DATA SOCIETY:

Interactive visualization with R - Network Graphs - 1

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

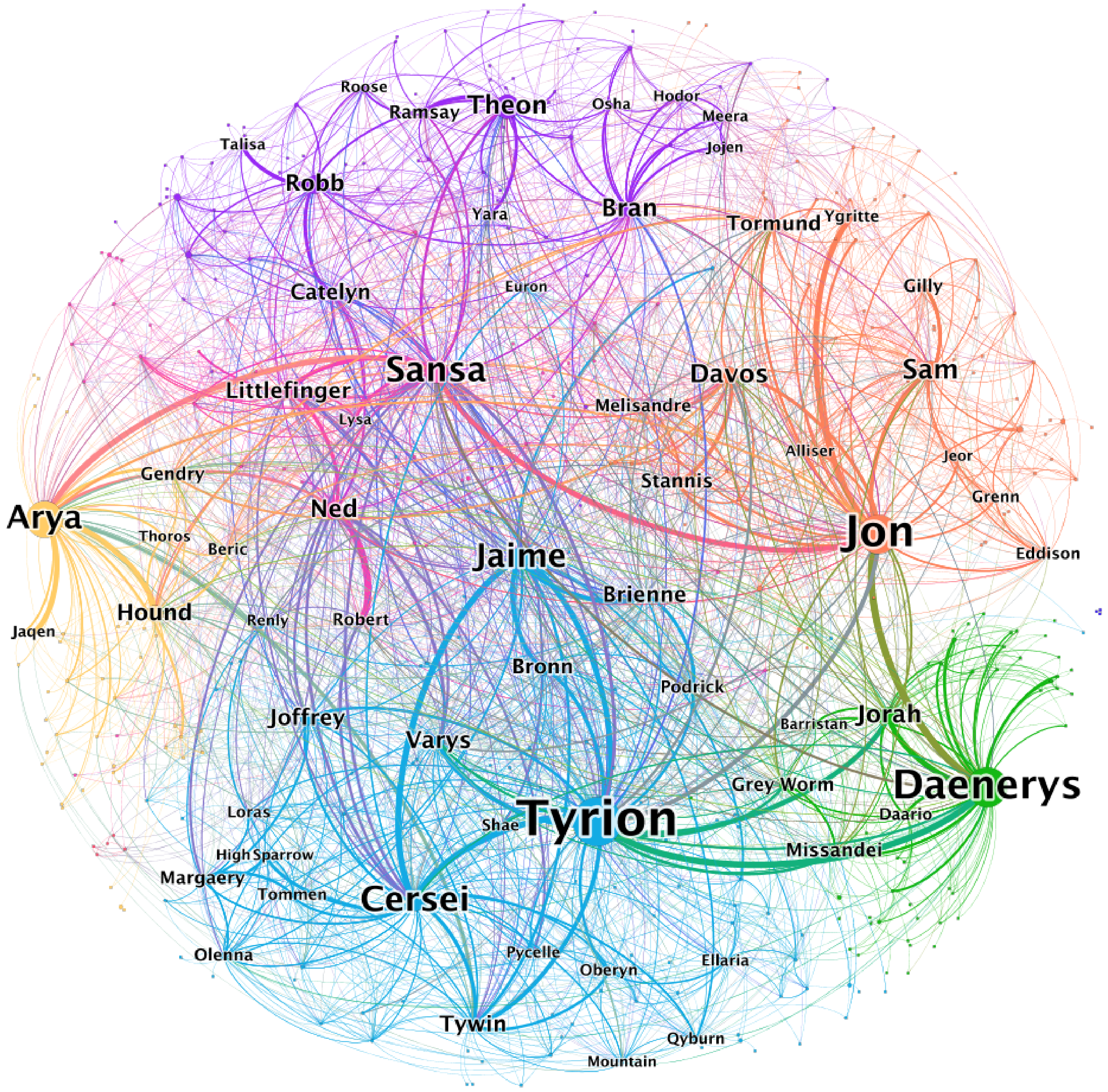
Network graphs: topic introduction

In this part of the course, we will cover the following concepts:

- Transform and prepare data for a network graph visualization
- Build and customize interactive network graphs

Warm-up

- R packages help create specific kinds of visualizations, and in this module, we will discuss network graphs, which display relationships between elements using simple links
- Take 5 minutes to ***explore network graphs*** based on the characters in the *Game of Thrones* book and television series
  - What insights can you draw from these visualizations?
  - Share your thoughts in the chat

