Assignment 1

2024-03-15

First let's load the dataset and packages:

```
connections <- read.csv("/cloud/project/connections.csv")
attach(connections)

install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
install.packages("tidygraph")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
library(tidygraph)
```

```
##
## Attaching package: 'tidygraph'
##
## The following object is masked from 'package:stats':
##
##     filter
```

```
install.packages("igraph")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following object is masked from 'package:tidygraph':
##
```

```
##     groups
##
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
##
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
## The following object is masked from 'package:tidyr':
##
##     crossing
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

We have to count the number of contacts that work at or are associated to a current employer as well as the total number overall:

```
number_of_connections = connections %>%
  group_by(Company) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

total_num = nrow(connections)

number_of_connections
```

```
## # A tibble: 399 x 2
##    Company                                                 count
##    <chr>                                                   <int>
##  1 "McGill University - Desautels Faculty of Management"      24
##  2 "McGill University"                                        16
##  3 "Deloitte"                                                 12
##  4 ""                                                         10
##  5 "L'Oreal"                                                   7
##  6 "Bell"                                                      6
##  7 "CN"                                                        6
##  8 "Desjardins"                                                6
##  9 "TD"                                                        6
## 10 "BDC"                                                       5
## # i 389 more rows
```

```
total_num
```

```
## [1] 583
```

```
# I have 583 connections in total
```

We'll install ggraph to plot later on:

```
install.packages("ggraph")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
library(ggraph)
```

We remove our contacts' full last names and replace them with only the first letter:

```
# Create a unique identifier for each contact
connections <- connections %>%
  mutate(contact_id = row_number(),
         label = paste(connections$First, substr(connections$Last, 1, 1), sep = " "))
```

We create our nodes for each contact and specify which companies I'm affiliated to:

```
# Create nodes dataframe
nodes <- connections %>%
  select(contact_id, label, Company) %>%
  distinct()

# Create a dataframe to map companies to specific groups for adding edges to specified companies
specified_companies <- c("McGill University - Desautels Faculty of Management",
                         "McGill University", "MetaMusique",
                         "Elite Transport Solutions", "Elite Transport Solutions inc")
```

We create the edges based on the contacts who are part of a company I am or was associated to:

```
# Create edges based on company affiliation
edges <- connections %>%
  select(contact_id, Company) %>%
  filter(Company %in% specified_companies) %>%
  distinct() %>%
  # Dummy node for me with id 0 & connect to contacts from companies
  mutate(from = 0, to = contact_id) %>%
  select(from, to)

# Add a dummy node for yourself to the nodes dataframe
my_node <- data.frame(contact_id = 0, label = "Me", Company = NA)
nodes <- bind_rows(my_node, nodes)
```

We create internal edges and combine everything to set up the graph:

```
install.packages("purrr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
library(purrr)

# Create internal edges
```

```r
internal_edges <- connections %>%
  select(contact_id, Company) %>%
  distinct() %>%
  group_by(Company) %>%
  filter(n() > 1 & !Company %in% specified_companies) %>%
  summarise(contact_ids = list(contact_id), .groups = 'drop') %>%
  mutate(pairs = map(contact_ids, ~combn(.x, 2, simplify = FALSE))) %>%
  unnest(pairs) %>%
  mutate(from = map_chr(pairs, ~as.character(.x[1])),
         to = map_chr(pairs, ~as.character(.x[2]))) %>%
  select(from, to)

# Convert 'from' and 'to' columns to numeric as they were generated as characters
internal_edges <- internal_edges %>%
  mutate(from = as.numeric(from), to = as.numeric(to))

# Combine the specified company edges with internal company edges
edges <- bind_rows(edges, internal_edges)

# Now, all contacts referenced in edges should exist in nodes
# Recreate the graph with corrected data
network <- graph_from_data_frame(d = edges, vertices = nodes, directed = FALSE)
```

We plot the graph:

```r
# Visualize the network with ggraph (the optional step)
ggraph(network, layout = 'fr') +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = label), repel = TRUE) +
  theme_graph()
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Jingwen (Hennieo<bc><89> H' in 'mbcsToSbcs': dot
## substituted for <bc>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Jingwen (Hennieo<bc><89> H' in 'mbcsToSbcs': dot
## substituted for <89>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Jingwen (Hennieo<bc><89> H' in 'mbcsToSbcs': dot
## substituted for <bc>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Jingwen (Hennieo<bc><89> H' in 'mbcsToSbcs': dot
## substituted for <89>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Jingwen (Hennieo<bc><89> H' in 'mbcsToSbcs': dot
## substituted for <bc>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Jingwen (Hennieo<bc><89> H' in 'mbcsToSbcs': dot
## substituted for <89>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
```