Network Science in R - A Tidy Approach

Lessons from DataCamp

Contents

Introduction

The following document outlines the written portion of the lessons from DataCamp's Network Science in R - A Tidy Approach. This requires Intermediate R-Knowledge and knowledge of the Tidyverse package.

As a note: All text is completely copied and pasted from the course. There are instances where the document refers to the "editor on the right", please note, that in this notebook document all of the instances are noted in the "r-chunks" (areas containing working r-code), which occurs below the text, rather than to the right. Furthermore, This lesson contained instructional videos at the beginning of new concepts that are not detailed in this document. However, even without these videos, the instructions are quite clear in indicating what the code is accomplishing.

If you have this document open on "R-Notebook", simply click "run" -> "Run all" (Or just press 'ctrl + alt + r'), let the "r-chunks" run (This might take a bit of time) then click "Preview". There are 5 necessary datasets to run this program, please create an r-project with this data or set a working directory (required files names are available in the "Required data for this session" section)

This document was created by Neil Yetz on 05/19/2018. Please send any questions or concerns in this document to Neil at ndyetz@gmail.com

Required packages for this session

Below are the install.packages and libraries you will need to have in order to run this session successfully.

```
#install.packages("readr")
#install.packages("igraph")
#install.packages("ggraph")

library(readr)
library(igraph)
library(dplyr)
library(ggplot2)
library(ggraph)
```

Required data for this session

```
#nodes  <- read_csv("nodes.csv")
#ties   <- read_csv("ties.csv")
```

Course Description

If you've ever wanted to understand more about social networks, information networks, or even the neural networks of our brains, then you need to know network science! It will demonstrate network analysis using several R packages, including dplyr, ggplot2, igraph, ggraph as well as visNetwork. You will take on the role of Interpol Analyst and investigate the terrorist network behind the Madrid train bombing in 2004. Following the course, you will be able to analyse any network with basic centrality and similarity measures and create beautiful and interactive network visualizations.

Chapter 1: The hubs of the network

The challenge in this chapter is to spot the most highly connected terrorists in the network. We will first import the dataset and build the network. Then we will learn how to visualize it in different layouts using ggraph package. Later on, we will compute two basic yet important centrality measures in network science - degree and strength. We will use them to spot highly connected terrorists. We will finally touch two alternative centrality measures, betweenness and closeness.

Explore the dataset

In this first exercise, you will explore the dataset. You will use the package `readr` to read the `nodes` and `ties` datasets from CSV files into variables in R. For your convenience, the package readr is already loaded into the workspace.

INSTRUCTIONS

Read the `nodes` and `ties` into variables with the `read_csv()` function. Print nodesand then ties to explore the nodes and ties in the console. How many nodes and ties are in the dataset?

```r
# read the nodes file into the variable nodes
nodes <- read_csv("nodes.csv")
```

```
## Parsed with column specification:
## cols(
##   id = col_integer(),
##   name = col_character()
## )
```

```r
# read the ties file into the variable ties
ties <- read_csv("ties.csv")
```

```
## Parsed with column specification:
## cols(
##   from = col_integer(),
##   to = col_integer(),
##   weight = col_integer()
## )
```

```r
# print nodes
nodes
```

```
## # A tibble: 64 x 2
##       id name
##    <int> <chr>
##  1     1 Jamal Zougam
##  2     2 Mohamed Bekkali
##  3     3 Mohamed Chaoui
##  4     4 Vinay Kholy
##  5     5 Suresh Kumar
##  6     6 Mohamed Chedadi
##  7     7 Imad Eddin Barakat
##  8     8 Abdelaziz Benyaich
##  9     9 Abu Abderrahame
## 10    10 Omar Dhegayes
## # ... with 54 more rows
```

```r
# print ties
ties
```

```
## # A tibble: 243 x 3
##     from    to weight
##    <int> <int>  <int>
##  1     1     2      1
##  2     1     3      3
##  3     1     4      1
##  4     1     5      1
##  5     1     6      1
##  6     1     7      4
##  7     1     8      1
##  8     1     9      1
##  9     1    11      4
## 10     1    12      1
## # ... with 233 more rows
```

Build and explore the network (part 1)

In this exercise, you are going to begin using the `igraph` package. This package lets you analyze data that are represented as networks, which are also called graphs by mathematicians. In particular, you will learn how to build a network from a data frame and explore the nodes and ties of the network.

For your convenience, the package `igraph` and the data frames nodes and ties are already loaded into the workspace.

INSTRUCTIONS

Use `graph_from_data_frame()` to build a network from the data frame ties and save it as `g`. Print the network to discover the number of nodes and ties.

Explore the nodes in `g` with `V()` and print the number of nodes with `vcount()`

Explore the ties in `g` with `E()` and print the number of ties with `ecount()`

```r
# make the network from the data frame ties and print it
g <- graph_from_data_frame(ties, directed = FALSE, vertices = nodes)
g
```

```
## IGRAPH 7257d2a UNW- 64 243 --
## + attr: name (v/c), weight (e/n)
## + edges from 7257d2a (vertex names):
##  [1] Jamal Zougam--Mohamed Bekkali     Jamal Zougam--Mohamed Chaoui
##  [3] Jamal Zougam--Vinay Kholy         Jamal Zougam--Suresh Kumar
##  [5] Jamal Zougam--Mohamed Chedadi     Jamal Zougam--Imad Eddin Barakat
##  [7] Jamal Zougam--Abdelaziz Benyaich  Jamal Zougam--Abu Abderrahame
##  [9] Jamal Zougam--Amer Azizi          Jamal Zougam--Abu Musad Alsakaoui
## [11] Jamal Zougam--Mohamed Atta        Jamal Zougam--Ramzi Binalshibh
## [13] Jamal Zougam--Mohamed Belfatmi    Jamal Zougam--Said Bahaji
## [15] Jamal Zougam--Galeb Kalaje        Jamal Zougam--Abderrahim Zbakh
## + ... omitted several edges
```

```r
# explore the set of nodes
V(g)
```

```
## + 64/64 vertices, named, from 7257d2a:
##  [1] Jamal Zougam           Mohamed Bekkali
##  [3] Mohamed Chaoui         Vinay Kholy
##  [5] Suresh Kumar           Mohamed Chedadi
##  [7] Imad Eddin Barakat     Abdelaziz Benyaich
##  [9] Abu Abderrahame        Omar Dhegayes
## [11] Amer Azizi             Abu Musad Alsakaoui
## [13] Mohamed Atta           Ramzi Binalshibh
## [15] Mohamed Belfatmi       Said Bahaji
## [17] Galeb Kalaje           Abderrahim Zbakh
## [19] Farid Oulad Ali        José Emilio Suárez
## + ... omitted several vertices
```

```r
# print the number of nodes
vcount(g)
```

```
## [1] 64
```

```r
# explore the set of ties
E(g)
```

```
## + 243/243 edges from 7257d2a (vertex names):
```

```
##  [1] Jamal Zougam--Mohamed Bekkali
##  [2] Jamal Zougam--Mohamed Chaoui
##  [3] Jamal Zougam--Vinay Kholy
##  [4] Jamal Zougam--Suresh Kumar
##  [5] Jamal Zougam--Mohamed Chedadi
##  [6] Jamal Zougam--Imad Eddin Barakat
##  [7] Jamal Zougam--Abdelaziz Benyaich
##  [8] Jamal Zougam--Abu Abderrahame
##  [9] Jamal Zougam--Amer Azizi
## [10] Jamal Zougam--Abu Musad Alsakaoui
## + ... omitted several edges
```

```r
# print the number of ties
ecount(g)
```

```
## [1] 243
```

Build and explore the network (part 2)

A network built using `igraph` can have attributes. These include:

- Network attributes: properties of the entire network
- Node attributes: properties of nodes
- Tie attributes: properties of ties

In this exercise, we will explore all these types of attributes.

`igraph` and the variable `g` containing the network are already loaded into the workspace.

INSTRUCTIONS

Give the name "Madrid network" to the network. Then print the network name attribute.

Add node attribute `id` and set the id numbers from 1 to the number of nodes of the network. Then print the `id`attribute.

Print the tie `weight` attribute.

