

DATA SOCIETY®

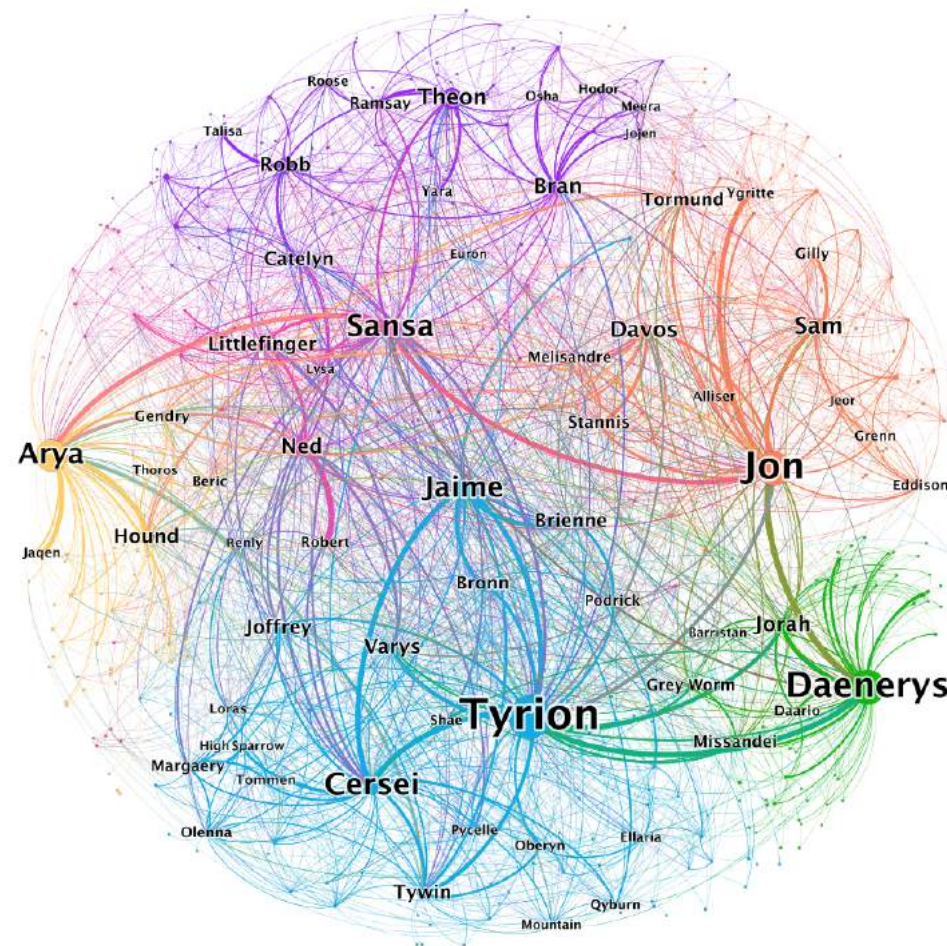
Interactive visualization in R - Part 3

*"One should look for what is and not what he thinks should be."
-Albert Einstein.*

Welcome back!

- In the last module, we learned how to create layered charts and maps using `highcharter`
- There are several other R packages that create R bindings to JavaScript libraries, which may be more appropriate for specific types of visualizations
- Today, we will learn how to use one of these to **visualize a network**

To start you can read a bit about a fun relationship network to visualize: Game of Thrones!



