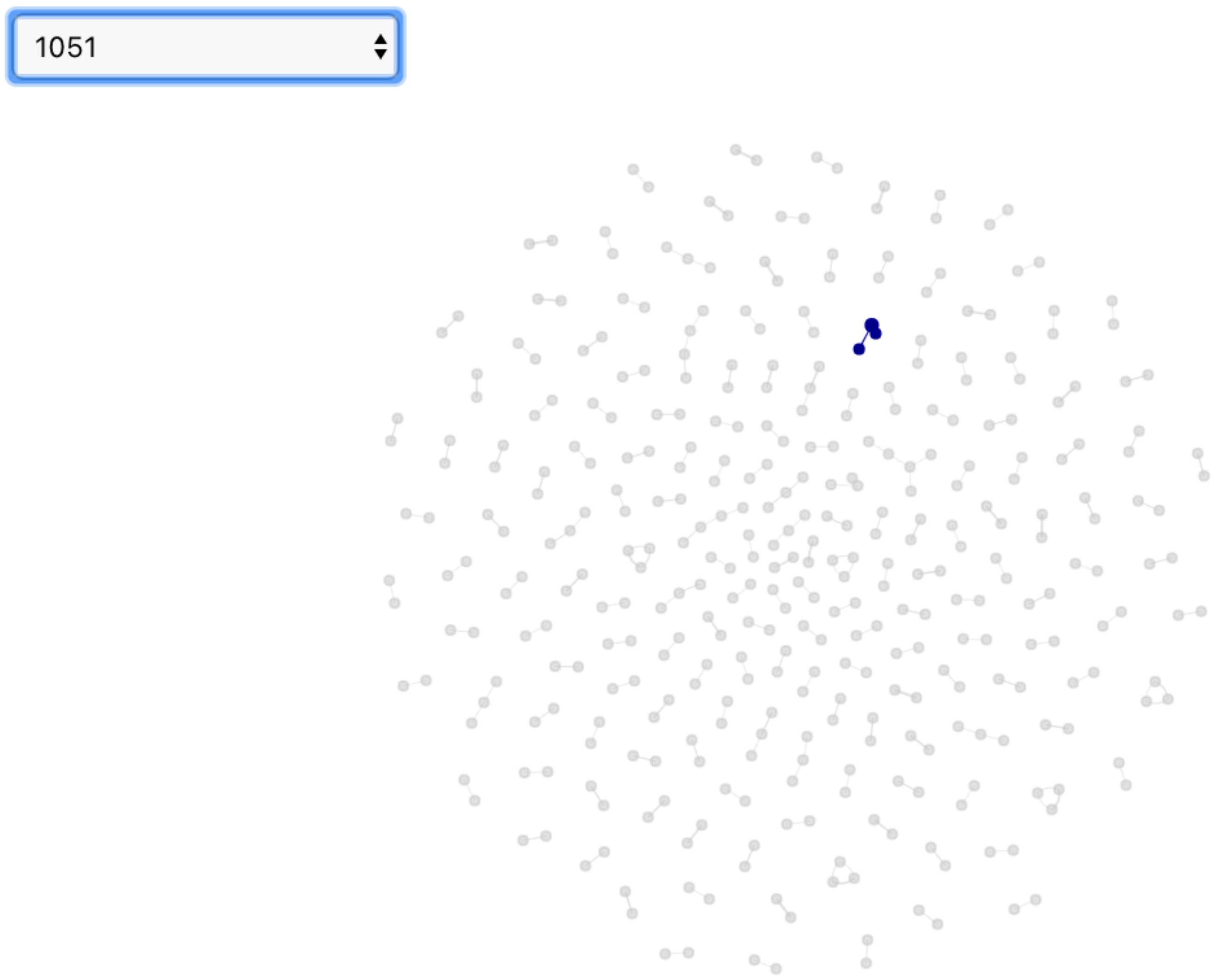
Customizing the network visualization

- Since we built the network graph as a widget, we can make it more interactive to incorporate additional information
- For example, we can add extra functionality using `visOptions`, such as:
  - Highlighting the nearest nodes when a single node is selected
  - Creating a dropdown menu to select specific nodes

```
# Add network visualizations
hds_network = visNetwork(hds_nodes,                 #<- set nodes
                           hds_edges)  %>%            #<- set edges
                 visOptions(highlightNearest = TRUE,    #<- highlight nearest nodes
                                                        #   when clicking on a node
                           nodesIdSelection = TRUE)     #<- create a dropdown menu to
                                                        #   select particular nodes
```

Customizing the network visualization (cont’d)

```
hds_network
```



1051

- *Click here* for the entire list of `visOptions()`

Saving networks with htmlwidgets

- It is possible to save the networks as HTML files, allowing us to share them, use them later, or embed them in presentations

```
# Set working directory to where you save interactive plots.
setwd(plot_dir)

# Load the library.
library(htmlwidgets)

# Save desired interactive plot to an HTML file.
saveWidget(hds_network,                       #<- plot object to save
           "network.html",                      #<- name of file to where the plot is to be saved
            selfcontained = TRUE)               #<- set `selfcontained` to TRUE, so that
                                                #    all necessary files and scripts are embedded
                                                #    into the HTML file itself
```

Knowledge check

Exercise

You are now ready to try tasks 3-7 in the Exercise for this topic

Module completion checklist

Objective	Complete
Create nodes and edges dataframes	✔
Build and customize a network HTMLwidget	✔

Network graphs: topic summary

In this part of the course, we have covered:

- Transforming and preparing data for a network graph visualization
- Building and customizing interactive network graphs

Congratulations on completing this module!