Assignment 1

Emery Dittmer 2023-03-09

LinkedIn Contact Analysis

This is an analysis of my linkedin network. The data data was extracted from linked in and reflects the connections with the companies that they currently work for. We will set up the data in visualizations.

We ideally want to set up the data in such a way that we are connected to a company (edge) which is connected to all the people we know at that compnay (node).

Pre-Processing

Library & Data

Before getting started on visualizing the network lets set up the libraries and data we need.

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidygraph)
```

```
##
## Attaching package: 'tidygraph'

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

```
library(reticulate)
library(stringr)
library(ggplot2)
library(ggraph)
library(tidyverse)
```

```
## ── Attaching packages
## ───────────────────────────────────────
## tidyverse 1.3.2 ──
## ✓ tibble  3.1.8     ✓ purrr   0.3.4
## ✓ tidyr   1.2.1     ✓ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ tidygraph::filter() masks dplyr::filter(), stats::filter()
## ✖ dplyr::lag()        masks stats::lag()
```

```
library(ggrepel)
use_condaenv("DataSci")
```

Let’s import the data.

```
#Import as Text
my_data <- read.delim('Connections.csv',sep =",", header = TRUE,skip=3)
#Import as CSV
connections=read.csv('Connections.csv',skip=3)

#limit data for simplicity
#connections=connections[1:50,]
```

Data pre-processing

we will now reshape the data so that we can get the names of companies, the company counts and add ourselvs to the network.

```
connections=drop_na(connections)
connections <- subset(connections, Company != "")

#this datframe shows all the connections with a total!
Connections_Detail_with_total <- connections %>%
  filter(!is.na(Company)) %>%
  #not all blanks were caught by the filter. Requires more robust approach
  filter(length(Company)>2) %>%
  group_by(Company) %>%
  summarise(weight=n()) %>%
  bind_rows(summarise(., across(where(is.numeric), sum),
                      across(where(is.character), ~'Total')))

Connections_Detail <- connections %>%
  filter(!is.na(Company)) %>%
  #not all blanks were caught by the filter. Requires more robust approach
  filter(length(Company)>2) %>%
  group_by(Company) %>%
  summarise(weight=n())

#get the list of names and how the connections fit
Connections_Detail=merge(Connections_Detail,connections,by='Company',all=TRUE)

#add First and last
Connections_Detail <- Connections_Detail %>%
  mutate(peoplenames = paste(First.Name,substring(Last.Name,1,1)))

#set column order
col_order=c("peoplenames","weight","Company","First.Name","Last.Name","Email.Address","Position","Connected.On")

#reorder columns
Connections_Detail <- Connections_Detail[, col_order]

#add to and from fields
Connections_Detail <- Connections_Detail %>%
  mutate(from = Company,
         to = peoplenames) %>%
  select(from, to, weight)
```

```
#get company names for later
companies <- unique(Connections_Detail$from)

#Add yourself to the network
##create a dataframe of yourself
You_network=data.frame(from=rep("you",length(unique(Connections_Detail$from))),to=unique(Connections_Detail$from),wei

#adding to the concctiond
Connections_Detail=rbind(Connections_Detail,You_network)
```

##Graph Table ### Set data into graph table In order to visualize the network we need to set the data into a special type of table. The graph table determines relationships between nodes. These can be to and from relationships in directed netwroks.

```
graph_connections <- as_tbl_graph(Connections_Detail)
```

We will need to modify the table and add a title field to make sure that it is accepted by the code that vislizes the data.

```
graph_connections <- graph_connections %>%
  activate(nodes) %>%
  mutate(
    title = str_to_title(name),
    label = str_replace_all(title, " ", "\n")
    )

graph_connections
```

```
## # A tbl_graph: 674 nodes and 680 edges
## #
## # A directed acyclic simple graph with 1 component
## #
## # Node Data: 674 × 3 (active)
##   name                title               label
##   <chr>               <chr>               <chr>
## 1 " -"                " -"                "\n-"
## 2 "ABB"               "Abb"               "Abb"
## 3 "Abbott"            "Abbott"            "Abbott"
## 4 "AbbVie"            "Abbvie"            "Abbvie"
## 5 "Aberdeen Advisors" "Aberdeen Advisors" "Aberdeen\nAdvisors"
## 6 "Absorb Software"   "Absorb Software"   "Absorb\nSoftware"
## # … with 668 more rows
## #
## # Edge Data: 680 × 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1   259      1
## 2     2   260      1
## 3     3   261      1
## # … with 677 more rows
```