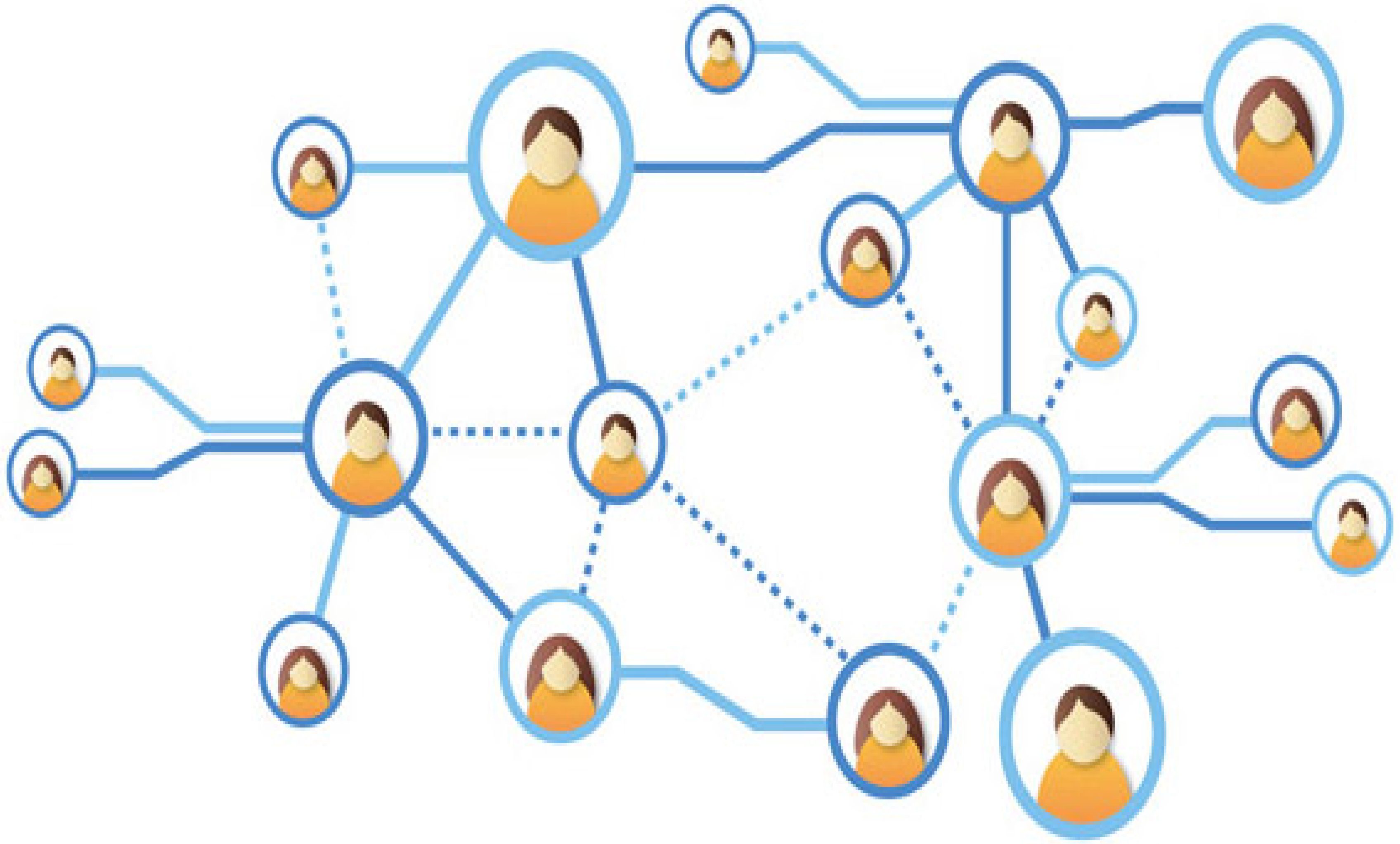


Module completion checklist

Objective	Complete
Summarize the concepts of distance matrix and network visualization	
Create a distance matrix for a given dataset	