Print the network. Can you spot the different types of attributes?

```r
# give the name "Madrid network" to the network and print the network `name` attribute
g$name <- "Madrid network"
g$name
```

```
## [1] "Madrid network"
```

```r
# add node attribute id and print the node `id` attribute
V(g)$id <- c(1:length(V(g)))
V(g)$id
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
```

```r
# print the tie `weight` attribute
E(g)$weight
```

```
##   [1] 1 3 1 1 1 4 1 1 4 1 1 2 2 2 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 3 1 1
##  [36] 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
##  [71] 1 1 1 1 1 1 1 1 3 1 1 1 2 2 3 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 1 1 2 1
## [106] 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 3 2 1 1 3 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
# print the network and spot the attributes
g
```

```
## IGRAPH 7257d2a UNW- 64 243 -- Madrid network
## + attr: name (g/c), name (v/c), id (v/n), weight (e/n)
## + edges from 7257d2a (vertex names):
##  [1] Jamal Zougam--Mohamed Bekkali     Jamal Zougam--Mohamed Chaoui
##  [3] Jamal Zougam--Vinay Kholy         Jamal Zougam--Suresh Kumar
##  [5] Jamal Zougam--Mohamed Chedadi     Jamal Zougam--Imad Eddin Barakat
##  [7] Jamal Zougam--Abdelaziz Benyaich  Jamal Zougam--Abu Abderrahame
##  [9] Jamal Zougam--Amer Azizi          Jamal Zougam--Abu Musad Alsakaoui
## [11] Jamal Zougam--Mohamed Atta        Jamal Zougam--Ramzi Binalshibh
## [13] Jamal Zougam--Mohamed Belfatmi    Jamal Zougam--Said Bahaji
## [15] Jamal Zougam--Galeb Kalaje        Jamal Zougam--Abderrahim Zbakh
## + ... omitted several edges
```

Visualize the network (part 1)

Welcome to the `ggraph` package! In this course, we will use this package to visualize networks.

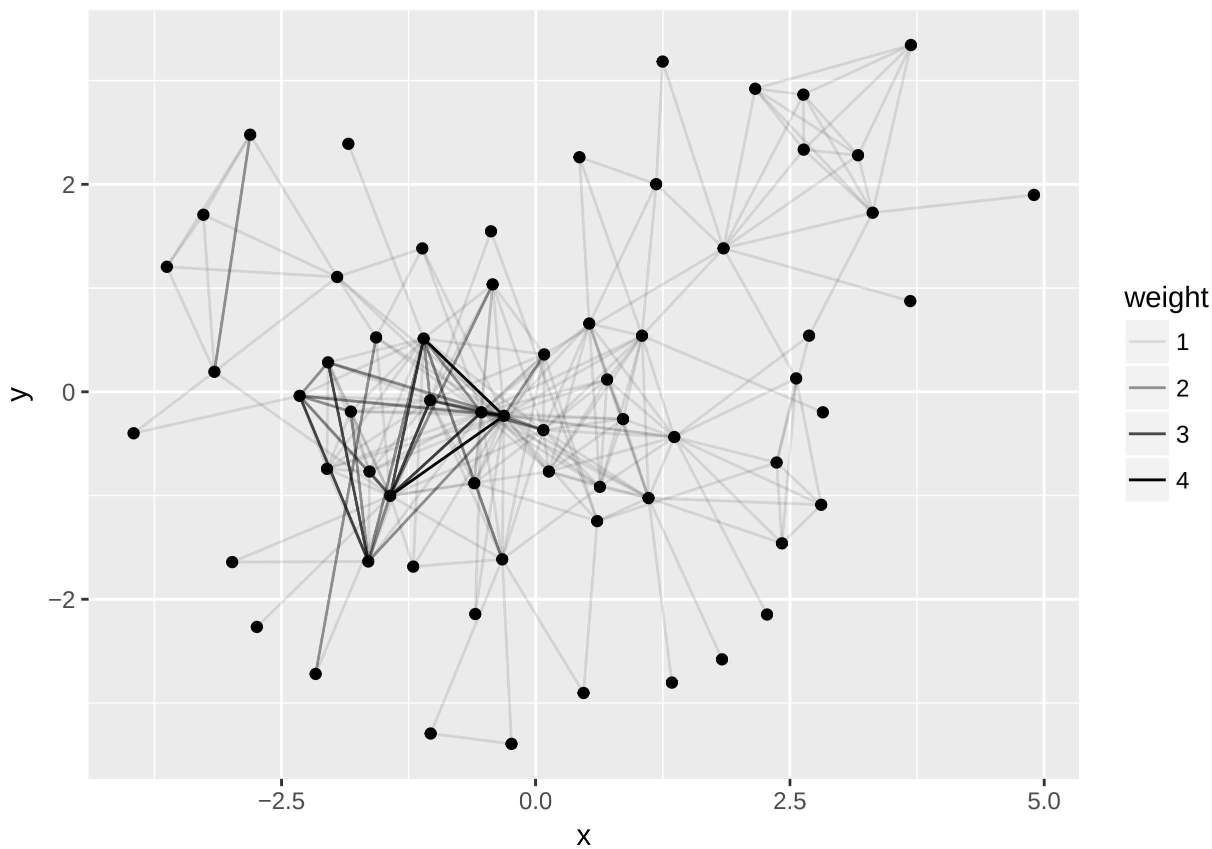
The package `ggraph` extends `ggplot2` by using geometries to visualize the nodes (`geom_node_point`) and ties (`geom_edge_link`) of a network.

If you already know a bit of `ggplot2`, you will learn `ggraph` quickly! For your convenience, `ggraph` is already loaded into the workspace, the graph theme is set with the function `set_graph_style()`, and the variable `g` containing the network is at your disposal.

INSTRUCTIONS 1/2

Visualize the network with the Kamada-Kawai layout and set the transparency of ties (`alpha`) equal to weight. Can you visually spot the important nodes and ties?

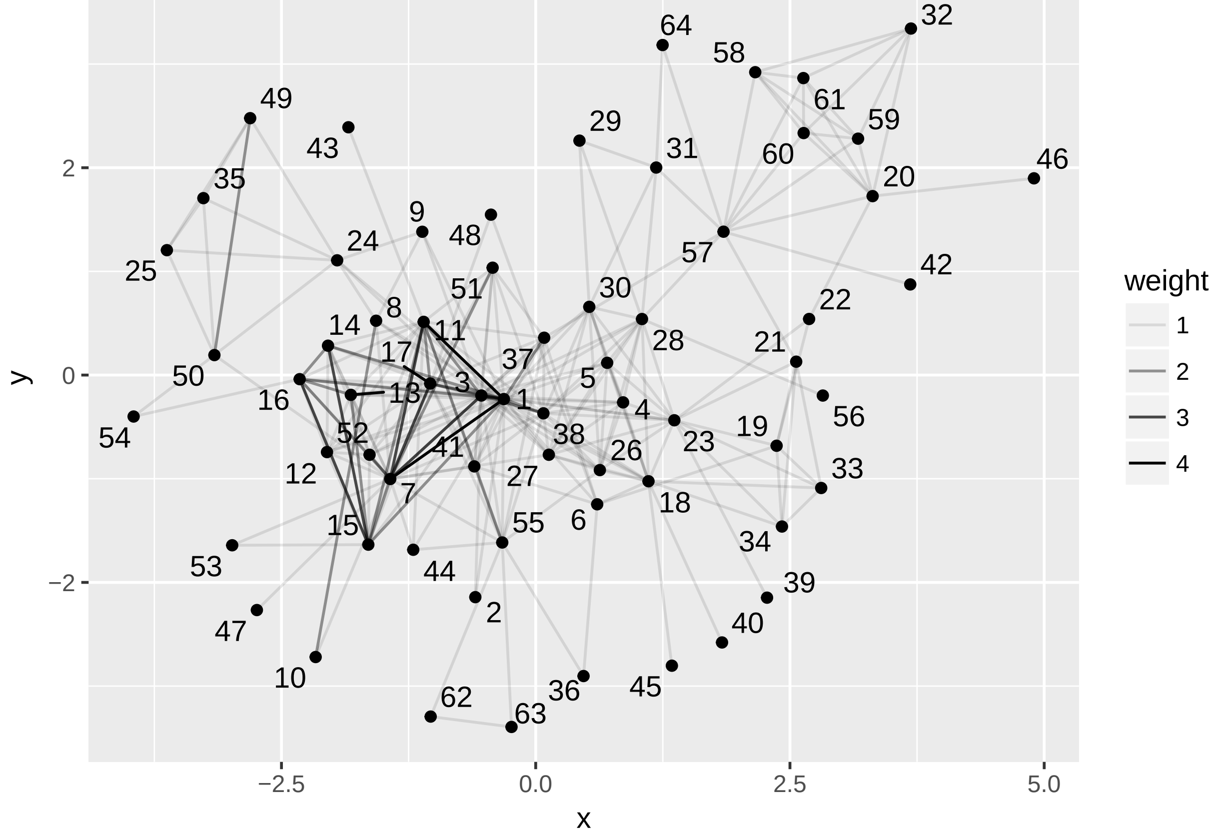
```
# visualize the network with layout Kamada-Kawai
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(alpha = weight)) +
  geom_node_point()
```