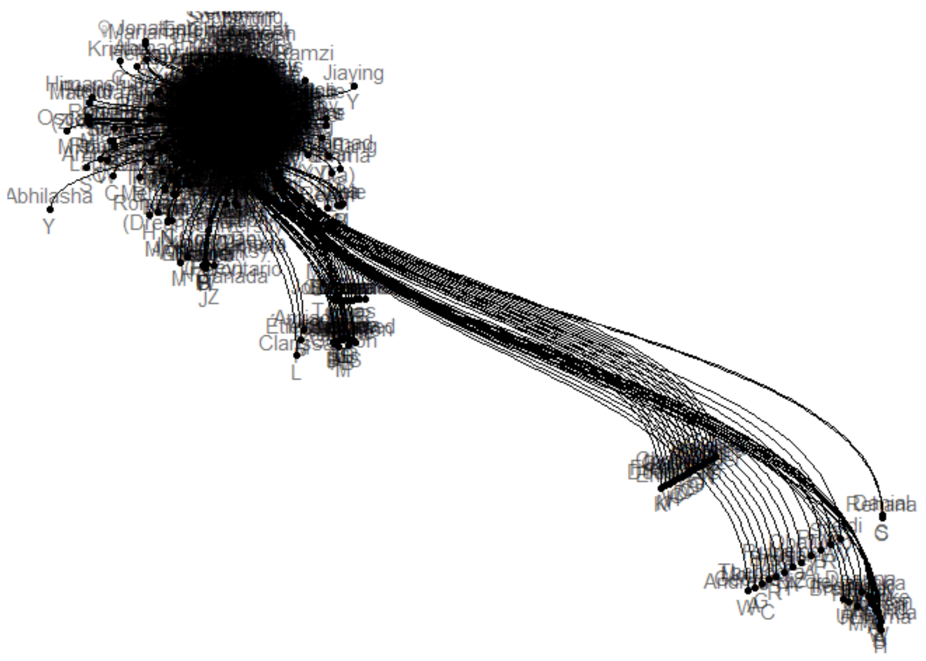
Plotting

Using GGraph

First we will use ggraph to map some of the connections

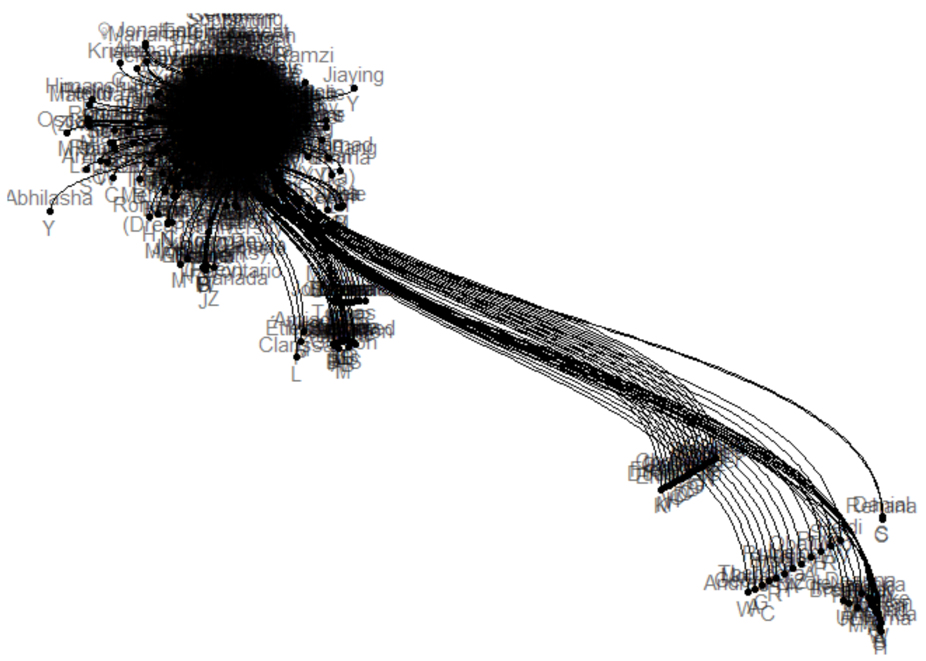
```
thm <- theme_minimal() +
  theme(
    legend.position = "none",
     axis.title = element_blank(),
     axis.text = element_blank(),
      panel.grid = element_blank(),
      panel.grid.major = element_blank(),
    )

theme_set(thm)
graph_connections %>%
  ggraph(layout = "kk") +
    geom_node_point() +
    geom_edge_diagonal() +
  geom_node_text(aes(label = label, alpha=0.1))
```

```
## Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` in the `default_aes` field and elsewhere instead.
```