<https://networkofthrones.wordpress.com/>

Module completion checklist

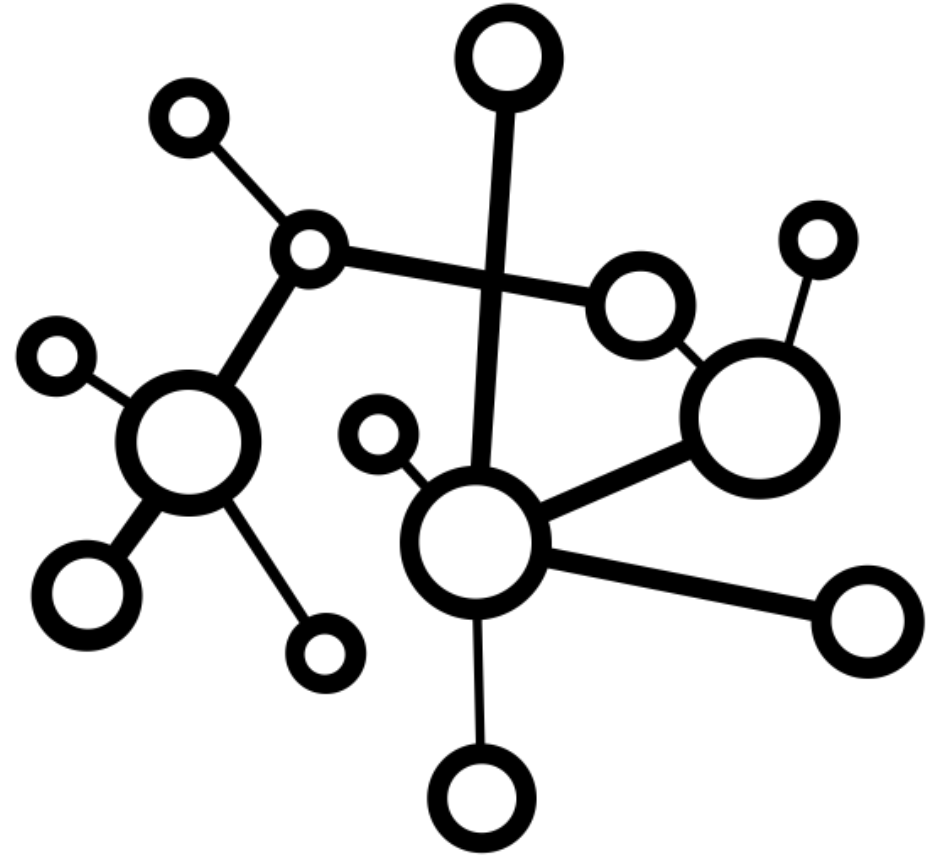
Objective	Complete
Summarize the concepts of distance matrix and network visualization	
Create a distance matrix for a given dataset	
Create nodes and edges dataframes	
Build and customize a Network HTMLwidget	

HTML Widgets with JS

- The `htmlwidgets` package provides a framework for easily creating R bindings to JavaScript libraries
- HTML widgets are useful because they can be:
 - used at the R console for data analysis just like conventional R plots
 - seamlessly embedded within R Markdown documents
 - saved as standalone web pages for ad-hoc sharing via email, Dropbox, etc.
- Some popular packages based on `htmlwidgets` are `leaflet` for maps, `dygraphs` for time series, and `rthreejs` for interactive 3D graphics
- In this module we will use `visNetwork` to create an HTML widget for network visualization