INSTRUCTIONS 2/2

Add a label to the nodes that corresponds with their ids using the `geom_node_text()` geometry and make sure to prevent the labels from overlapping.

```
# add an id label to nodes
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(alpha = weight)) +
  geom_node_point()  +
  geom_node_text(aes(label = id), repel = TRUE)
```



Visualize the network (part 2)

In the previous exercise, we used a force-directed layout (the Kamada-Kawai layout) to visualize the nodes and ties, in other words, it placed tied nodes at equal distances, so that all ties had roughly the same length.

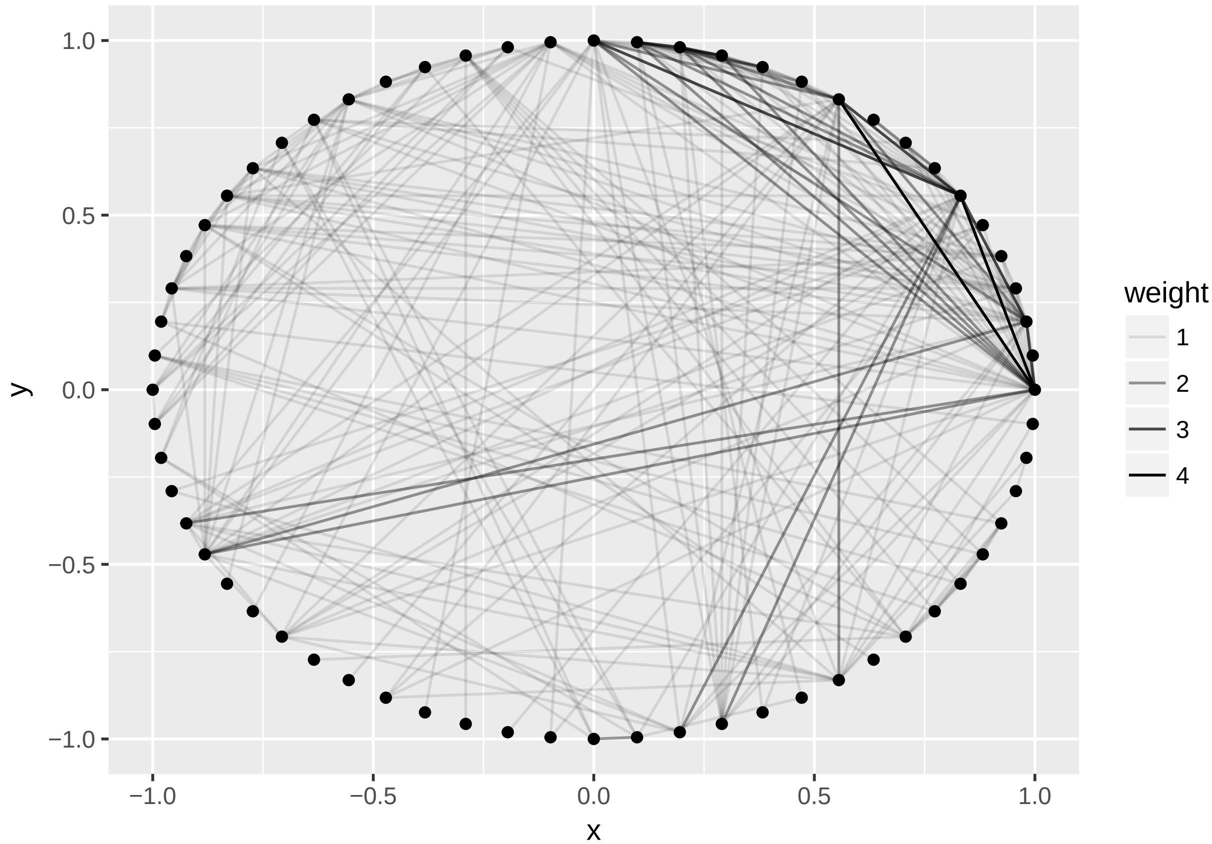
In this exercise, we will use two alternative layouts: circle, which places nodes on a circle, and grid, which places nodes on a grid.

For your convenience, the variable `g` containing the network is at your disposal.

INSTRUCTIONS 1/2

Visualize the network with a circular layout. Set tie transparency proportional to weight.

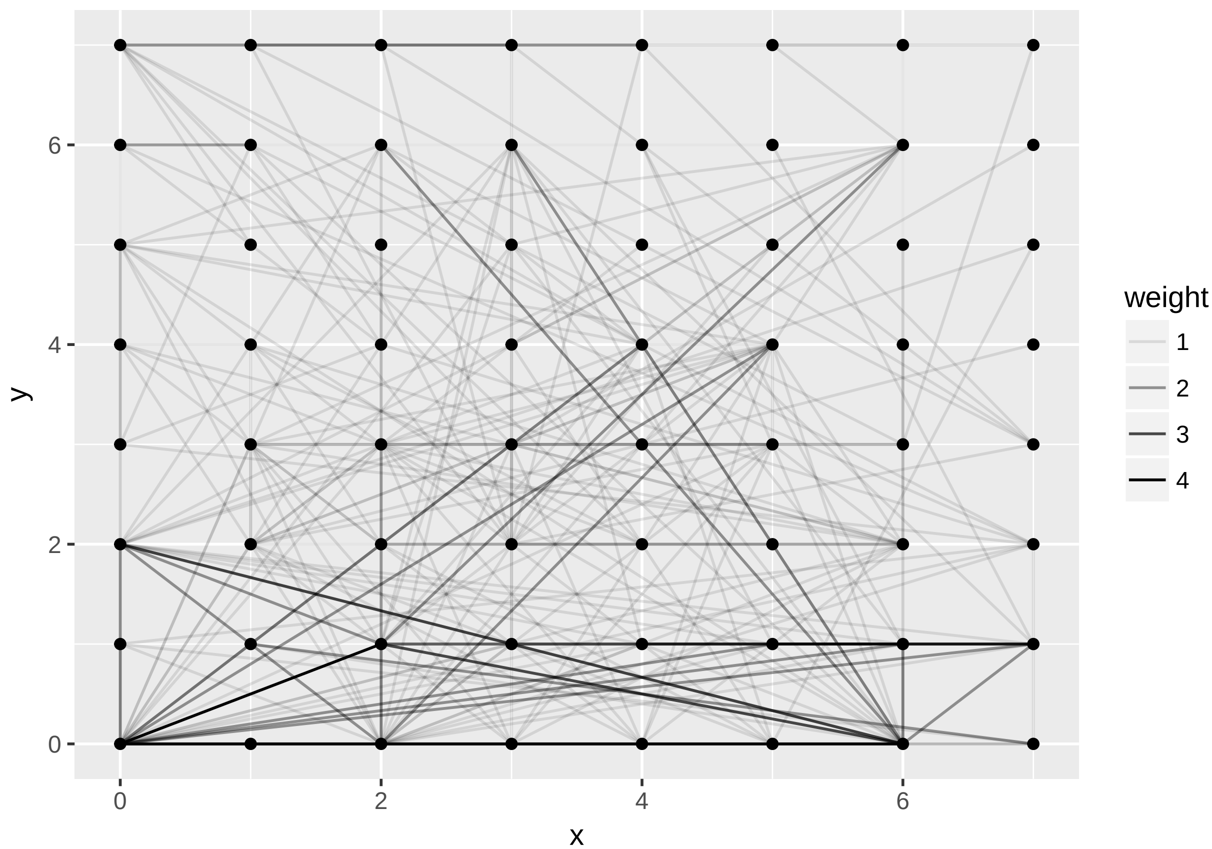
```r
# visualize the network with circular layout. Set tie transparency proportional to its weight
ggraph(g, layout = "in_circle") +
  geom_edge_link(aes(alpha = weight)) +
  geom_node_point()
```



INSTRUCTIONS 2/2

Visualize the network with a grid layout. Set tie transparency proportional to weight.

```
# visualize the network with grid layout. Set tie transparency proportional to its weight
ggraph(g, layout = "grid") +
  geom_edge_link(aes(alpha = weight)) +
  geom_node_point()
```



Find the most connected terrorists

The challenge of this exercise is to spot the most connected terrorists of the train bombing network. We will take advantage of the most simple and popular centrality measure in network science: degree centrality.