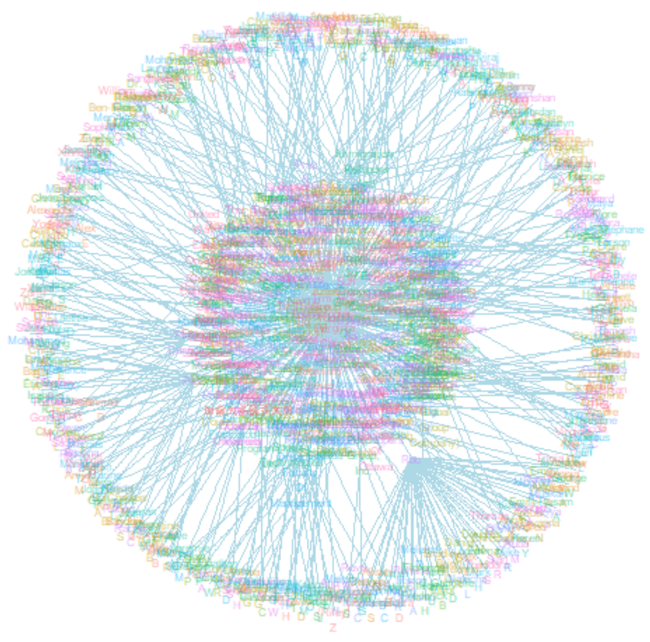
The cloud is complex, let's try some other methods to clear it up.

```
theme_set(thm)
graph_connections %>%
  ggraph(layout = "kk") +
    geom_node_point() +
    geom_edge_diagonal() +
  geom_node_text(aes(label = label, alpha=0.1))
```

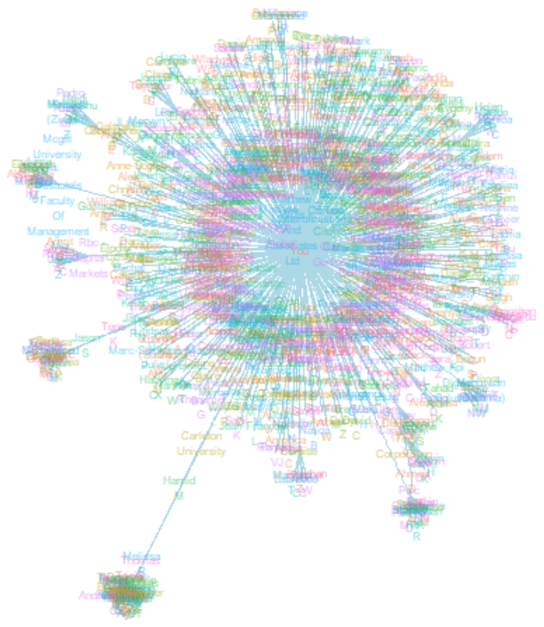


```
lapply(c('stress', 'fr', 'lgl', 'graphopt','kk'), function(layout) {
  graph_connections %>%
    ggraph(layout = layout) +
    geom_edge_fan(width = .2, color = 'lightblue') +
    geom_node_text(aes(label = label, color = name,alpha=0.1), size = 2) +
    coord_fixed()+
    scale_fill_brewer()
})
```