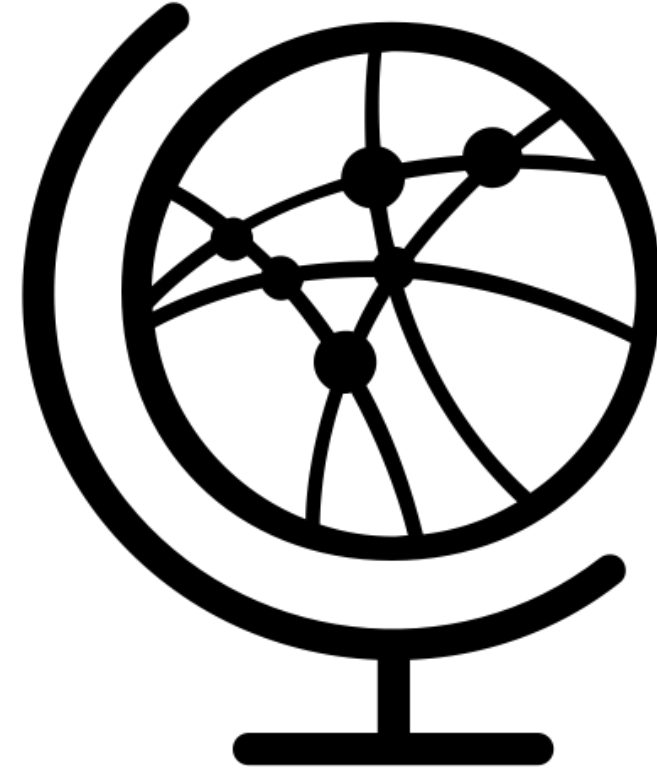
Network and similarity

- Network visualization is a great way of understanding the **relationships** between **individual observations or groups of observations** in your data
- Networks are a collection of connected objects:
 - **Nodes**: the objects, usually represented as points
 - **Edges**: the relationship between a pair of nodes, usually represented by a line connecting the nodes



Network and similarity in the social sciences

- In the social sciences, we often want to study patterns of **relationships** that connect **members of a social system** at all scales
 - **Nodes**: a social entity (e.g., countries)
 - **Edges**: a social relation of interest (e.g., the trade flow between the countries)

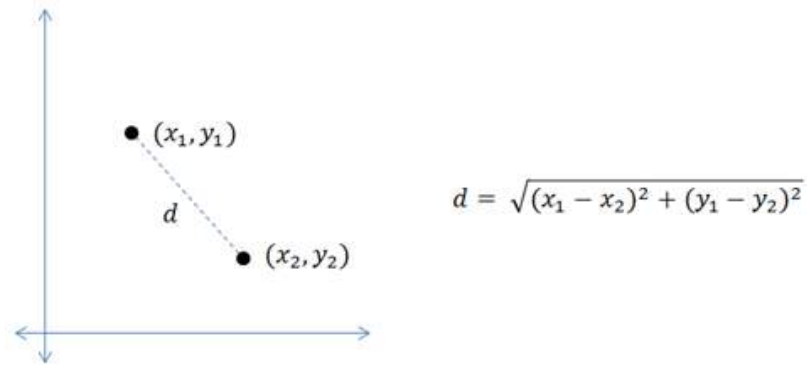


Distance and similarity

- In network theory, we use the distance between two nodes to describe how connected they are
- The **smaller** the distance the **more similar** the two nodes are
- There are different distance metrics: Euclidean, Manhattan, Binary, and Minowski
- We will use the Euclidean distance metric in this module

Distance and similarity

- The Euclidean distance between two points is the length of the line segment connecting them
- The distance formula can be extended for Euclidean spaces with more than two dimensions



Distance Matrix

- A distance matrix for N nodes is of size $N \times N$ where each value corresponds to the distance between a pair of nodes
- Properties of the distance matrix:
 - Values on the main diagonal are zero
 - The matrix is symmetric
- Thus we can capture all the required information only using the lower triangle of the matrix

	A	B	C	D	E	F
A	0	16	47	72	77	79
B	16	0	37	57	65	66
C	47	37	0	40	30	35
D	72	57	40	0	31	23
E	77	65	30	31	0	10
F	79	66	35	23	10	0

B	16				
C	47	37			
D	72	57	40		
E	77	65	30	31	
F	79	66	35	23	10
	A	B	C	D	E

visNetwork

- visNetwork is an R package for network visualization, using the [vis.js JavaScript library](#) and based on `htmlwidgets`
- visNetwork needs at least two arguments to plot a basic network:
 - a **nodes** dataframe with an `id` column
 - an **edges** dataframe with `from` and `to` columns
- It has additional functions to customize the network and add interactivity

```
library(visNetwork)
```

```
?visNetwork
```

R: Network visualization ▾ Find in Topic

visNetwork {visNetwork}

R Documentation

Network visualization


Description

Network visualization using vis.js library. For full documentation, have a look at [visDocumentation](#).