You will use both `igraph` and `dplyr`, which are already loaded in the workspace. The variables `g`, which contains the network, and a data frame, `nodes`, which contains the nodes of the network are also pre-loaded.

Before starting, search on Wikipedia for `"Jamal Zougam"` to check whether he was involved in the bombings.

INSTRUCTIONS

Use `degree()` to compute the degrees of nodes and save them in a variable `dgr`.

Mutate the data frame `nodes`, add the `degree` variable, and set it to `dgr`.

Add a node attribute `degree` to the network using the variable `dgr`.

Arrange the terrorists in the `nodes` data frame in decreasing order of degree.

```
# compute the degrees of the nodes
dgr <- degree(g)

# add the degrees to the data frame object
nodes <- mutate(nodes, degree = dgr)

# add the degrees to the network object
V(g)$degree <- dgr
```

```r
# arrange the terrorists in decreasing order of degree
arrange(nodes, -degree)
```

```
## # A tibble: 64 x 3
##       id name               degree
##    <int> <chr>               <dbl>
##  1     1 Jamal Zougam          29.
##  2     3 Mohamed Chaoui        27.
##  3     7 Imad Eddin Barakat    22.
##  4    11 Amer Azizi            18.
##  5    38 Said Berrak           17.
##  6    17 Galeb Kalaje          16.
##  7    23 Naima Oulad Akcha     16.
##  8    18 Abderrahim Zbakh      15.
##  9    28 Jamal Ahmidan         14.
## 10    55 Mohamed El Egipcio    13.
## # ... with 54 more rows
```

Find the most strongly connected terrorists

The challenge in this exercise is to spot the most strongly connected terrorists of the train bombing network. We will exploit another centrality measure in network science: strength centrality.

Again, you will take advantage of `igraph` and `dplyr`, which are already loaded in the workspace. The variable `g`, which contains the network, and the data frame nodes, which contains the nodes of the network are at your disposal.

INSTRUCTIONS

Use `strength()` to compute the strength of the nodes.

Mutate the data frame `nodes`, add the `strength` variable, and set it to `stg`.

Add a node attribute, `strength`, to the network using the `stg` variable.

Arrange the terrorists in decreasing order of strength and degree. Do you notice any correlation between the two?

```r
# compute node strengths
stg <- strength(g)

# add strength to the data frame object
nodes <- mutate(nodes, strength = stg)

# add strength to the network object
V(g)$strength <- stg

# arrange terrorists in decreasing order of strength and then in decreasing order of degree
arrange(nodes, -strength)
```

```
## # A tibble: 64 x 4
##       id name               degree strength
##    <int> <chr>               <dbl>    <dbl>
##  1     1 Jamal Zougam          29.      43.
##  2     7 Imad Eddin Barakat    22.      35.
##  3     3 Mohamed Chaoui        27.      34.
##  4    11 Amer Azizi            18.      27.
```

```
## 5     17 Galeb Kalaje          16.      21.
## 6     15 Mohamed Belfatmi      11.      19.
## 7     38 Said Berrak           17.      19.
## 8     16 Said Bahaji           11.      17.
## 9     23 Naima Oulad Akcha     16.      16.
## 10    18 Abderrahim Zbakh      15.      15.
## # ... with 54 more rows
```

```r
arrange(nodes, -degree)
```

```
## # A tibble: 64 x 4
##       id name               degree strength
##    <int> <chr>               <dbl>    <dbl>
## 1      1 Jamal Zougam          29.      43.
## 2      3 Mohamed Chaoui        27.      34.
## 3      7 Imad Eddin Barakat    22.      35.
## 4     11 Amer Azizi            18.      27.
## 5     38 Said Berrak           17.      19.
## 6     17 Galeb Kalaje          16.      21.
## 7     23 Naima Oulad Akcha     16.      16.
## 8     18 Abderrahim Zbakh      15.      15.
## 9     28 Jamal Ahmidan         14.      14.
## 10    55 Mohamed El Egipcio    13.      14.
## # ... with 54 more rows
```

More on centrality

There are other centrality measures, but covering them all is beyond the scope of this course. A couple other centrality measures include:

- Betweenness: a measure that quantifies how often a node lies on the shortest path between other nodes.
- Closeness: a measure that quantifies how close a node is to all other nodes in the network in terms of shortest path distance.

Use the console to read the R documentation for betweenness (type `?betweenness`) and find out how to determine the terrorist with the highest betweenness in the network.

The data frame `nodes` and the network `g` are loaded for you.

```r
betweenness(g)
```

```
##          Jamal Zougam        Mohamed Bekkali         Mohamed Chaoui
##          264.04698674             0.05555556           373.59856993
##           Vinay Kholy           Suresh Kumar        Mohamed Chedadi
##            0.51347578             0.51347578            71.55768398
##    Imad Eddin Barakat     Abdelaziz Benyaich      Abu Abderrahame
##          252.18430109             8.17428476             0.05555556
##        Omar Dhegayes             Amer Azizi    Abu Musad Alsakaoui
##            0.00000000            91.35033809            15.31704753
##         Mohamed Atta        Ramzi Binalshibh       Mohamed Belfatmi
##            8.25333901             5.70133262            14.49931846
##          Said Bahaji           Galeb Kalaje       Abderrahim Zbakh
##           48.79917687            65.38033448           178.22501980
##      Farid Oulad Ali      José Emilio Suárez     Khalid Ouled Akcha
##           17.48896104            91.81241982            27.94761905
##           Rafa Zuher       Naima Oulad Akcha    Abdelkarim el Mejjati
```

```
##                 41.10778668            233.82992048            162.33874459
##          Anwar Adnan Ahmad           Basel Ghayoun   S B Abdelmajid Fakhet
##                  1.26666667             14.82657102             56.88621151
##               Jamal Ahmidan            Said Ahmidan           Hamid Ahmidan
##                260.20244528              0.00000000             53.16466626
##             Mustafa Ahmidan            Antonio Toro     Mohamed Oulad Akcha
##                 26.03946609              0.00000000              1.75000000
##         Rachid Oulad Akcha      Mamoun Darkazanli Fouad El Morabit Anghar
##                  1.75000000              1.26666667              1.12500000
##          Abdeluahid Berrak            Said Berrak Waanid Altaraki Almasri
##                267.32672392            109.38619492              0.00000000
##            Abddenabi Koujma           Otman El Gnaut       Abdelilah el Fouad
##                  0.00000000             20.80386386              0.00000000
##       Parlindumgan Siregar                El Hemir       Anuar Asri Rifaat
##                  0.00000000              4.56662067              0.00000000
##                Rachid Adli      Ghasoub Al Albrash             Said Chedadi
##                  0.00000000              0.00000000              0.92821068
##             Mohamed Bahaiah           Taysir Alouny   OM. Othman Abu Qutada
##                  0.00000000             45.09341399              3.95821092
##                      Shakur            Driss Chebli             Abdul Fatal
##                  7.67785202              1.34090909              2.81175769
##          Mohamed El Egipcio      Nasredine Boushoa          Semaan Gaby Eid
##                171.21753421              0.00000000            448.45887998
##               Emilio Llamo            Ivan Granados     Raul Gonzales Perez
##                  9.85939504              9.85939504              9.85939504
##               El Gitanillo          Moutaz Almallah         Mohamed Almallah
##                  9.85939504              0.00000000              0.00000000
##             Yousef Hichman
##                  0.00000000
```

INSTRUCTIONS

Possible Answers (Correct Answer is **Bolded**)

Semaan Gaby Eid

Mohamed Chaoui

Abdeluahid Berrak

Jamal Zougam

Chapter 2: In its weakness lies its strength

In this chapter we will spot the most influential ties among terrorists in the network. We will use a centrality measure on ties, called betweenness, and will learn how to visualize the network highlighting connections with high betweenness centrality. Moreover, we will provide some alternative evidence regarding Mark Granovetter's theory of strength of weak ties, confirming that looser connections are crucial as demonstrated in the Madrid terrorism network.