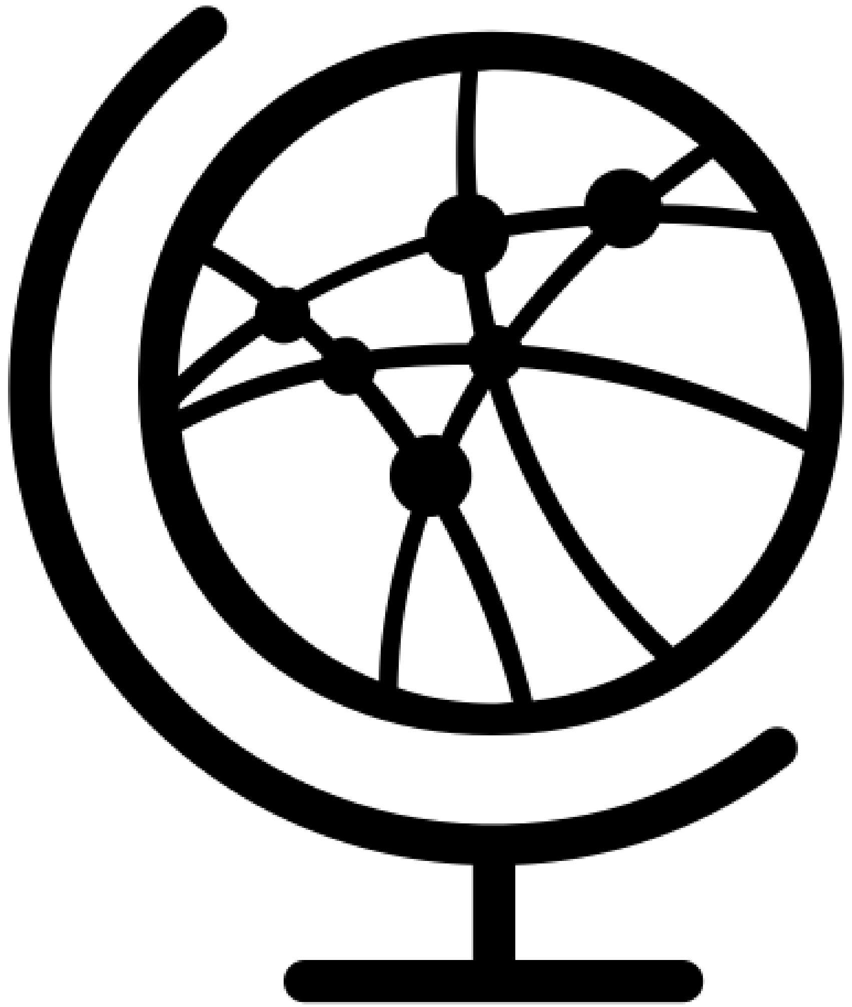
Network graphs

- Network visualization is an excellent 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
- A network graph may also be known as a link chart, a node-link diagram, or a network map



Different kinds of networks

- Within the framework of “nodes” and “edges,” network graphs may represent many different kinds of entities and relationships; for example:
    - the volume of communication between two organizational units
    - the geographical distance between two types of users
    - the flow of goods between two manufacturing sites
- Though beyond the scope of this course, edges can be imbued with data about direction, density, or strength