Usage

```
visNetwork(nodes = NULL, edges = NULL, dot = NULL, gephi = NULL,  
           width = NULL, height = NULL, main = NULL, submain = NULL,  
           footer = NULL, background = "rgba(0, 0, 0, 0)", ...)
```

Module completion checklist

Objective	Complete
Summarize the concepts of distance matrix and network visualization	
Create a distance matrix for a given dataset	
Create nodes and edges dataframes	
Build and customize a Network HTMLwidget	

Directory settings

- In order to maximize the efficiency of your workflow, you should encode your directory structure into *variables*
- Let the `main_dir` be the variable corresponding to your `skillsoft` folder on your Desktop

```
# Set `main_dir` to the location of your `skillsoft` folder (for Mac/Linux).
main_dir = "~/Desktop/skillsoft"

# Set `main_dir` to the location of your `skillsoft` folder (for Windows).
main_dir = "C:/Users/[username]/Desktop/skillsoft"

# Make `data_dir` from the `main_dir` and
# remainder of the path to data directory.
data_dir = paste0(main_dir, "/data")

# Do the same for your 'plot_dir' which is where your interactive plots will be stored.
plot_dir = paste0(main_dir, "/plots")
```

Loading packages

- These are the packages we will need for today

```
library(htmlwidgets)
library(tidyverse)
library(broom)
library(dplyr)
library(visNetwork)
```

Our datasets

- We will use two datasets today:
 - The slides will use **Costa Rica household poverty data**
 - The exercises will use **Chicago census data**
- The data dictionaries can be found through the links above.



Case study: poverty in Costa Rica

- We will be diving into a case study from the **Inter-American Development Bank (IDB)**, which is related to a data science competition that was hosted on [Kaggle.com](https://www.kaggle.com)
- It starts with a model that agencies use to classify a household's poverty level and predict their level of need, based on observable attributes like the material of their walls and ceiling or the assets found in their home



Dataset description

- The given dataset contains variables about:
 - **Households**: features about the house they live in region, etc.
 - **Individuals in the household** : gender, age, education, etc.
- The Target variable has four categories:
 - extreme poverty
 - moderate poverty
 - vulnerable households
 - non-vulnerable households



Loading the dataset

- In this dataset, we are interested in seeing how similar two households are to each other

```
# Read in the Costa Rican dataset.
setwd(data_dir)

costa = read.csv("costa_rica_poverty.csv", header = TRUE)

# View the dimensions of the dataset.
dim(costa)
```

```
[1] 9557    84
```

```
# View first few rows and columns
costa[1:5, 1:10]
```

```
  household_id      ind_id rooms tablet males_under_12 males_over_12 males_tot
1  21eb7f̄cc1 ID_279628̄684     3      0                0                1         1
2  0e5d7a658 ID_f29eb3ddd     4      1                0                1         1
3  2c7317ea8 ID_68de51c94     8      0                0                0         0
4  2b58d945f ID_d671db89c     5      1                0                2         2
5  2b58d945f ID_d56d6f5f5     5      1                0                2         2
  females_under_12 females_over_12 females_tot
1                0                0          0
2                0                0          0
3                0                1          1
4                1                1          2
5                1                1          2
```

Subsetting the dataset

- Since network visualization is easier with smaller sets of data, we will subset households from a particular region and use only a few columns

```
# Subset one region.
costa_subset_target = subset(costa, region_central == 1)

# Subset a few columns.
costa_small = costa_subset_target %>%
  select(rooms, ppl_total,
         monthly_rent, Target)

# View the first few rows of the dataset.
head(costa_small)
```

	rooms	ppl_total	monthly_rent	Target
1	3	1	190000	4
2	4	1	135000	4
3	8	1	NA	4
4	5	4	180000	4
5	5	4	180000	4
6	5	4	180000	4