Betweenness of ties

Betweenness of ties is defined by the number of shortest paths going through a tie.

Ties with high betweenness may have considerable influence within a network by virtue of their control over information passing between nodes. They are also the ones whose removal will most disrupt communication between nodes.

We will compute a weighted version of betweenness, with tie weights inversely proportional to tie strength.

The network `g` and the data frame `ties` are at your disposal.

INSTRUCTIONS

Put the inverse of the tie weights in a variable called `dist_weight`.

Compute the weighted tie betweenness with `edge_betweenness()` and save it to `btw`.

Mutate the data frame `ties`, add the variable `betweenness`, and set it to `btw`.

Add the tie attribute `betweenness` to the network `g`.

```r
# save the inverse of tie weights as dist_weight
dist_weight <- 1 / E(g)$weight

# compute weighted tie betweenness
btw <- edge_betweenness(g, weights = dist_weight)

# mutate the data frame ties adding a variable betweenness using btw
ties <- mutate(ties, betweenness = btw)

# add the tie attribute betweenness to the network
E(g)$betweenness <- btw
```

Find ties with high betweenness

In the tidy approach to network science, a network is represented with a pair of data frames: one for nodes and one for ties.

In this exercise, we will exploit the `dplyr` function `left_join()` to extract information from both the `nodes` and `ties` data frames. We need to use a join because the `ties` data frame contains the IDs of the terrorists, not their names, which are stored in the nodes data frame.

The data frames `nodes` and `ties` are already loaded in the workspace.

INSTRUCTIONS 1/3

Join the ties with the nodes using left_join (twice) to find the names of terrorists corresponding to the tied nodes.

```r
# join ties with nodes
ties_joined <- ties %>%
  left_join(nodes, c("from" = "id")) %>%
  left_join(nodes, c("to" = "id"))
```

INSTRUCTIONS 2/3

Select only the relevant variables: `ids` and names of tied terrorists and betweenness.

```r
# select only relevant variables and save to ties
ties_selected <- ties_joined %>%
  select(from, to, name_from = name.x, name_to = name.y, betweenness)
```

INSTRUCTIONS 3/3

Finally, arrange the ties in decreasing order of betweenness.

```r
# arrange named ties in decreasing order of betweenness
arrange(ties_selected, -betweenness)
```

```
## # A tibble: 243 x 5
##     from    to name_from          name_to               betweenness
##    <int> <int> <chr>              <chr>                       <dbl>
##  1    37    57 Abdeluahid Berrak  Semaan Gaby Eid              346.
##  2     1    37 Jamal Zougam       Abdeluahid Berrak            292.
##  3     1     7 Jamal Zougam       Imad Eddin Barakat           268.
##  4     1    11 Jamal Zougam       Amer Azizi                   185.
##  5    11    55 Amer Azizi         Mohamed El Egipcio           164.
##  6     1    23 Jamal Zougam       Naima Oulad Akcha            140.
##  7     7    50 Imad Eddin Barakat Taysir Alouny                132.
##  8     1    24 Jamal Zougam       Abdelkarim el Mejjati        108.
##  9    20    57 José Emilio Suárez Semaan Gaby Eid              106.
## 10     1    18 Jamal Zougam       Abderrahim Zbakh             100.
## # ... with 233 more rows
```

Visualize node centrality

In this exercise, you will use the `ggraph` package to visualize the network by making the node size proportional to its centrality (either degree or strength).

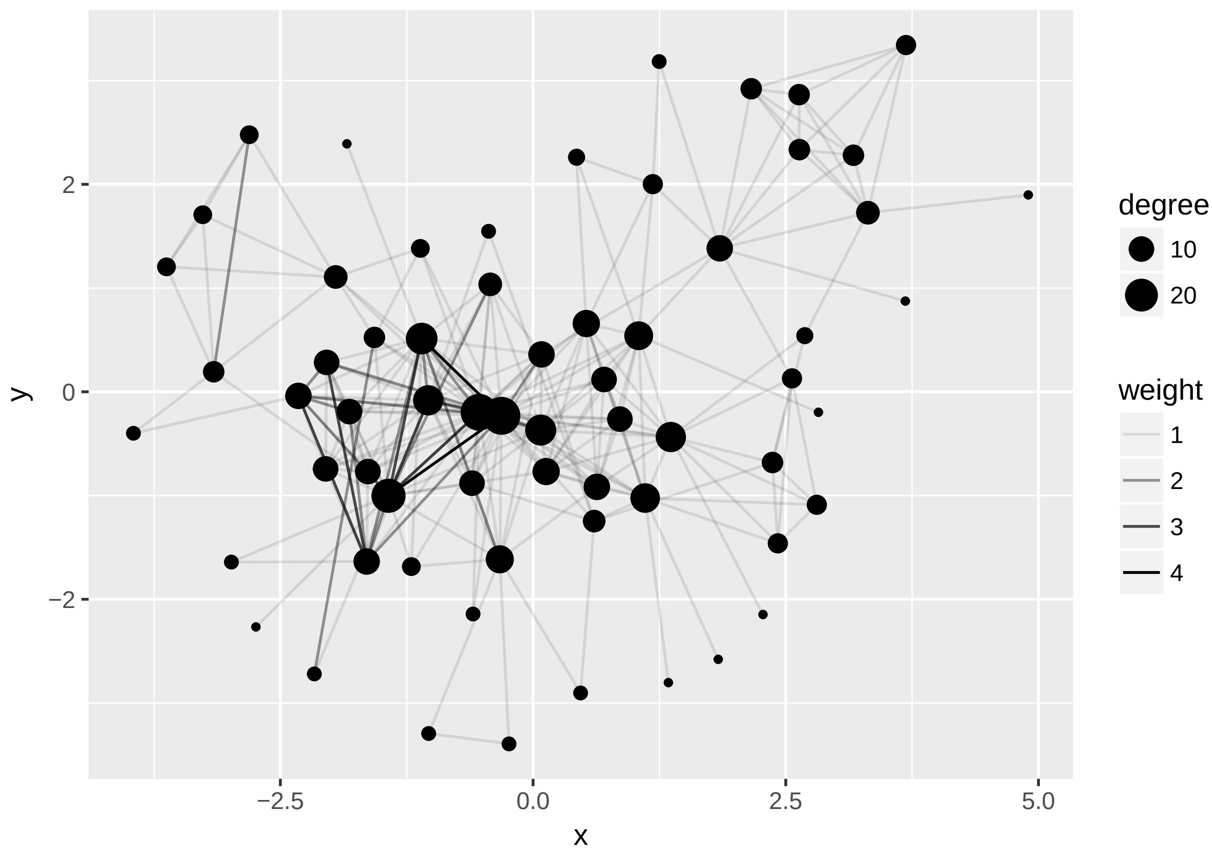
This is useful to visually spot the central nodes in the network. Are these nodes part of the central core or in the periphery?

The network `g` is already loaded in the workspace.

INSTRUCTIONS 1/2

Set network layout to Kamada-Kawai, the tie `alpha` to weight, and node `size` to degree.