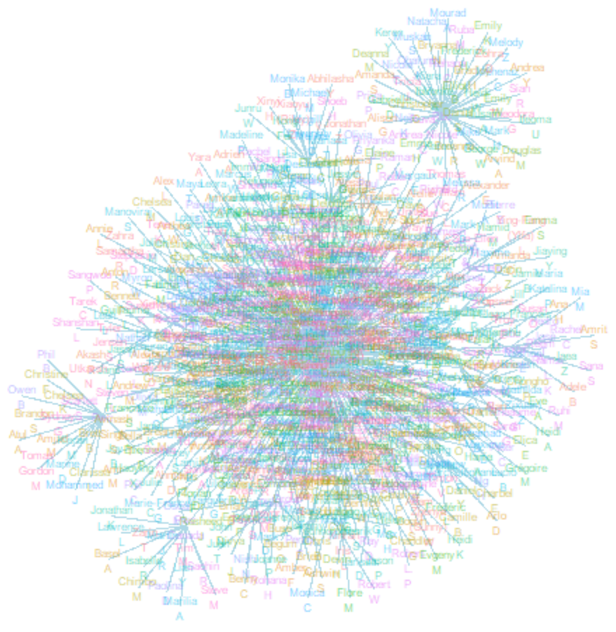
```
## [[1]]
```



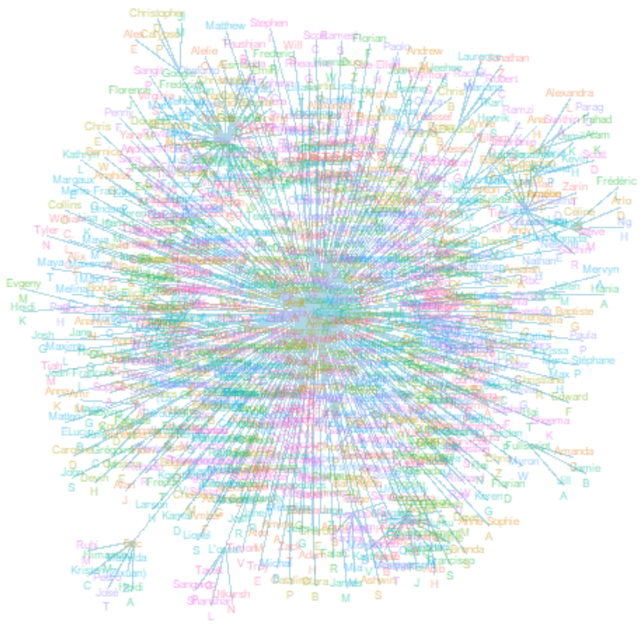
```
##
## [[2]]
```



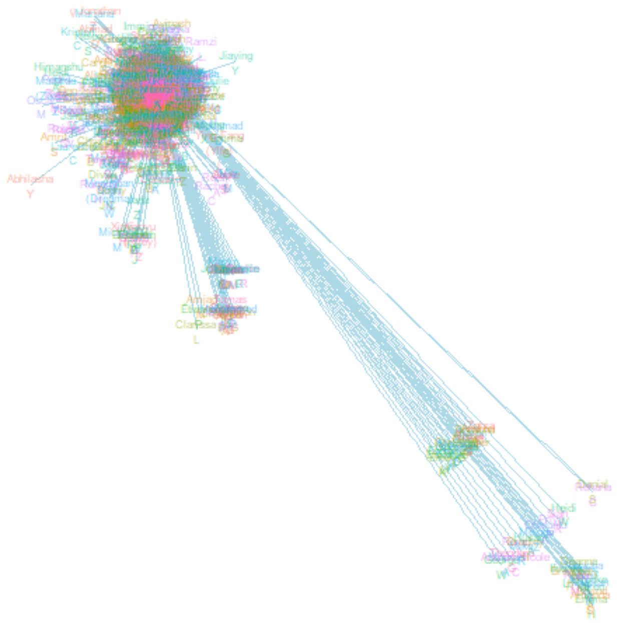
```
##
## [[3]]
```



```
##
## [[4]]
```