Cleaning the dataset

```
# Remove rows with NA.  
costa_small = na.omit(costa_small)  
  
# We keep only the unique rows since duplicate rows would have a distance of 0.  
costa_small= unique(costa_small)  
  
head(costa_small)
```

	rooms	ppl_total	monthly_rent	Target
1	3	1	190000	4
2	4	1	135000	4
4	5	4	180000	4
8	2	4	130000	4
12	3	2	100000	4
16	2	4	90000	4

Measuring similarity of households

- To measure the similarity between households, we will use the `dist` function
- `dist` takes **a numeric matrix or a dataframe** as input
- It returns **the distance matrix** stored by columns in a vector
- It only returns the lower triangle of the distance matrix since the matrix is symmetric and the diagonal elements are 0

?dist

R: Distance Matrix Computation ▾

Find in Topic

dist {stats}

R Documentation

Distance Matrix Computation

Description

This function computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix.

Usage

```
dist(x, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
```

Measuring similarity of terms

- We normalize the distances to values between 0 and 1 so that we can interpret the distances easily

```
# Create distance matrix.  
costa_distance = dist(costa_small)  
  
# `dist` returns the lower triangle of the distance matrix as a vector.  
head(costa_distance)
```

```
[1] 55000 10000 60000 90000 100000 25000
```

```
# Normalize the distances to values between 0 and 1.  
costa_distance = costa_distance/max(costa_distance)  
  
head(costa_distance)
```

```
[1] 0.023369678 0.004249033 0.025494194 0.038241292 0.042490324 0.010622581
```

Measuring similarity of terms

- The **greater the distance** between two observations, the **more different** they are
- Thus, the **1 - normalized distance** indicates how **similar** two observations are

```
# Use 1- distance to obtain the value for similarity.  
costa_sim = 1-costa_distance  
head(costa_sim)
```

```
[1] 0.9766303 0.9957510 0.9745058 0.9617587 0.9575097 0.9893774
```

Knowledge check 1



Exercise 1



Module completion checklist

Objective	Complete
Summarize the concepts of distance matrix and network visualization	✓
Create a distance matrix for a given dataset	✓
Create nodes and edges dataframes	
Build and customize a Network HTMLwidget	

Creating the network: edges

- The edge dataframe tells `visNetwork`:
 - from which node to which node to draw an edge
 - the `value` (the thickness) of the edge
- We need to transform our similarity matrix into an **edge** dataframe
- 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

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

```
# A tibble: 6 x 3
  from to value
  <fct> <fct> <dbl>
1 1     2  0.977
2 1     4  0.996
3 1     8  0.975
4 1    12  0.962
5 1    16  0.958
6 1    20  0.989
```

Creating the network: edges

- We need to choose a similarity threshold to create **edges** for the network or else it will be very sparse
- 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

Creating the network: edges

- We subset only the first 200 edges ranked by their values, in order to simplify our network viz

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

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

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

Creating the network: 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
costa_nodes_from = data.frame(id = unique(costa_edges$from))
costa_nodes_to = data.frame(id = unique(costa_edges$to))
costa_nodes = rbind(costa_nodes_from, costa_nodes_to)

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

Creating the network: nodes

- It can also have additional attributes like color, shape, labels, etc.

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

# Merge nodes dataframe with the dataframe with Target value
costa_nodes = merge(costa_nodes, costa_small,
                    by = "id", all.x = TRUE) #<- merge() needs the `id` column to
# join the two dataframes

# Assign color to nodes based on the Target value
costa_nodes$color = factor(costa_nodes$Target, #<- create a factor
labels = c("orange", "darkblue", "maroon", "seagreen"), #<- assign color
levels = c(1, 2, 3, 4)) # based on Target value

# We do not need the Target column anymore.
costa_nodes = select(costa_nodes, c(id, color))
head(costa_nodes)
```

```
   id   color
1 1004 seagreen
2 1049 seagreen
3 1061  maroon
4 1083 seagreen
5 1095 darkblue
6 1101 seagreen
```

Module completion checklist

Objective	Complete
Summarize the concepts of distance matrix and network visualization	✓
Create a distance matrix for a given dataset	✓
Create nodes and edges dataframes	✓
Build and customize a Network HTMLwidget	

Creating the network

- All we need to create the visNetwork are the nodes and edges dataframes