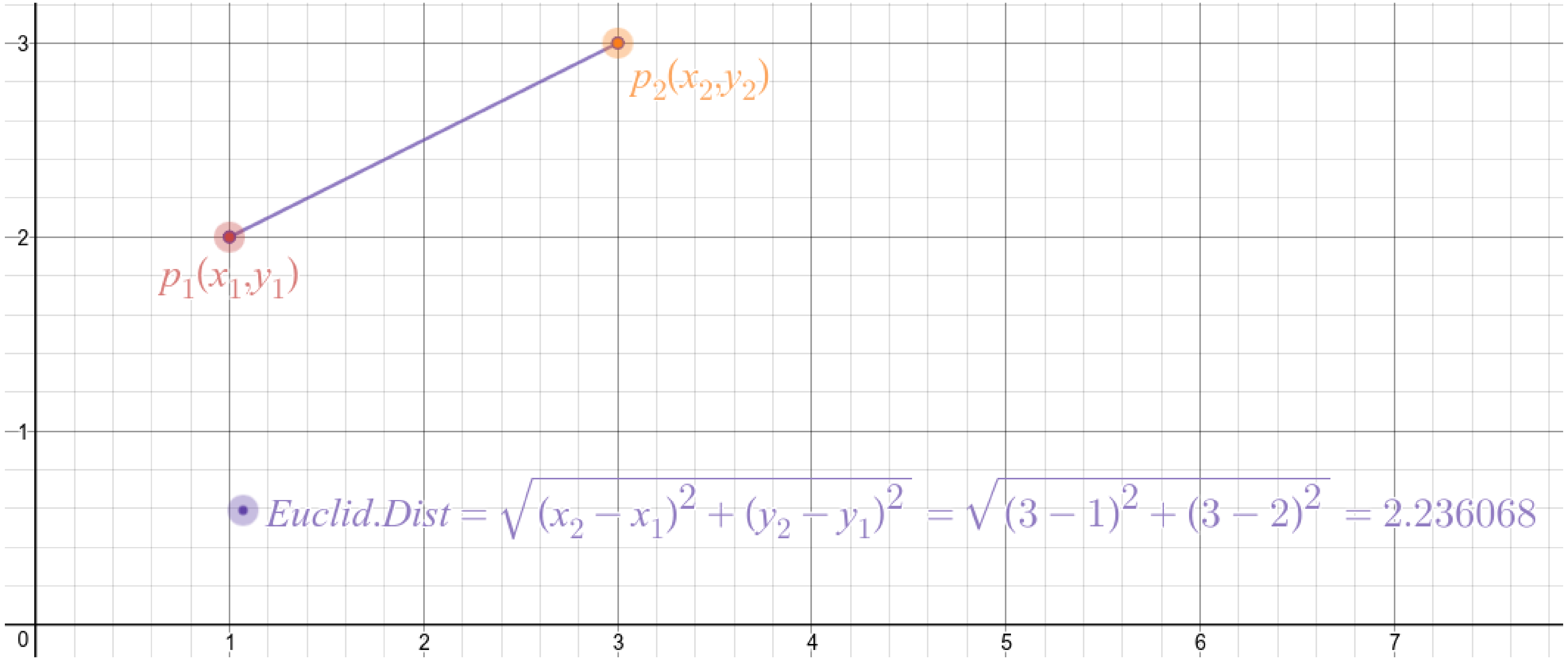


Distance and similarity

- In network theory, we use the distance between two nodes to describe their connection
- The smaller the distance, the more similar the two nodes are
- Depending on the situation, different distance metrics might be useful:
    - Euclidean
    - Manhattan
    - Binary
    - Minkowski
- In this module, we will use the **Euclidean distance** metric

Distance and similarity

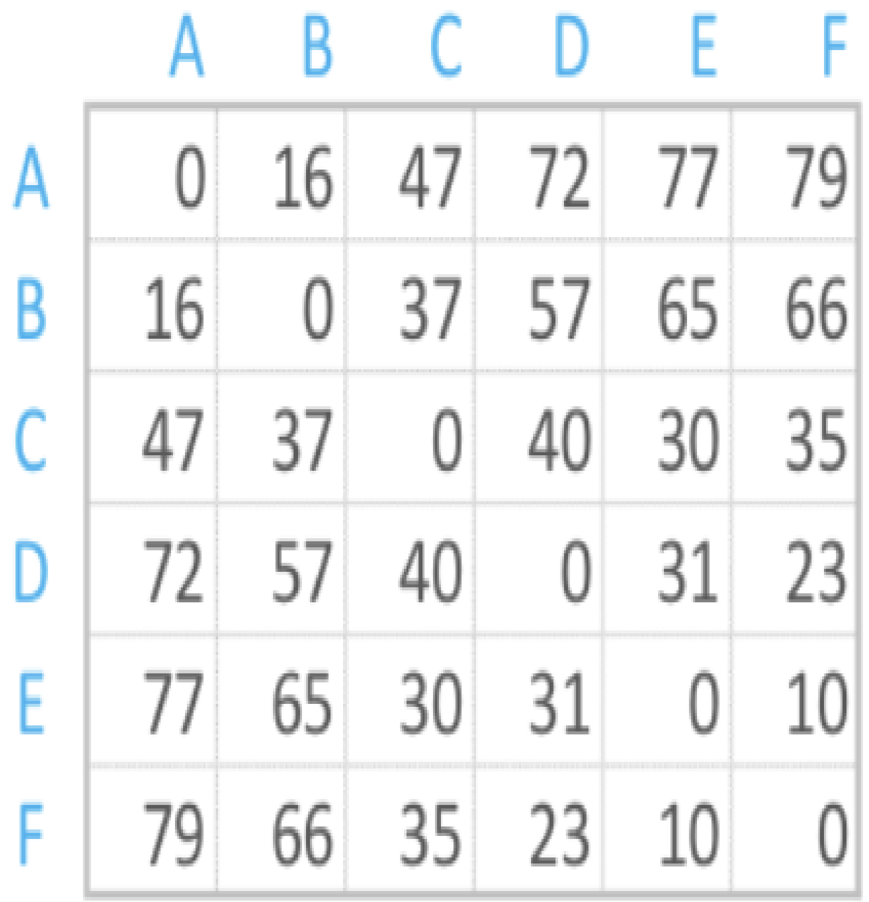
- The Euclidean distance between two points is the length of the line segment connecting them
- The distance formula can be extended for Euclidean spaces with more than two dimensions