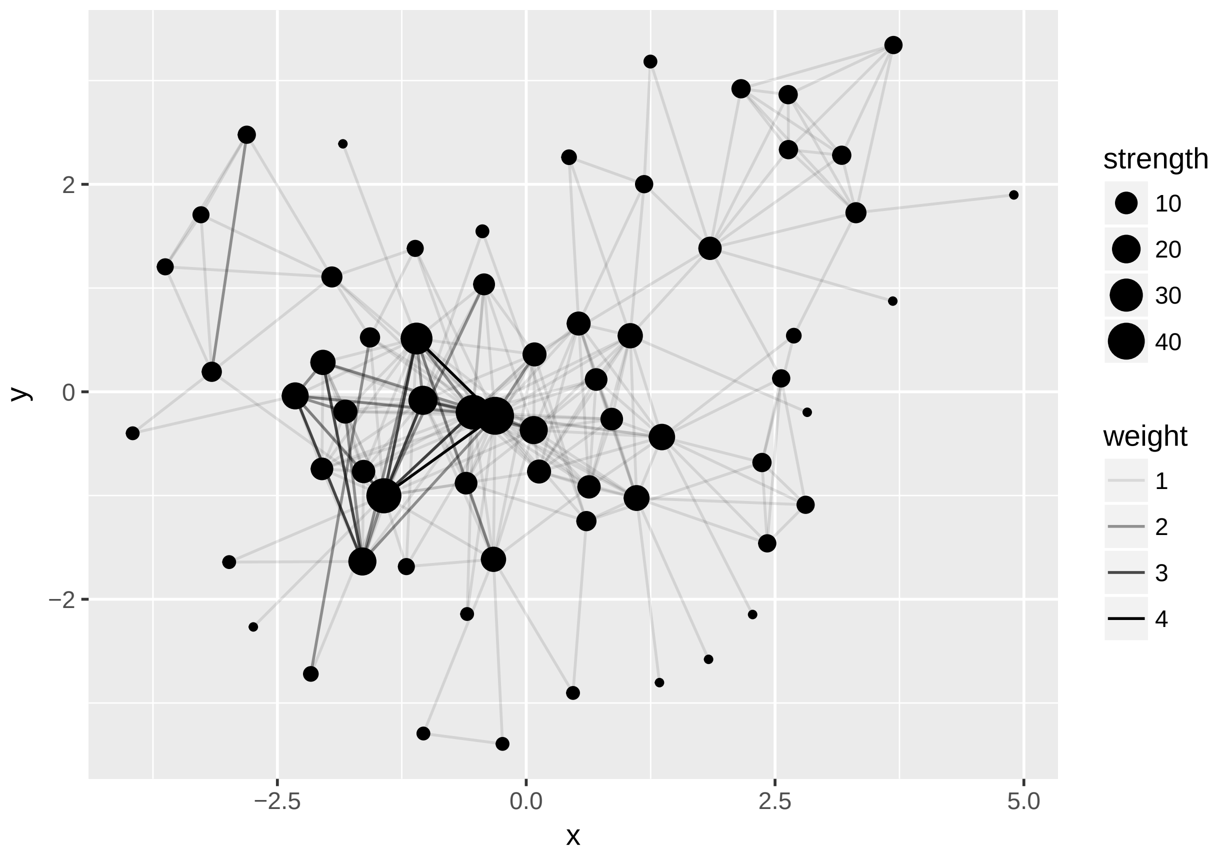
```r
# set (alpha) proportional to weight and node size proportional to degree
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(alpha = weight)) +
  geom_node_point(aes(size = degree))
```



INSTRUCTIONS 2/2

Produce the same visualization, but set node size proportional to strength.

```
# produce the same visualization but set node size proportional to strength
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(alpha = weight)) +
  geom_node_point(aes(size = strength))
```



Visualize tie centrality

In this exercise, you will use the ggraph package again, but this time you will visualize the network by making tie size proportional to tie betweenness centrality.

Can you visually spot the central ties in the network topology? Recall that high betweenness ties typically act as bridges between different communities of the network.

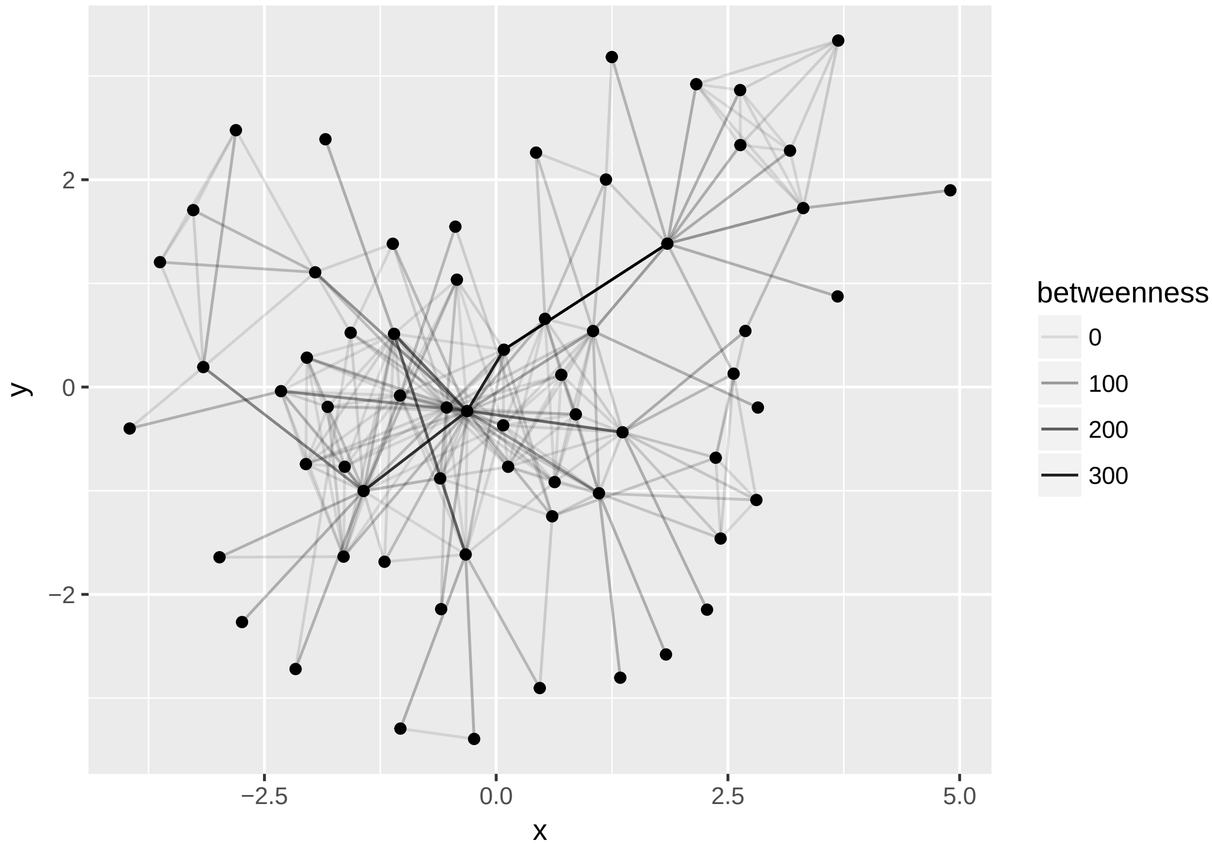
Next, we will add degree centrality to visualize important nodes.

The network `g` is already loaded in the workspace.

INSTRUCTIONS 1/2

Use `ggraph` to visualize the network with the Kamada-Kawai layout (`"with_kk"`). Set the tie transparency using the `alpha` argument proportional to tie betweeness.