```
# Create network.  
costa_network = visNetwork(costa_nodes,          #<- set nodes  
                           costa_edges)         #<- set edges
```

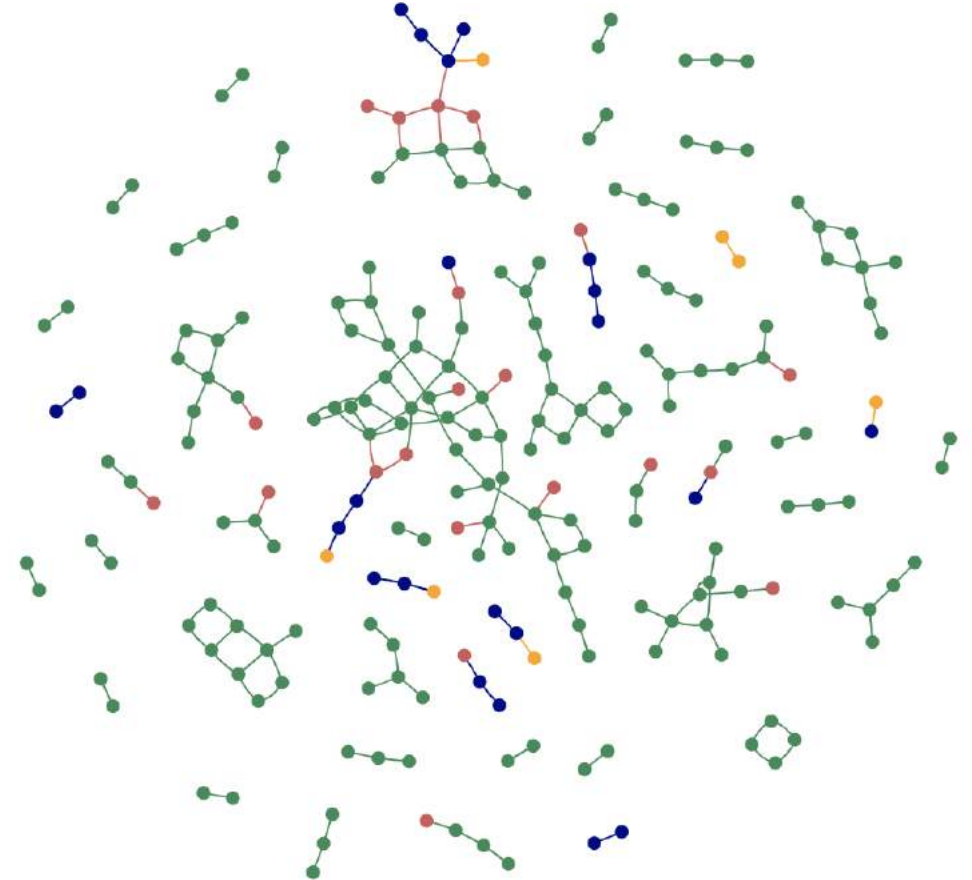
```
costa_network
```


Analyzing the network

- Zoom in to see the individual nodes.
- Remember that the green nodes are the non-vulnerable households, while the remaining nodes are vulnerable households.

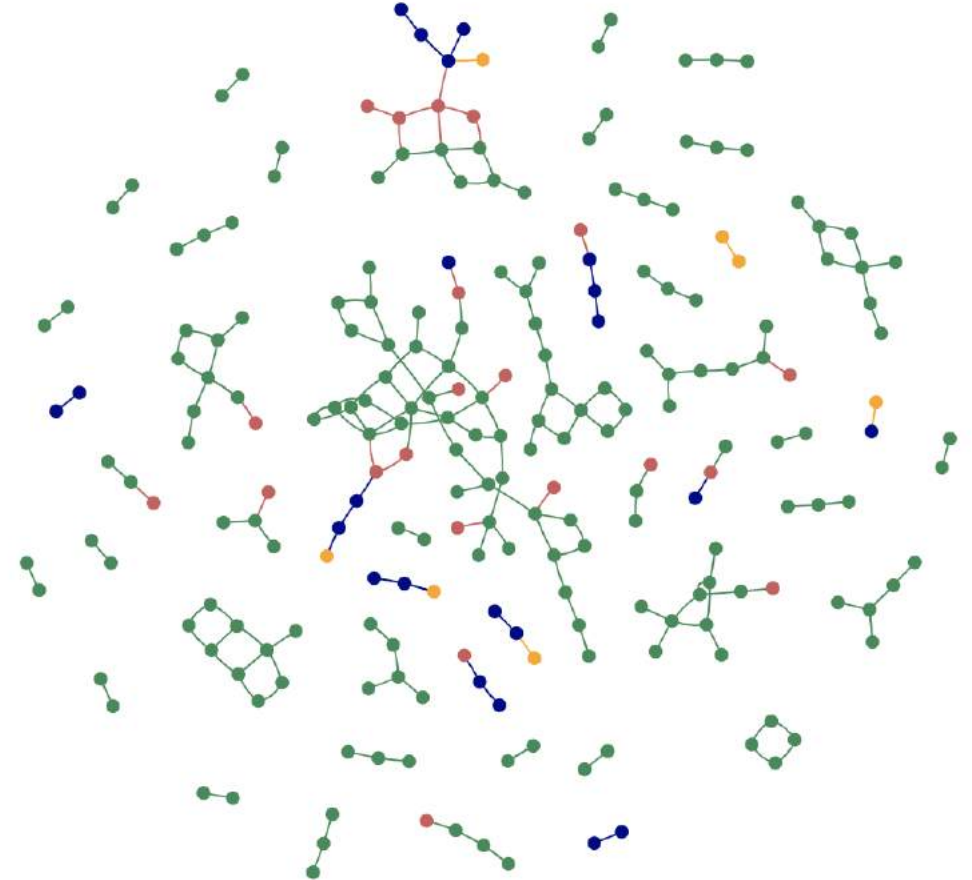
Polling questions:

- Which target has more nodes?
- Are the nodes for vulnerable households connected to the nodes for non-vulnerable households?
- What could this tell us about how similar the households are?



Analyzing the network

- We notice that there are:
 - more nodes with Target 4 (non-vulnerable households)
 - fewer nodes with Target = 1, 2, and 3 (vulnerable households)
- The nodes for vulnerable households do not appear very connected to the nodes for non-vulnerable households
- This could tell us that these households are not very similar



Customizing the network visualization

- We can add extra functionality using `visOptions` such as:
 - Highlighting nearest nodes when a single node is selected
 - Creating a dropdown menu to select specific nodes
- The entire list of `visOptions` can be found [here](#)
- Click on individual nodes to highlight them and select IDs from the dropdown

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

```
costa_network
```

Saving networks with htmlwidgets

- We can save the networks as html files to be able to share the files or use them at a later stage
- These html files can also be embedded in presentations, etc.

```
# 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(costa_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 2



Exercise 2



Module completion checklist

Objective	Complete
Summarize the concepts of distance matrix and network visualization	✓
Create a distance matrix for a given dataset	✓
Create nodes and edges dataframes	✓
Build and customize a Network HTMLwidget	✓

Summary

- Today we learned how to create networks using `visNetwork`
- A similar approach can be used to create other **htmlWidgets** using packages like `leaflet` for maps and `dygraphs` for time series
- The important thing is to transform your data into the shape the functions expect (like nodes and edges for `visNetwork`)
- Most packages have extensive customization capabilities built into their functions
- For any additional customization, you can use **JavaScript** since the packages were originally written in this language
- In the next module, we will learn about **RShiny** where we can create dashboards to combine all the visualizations we have covered

This completes our module
Congratulations!