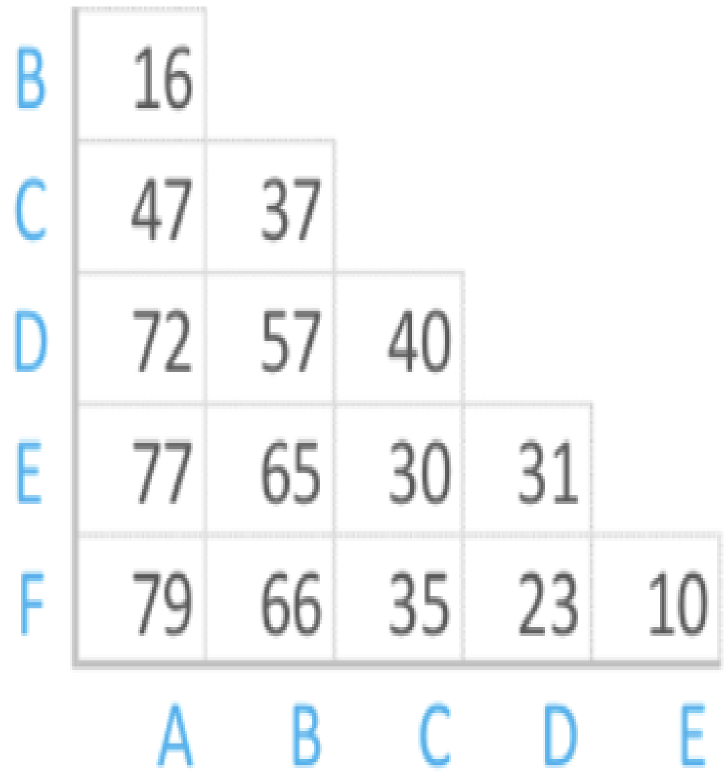
Distance matrix

- To generate a graph, we must pass the model a distance matrix

- A distance matrix for `N` nodes is of size `N * N`, where each value corresponds to the distance between a pair of nodes
- Since the distance matrix is **symmetrical** and values along the **diagonal** are 0, all of the required information is contained in 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



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

Module completion checklist

Objective	Complete
Summarize the concepts of distance matrix and network visualization	✔
Create a distance matrix for a given dataset	

Stroke: case study

- According to the World Health Organization (WHO), stroke is the 2nd leading cause of death globally
- ***Click here*** to see a dataset showing the results of a clinical trial of a Stroke drug survey on a sample of US adults
- Each row in the data provides relevant information about the adult, including whether they had a stroke or not

HTML widgets with JavaScript

- Rather than create a static network graph, we will render ours as an HTML widget
- The ***`htmlwidgets`*** package provides a framework for quickly creating R bindings to JavaScript libraries
- HTML widgets are helpful 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.

HTML widgets with JavaScript (cont'd)

- Some popular packages based on `htmlwidgets` are:
  - `leaflet` for maps
  - `dygraphs` for time series
  - `rthreejs` for interactive 3D graphics
- In this module we will use `visNetwork` to create an HTML widget for network visualization