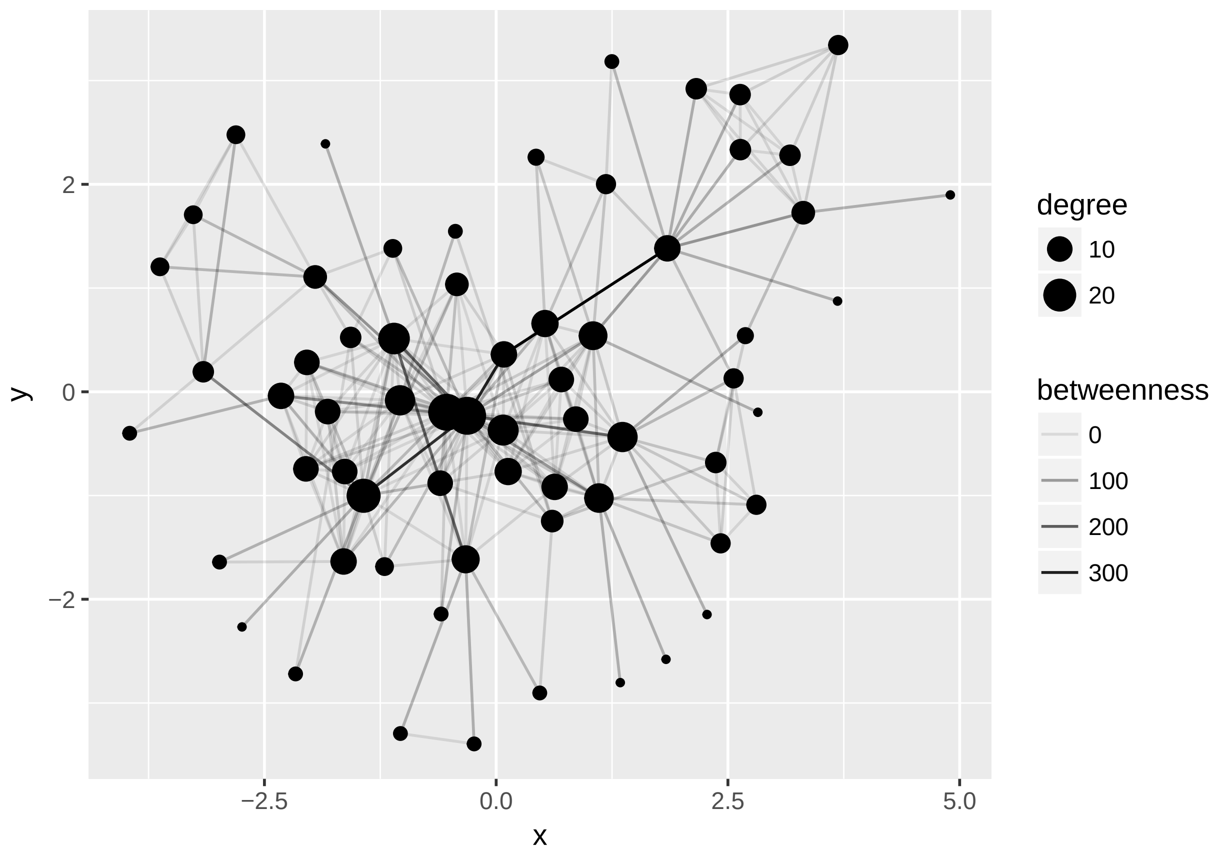
```
# visualize the network with tie transparency proportional to betweenness
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(alpha = betweenness)) +
  geom_node_point()
```



INSTRUCTIONS 2/2

Produce the same visualization with node size proportional to degree.

```
# add node size proportional to degree
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(alpha = betweenness)) +
  geom_node_point(aes(size = degree))
```



Filter important ties

In this exercise, you will use the `ggraph` package once again, but this time we will filter out ties with small betweenness values and only include ties with a large value of betweenness (larger than the median). This will remove half of the ties from the visualization and leave only the important ties.

The network `g` is already loaded in the workspace.

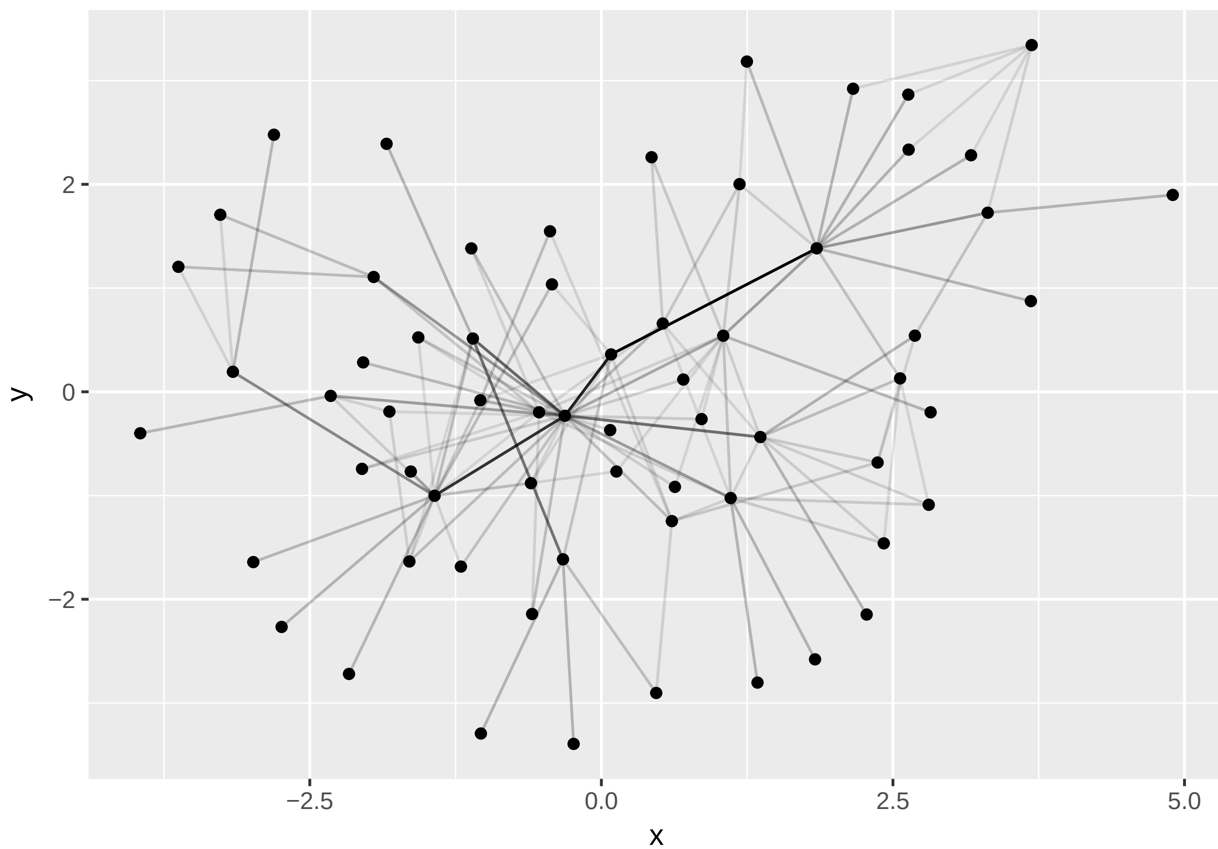
INSTRUCTIONS

Find median betweenness using `median()`.

Use `ggraph` to visualize the network with only ties with betweenness larger than the median.

```r
# find median betweenness
q = median(E(g)$betweenness)

# filter ties with betweenness larger than the median
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(alpha = betweenness, filter = (betweenness > q))) +
  geom_node_point() +
  theme(legend.position="none")
```



How many weak ties are there?

Recall that a weak tie as a tie with a weight equal to 1 (the minimum weight).

In this exercise, we are going to use the `dplyr` function `group_by()` to group ties by their weights and the `summarise()` function to count them. Hence, we are going to discover how many weak ties there are in the network.

The `ties` data frame is loaded in the workspace.

INSTRUCTIONS

Use `group_by()` to group ties by their weight.

Use `summarise()`, `n()`, and `nrow()` to find the total number of ties and calculate the percentage of weak ties.

Finally `arrange()` the groups by decreasing order of the number of ties.

```
# find number and percentage of weak ties
ties %>%
  group_by(weight) %>%
  summarise(number = n(), percentage = number/nrow(ties)) %>%
  arrange(-number)
```

```
## # A tibble: 4 x 3
##   weight number percentage
##    <int>  <int>      <dbl>
## 1      1    214    0.881
## 2      2     21    0.0864
```

```
## 3      3      6     0.0247
## 4      4      2     0.00823
```

Visualize the network highlighting weak ties

In this exercise, we use the `ggraph` package to visualize weak and strong ties in different colors. It is useful to have an immediate visual perception of the importance of weak ties in a network.

The `ties` data frame and the network `g` are already loaded in the workspace for your convenience.

INSTRUCTIONS

Build a Boolean vector named `weakness` that, for each tie, contains `TRUE` if the tie is weak (has weight equal to 1) and `FALSE` otherwise. Take advantage of function `E()` on the weight variable of the network.

Use `sum()` to check that weakness contains the correct number of weak ties (recall that there are 214 weak ties).

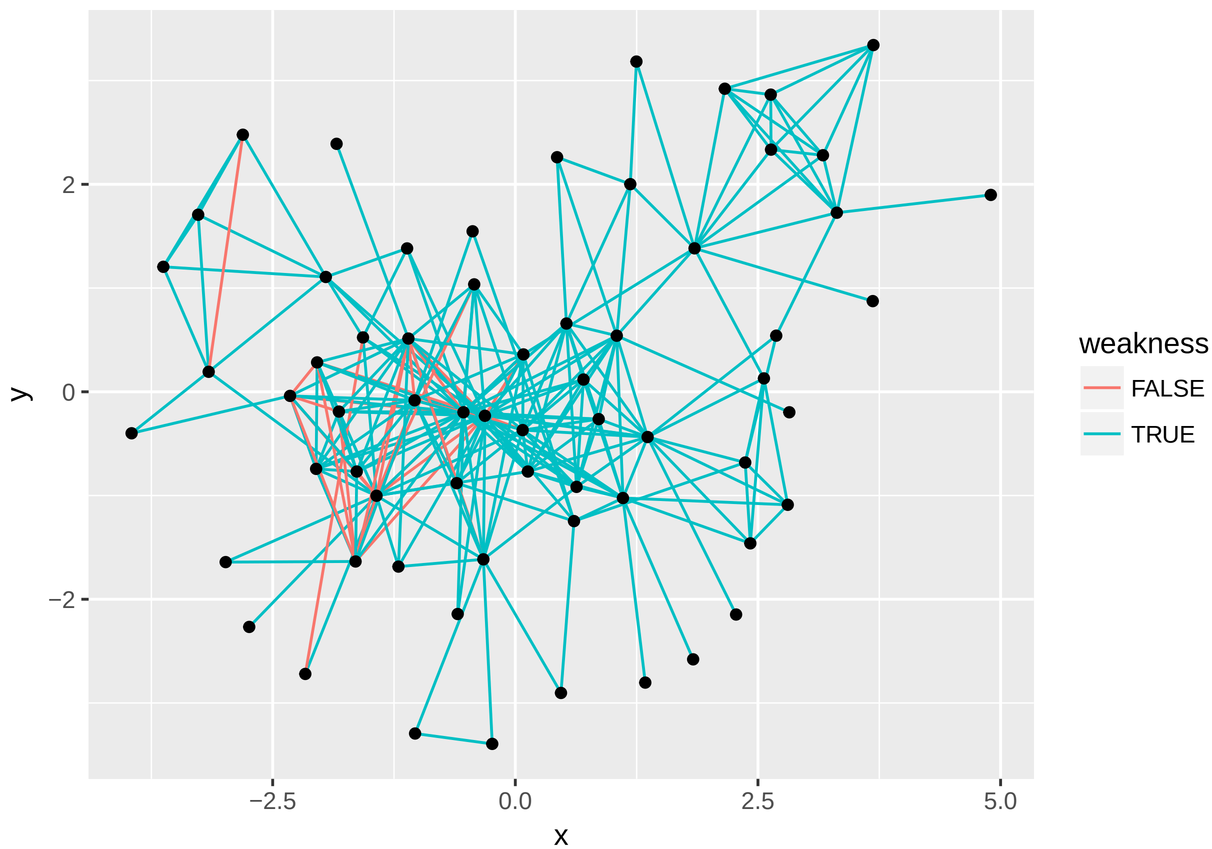
Visualize the network with `ggraph()` by setting the tie `color` to `weakness`.

```r
# build vector weakness containing TRUE for weak ties
weakness <- E(g)$weight <= 1

# check that weakness contains the correct number of weak ties
sum(weakness)
```

```
## [1] 214
```

```r
# visualize the network by coloring the weak and strong ties
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(color = weakness)) +
  geom_node_point()
```



Visualize the sub-network of weak ties

In this exercise, we will use `ggraph` again to visualize the sub-network containing only the weak ties. We will use the aesthetic `filter` to filter the ties.

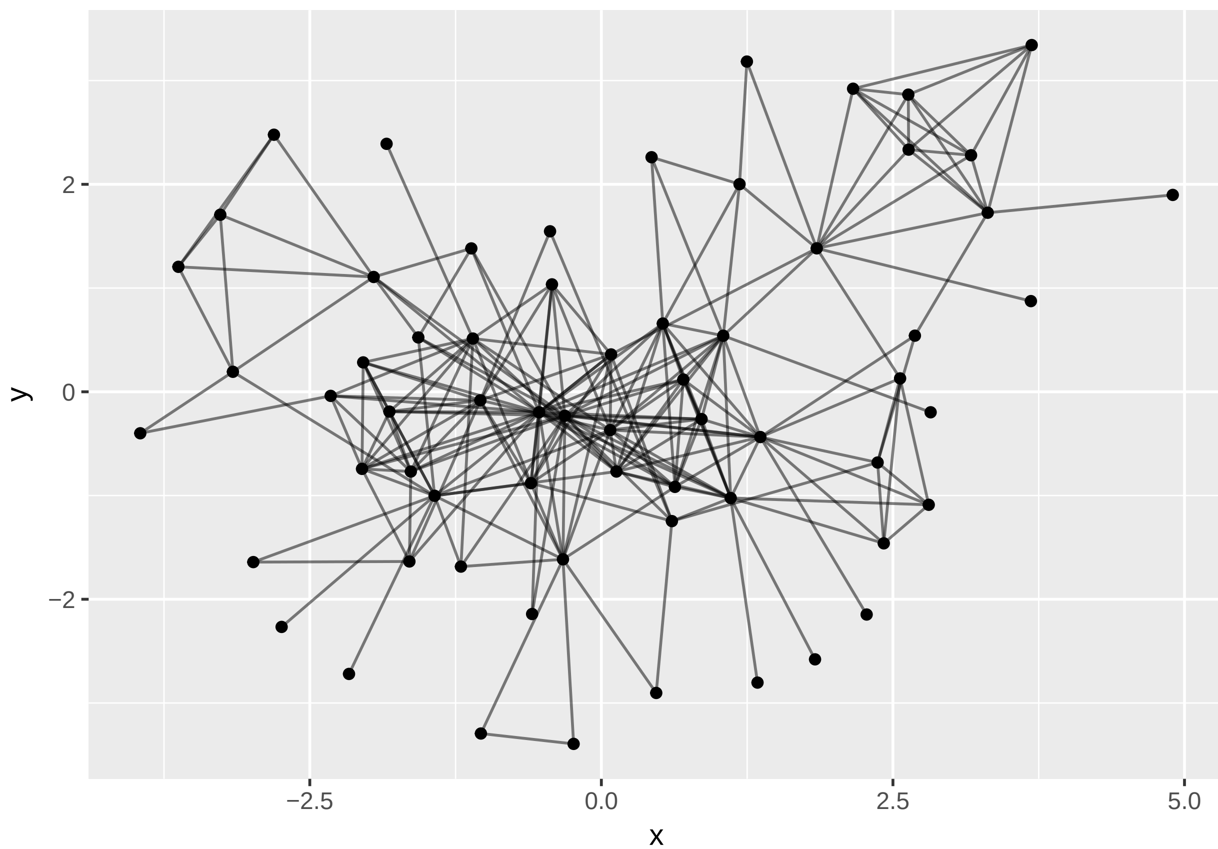
The network `g` and the Boolean vector `weakness` are already loaded in the workspace for your convenience.

INSTRUCTIONS

Visualize the network with only weak ties using the `filter` aesthetic set to the weakness variable.

Set the global transparency, `alpha`, to 0.5 for all ties.

```
# visualize the network with only weak ties using the filter aesthetic
ggraph(g, layout = "with_kk") +
  geom_edge_link(aes(filter = weakness), alpha = 0.5) +
  geom_node_point()
```



More on betweenness

Typically, only the shortest paths are considered in the definition of betweenness. However, there are a couple issues with this approach:

- All paths (even slightly) longer than the shortest ones are not considered.
- The actual number of shortest paths that lie between the two nodes is irrelevant.

In many applications, however, it is reasonable to consider both the quantity and the length of all paths of the network, since communication on the network is enhanced as soon as more routes are possible, particularly if these pathways are short.

Which of the following is not an issue of shortest path betweenness?

ANSWER THE QUESTION

Possible Answers (Correct Answer is **Bolded**)

Only optimal paths are considered.

The quantity of paths between nodes is not considered.

Paths longer than the shortest path are irrelevant.

The computational complexity is prohibitive.

Chapter 3: Connection Patterns

The challenge in this chapter is to discover pairs of similar (and dissimilar) terrorists. We will introduce the adjacency matrix as a mathematical representation of a network and use it to find terrorists with similar connection patterns. We will also learn how to visualize similar and dissimilar pairs of individuals using ggraph.

Chapter 4: Similarity clusters

In this chapter we will discover cells of similar terrorists. We will explore hierarchical clustering to find groups of similar terrorists building on the notion of similarity of connection patterns developed in the previous chapter. Furthermore, we will explore the visNetwork package to produce fulfilling interactive network visualizations.