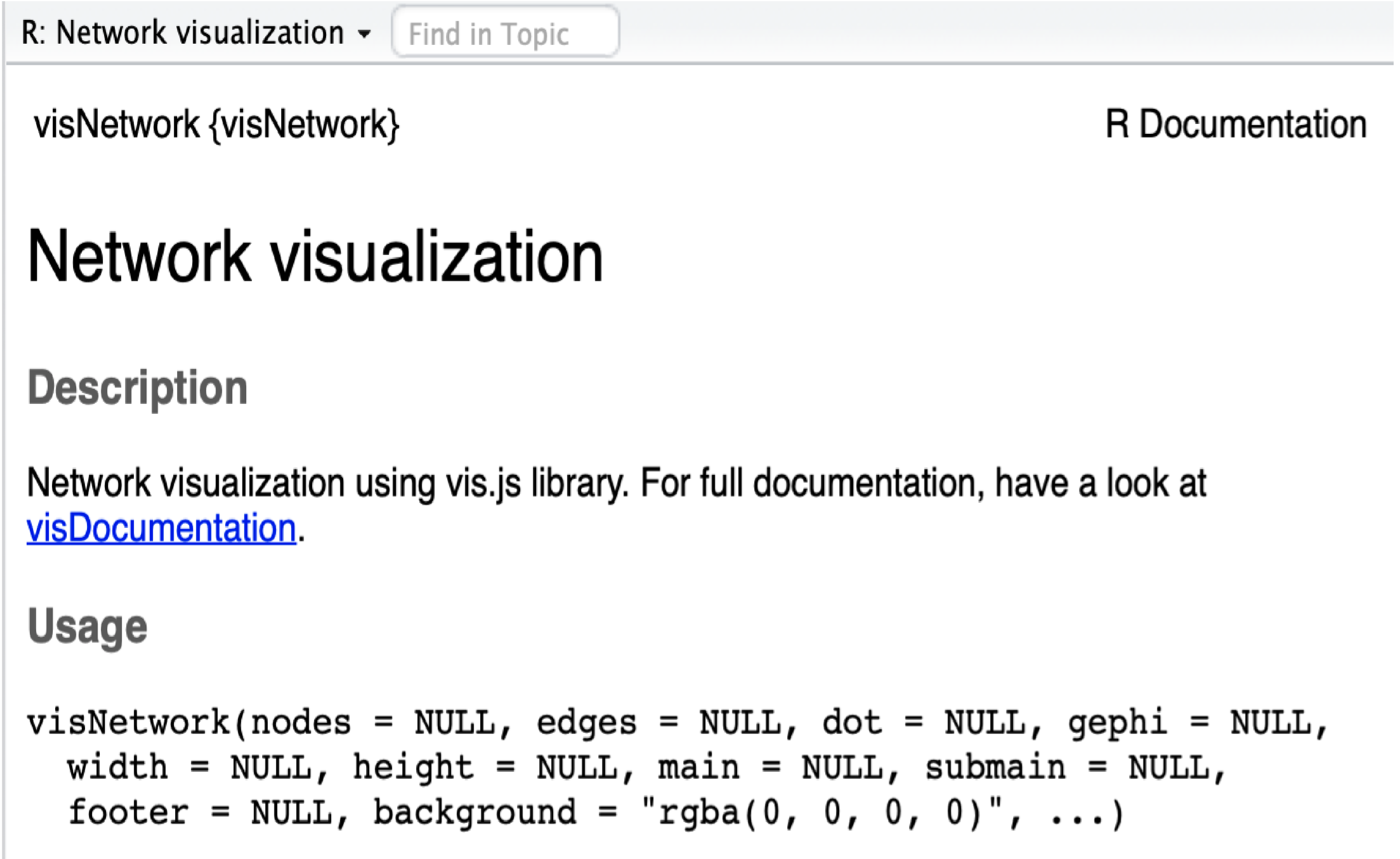
visNetwork

- visNetwork is an R package for network visualization, using the ***vis.js JavaScript library*** and based on `htmlwidgets`

```
library(visNetwork)

?visNetwork
```

- visNetwork needs at least two arguments to plot a basic network:
  - **nodes** dataframe with an `id` column
  - **edges** dataframe with `from` and `to` columns
- It has additional functions to customize the network and add interactivity



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)", ...)
```

Directory settings

- In order to maximize the efficiency of your workflow, use the `box` package and encode your directory structure into `variables`
- Let the `main_dir` be the variable corresponding to your materials folder

```
# Set `main_dir` to the location of your materials folder.

path = box::file()
main_dir = dirname(dirname(path))
```

Directory settings (cont’d)

- We will store all datasets in the `data` directory inside the materials folder in your environment; hence we will save their path to a `data_dir` variable
- We will save all the plots in the `plots` directory corresponding to `plot_dir` variable

- To append one string to another, use `paste0` command and pass the strings you would like to paste together

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

Loading packages

- Here are the packages we will need for today:

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

Loading the dataset

- Before creating a network visualization, let’s profile the dataset

```
# Read in the healthcare-dataset-stroke dataset.
hds = read.csv(file = file.path(data_dir,"/healthcare-dataset-stroke-data.csv"), #<- provide file
path
               header = TRUE,            #<- if file has header set to TRUE
               stringsAsFactors = FALSE) #<- read strings as characters, not as factors
# View the dimensions of the dataset.
dim(hds)
```

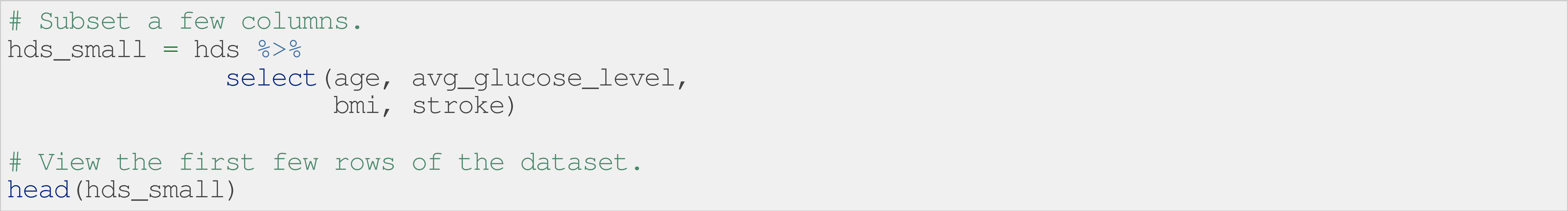
```
[1] 5110   12
```

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

```
     id gender age hypertension heart_disease ever_married     work_type
1  9046   Male  67            0             1          Yes       Private
2 51676 Female  61            0             0          Yes Self-employed
3 31112   Male  80            0             1          Yes       Private
4 60182 Female  49            0             0          Yes       Private
5  1665 Female  79            1             0          Yes Self-employed
  Residence_type avg_glucose_level  bmi
1          Urban            228.69 36.6
2          Rural            202.21   NA
3          Rural            105.92 32.5
4          Urban            171.23 34.4
5          Rural            174.12 24.0
```

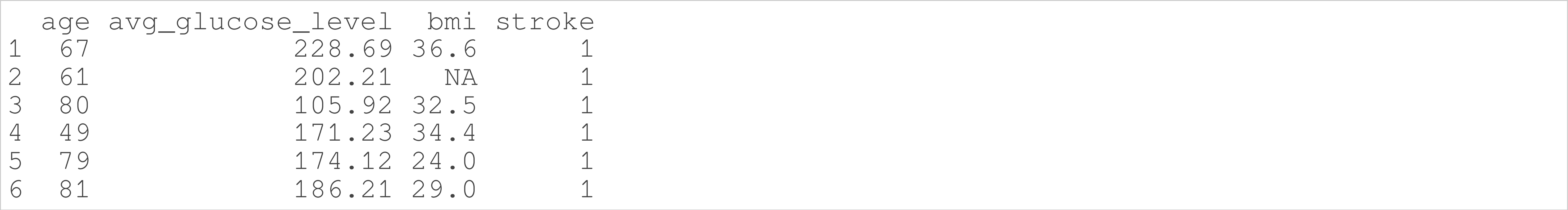
Subsetting the dataset

- Since network visualization is more straightforward with smaller datasets, we will subset the HDS dataset and use only a few columns: age, avg_glucose_level, bmi, and stroke



```r
# Subset a few columns.
hds_small = hds %>%
               select(age, avg_glucose_level,
                      bmi, stroke)

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



```
  age avg_glucose_level  bmi stroke
1  67            228.69 36.6      1
2  61            202.21   NA      1
3  80            105.92 32.5      1
4  49            171.23 34.4      1
5  79            174.12 24.0      1
6  81            186.21 29.0      1
```

Cleaning the dataset

- Next, we will remove NAs and duplicate rows from the dataset

```
# Convert BMI column to numerical
hds_small$bmi <- as.numeric(as.character(hds_small$bmi))
# Remove rows with NA.
hds_small = na.omit(hds_small)

# We keep only the unique rows since duplicate rows would have a distance of 0.
hds_small= unique(hds_small)

head(hds_small)
```

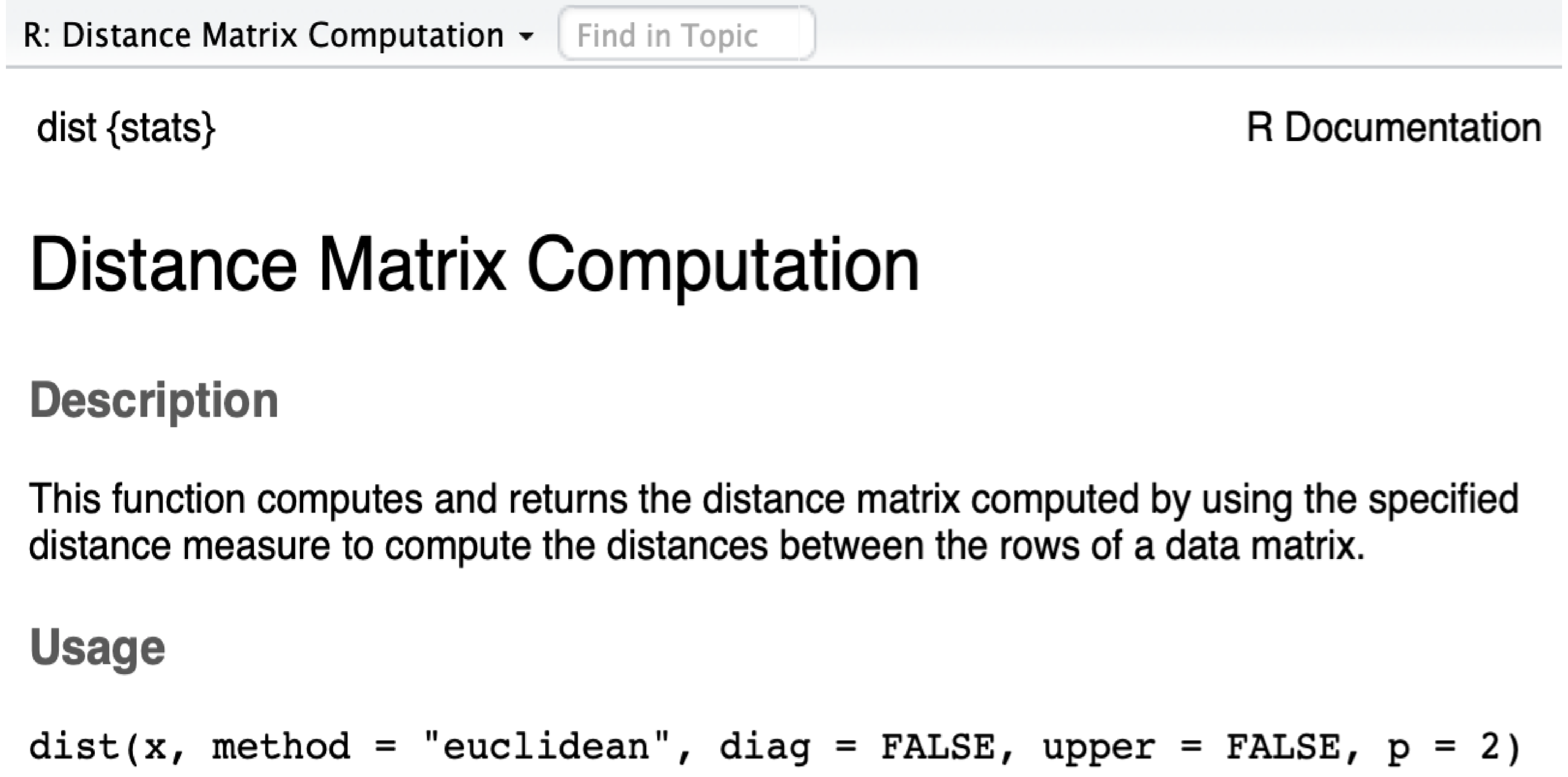
```
  age avg_glucose_level  bmi stroke
1  67            228.69 36.6      1
3  80            105.92 32.5      1
4  49            171.23 34.4      1
5  79            174.12 24.0      1
6  81            186.21 29.0      1
7  74             70.09 27.4      1
```

Measuring similarity

- To measure the similarity between households, we will use the `dist()` function

```
?dist
```

- `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 {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 (contd)

- Because some of our variables, like `age` and `avg_glucose_level`, are at very different scales, we must normalize the distance to values between 0 and 1 for easy interpretation

```
# Create distance matrix.
hds_distance = dist(hds_small)

# `dist` returns the lower triangle of the distance matrix as a vector.
head(hds_distance)
```

```
[1] 123.52442  60.25356  57.27691  45.36861 159.02075 135.02196
```

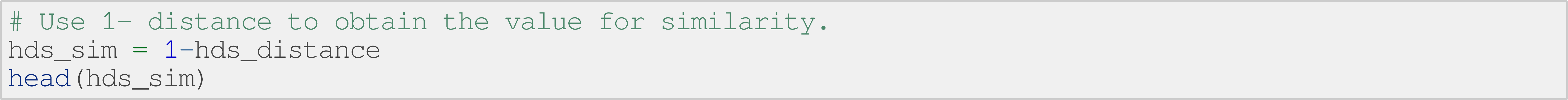
```
# Normalize the distances to values between 0 and 1.
hds_distance = hds_distance/max(hds_distance)

head(hds_distance)
```

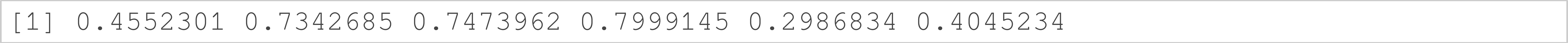
```
[1] 0.5447699 0.2657315 0.2526038 0.2000855 0.7013166 0.5954766
```

Interpreting similarity

- The **greater the distance between two observations, the more different** they are
- By subtracting the value of the normalized distance from 1, we can get an indication of how similar the two observations are



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



```
[1] 0.4552301 0.7342685 0.7473962 0.7999145 0.2986834 0.4045234
```

Knowledge check

Module completion checklist

Objective	Complete
Summarize the concepts of distance matrix and network visualization	✔
Create a distance matrix for a given dataset	✔

Congratulations on completing this module!

You are now ready to try tasks 1-2 in the Exercise for this topic