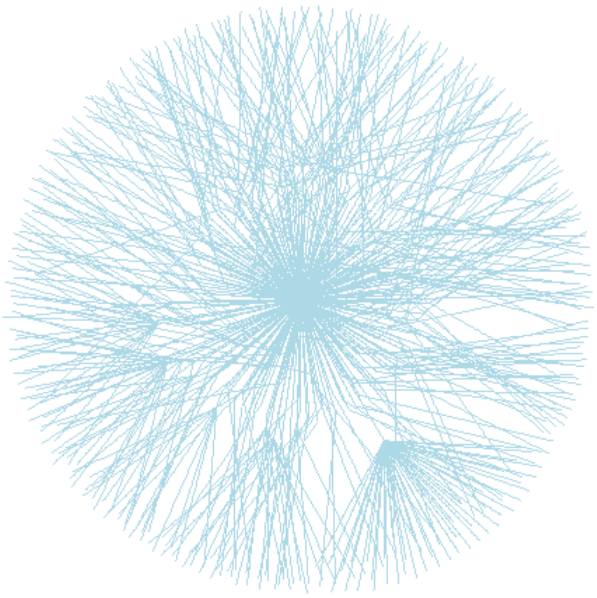


```
##
## [[5]]
```

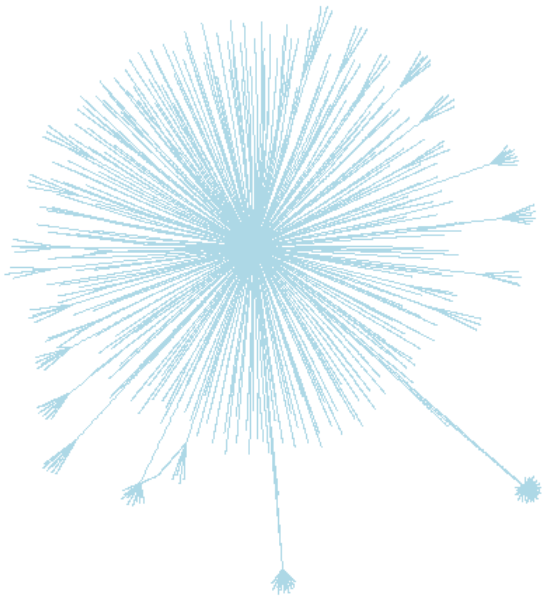


```
lapply(c('stress', 'fr', 'lgl', 'graphopt','kk'), function(layout) {
  graph_connections %>%
    ggraph(layout = layout) +
    geom_edge_fan(width = .2, color = 'lightblue') +
    coord_fixed()+
    scale_fill_brewer()
})
```

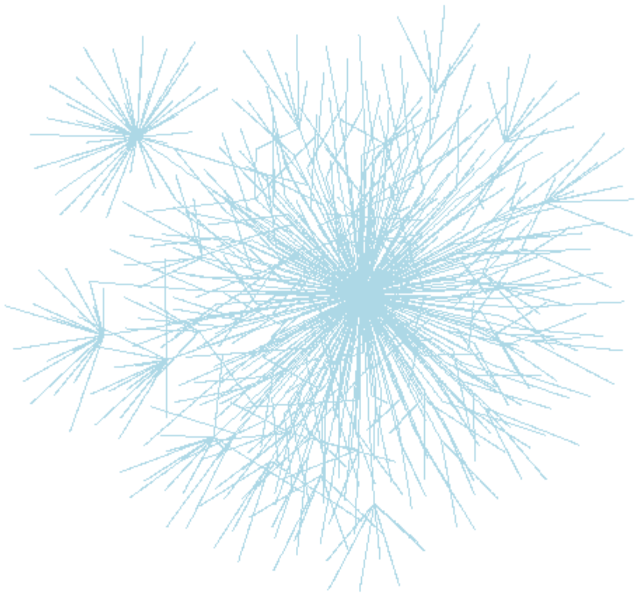
```
## [[1]]
```



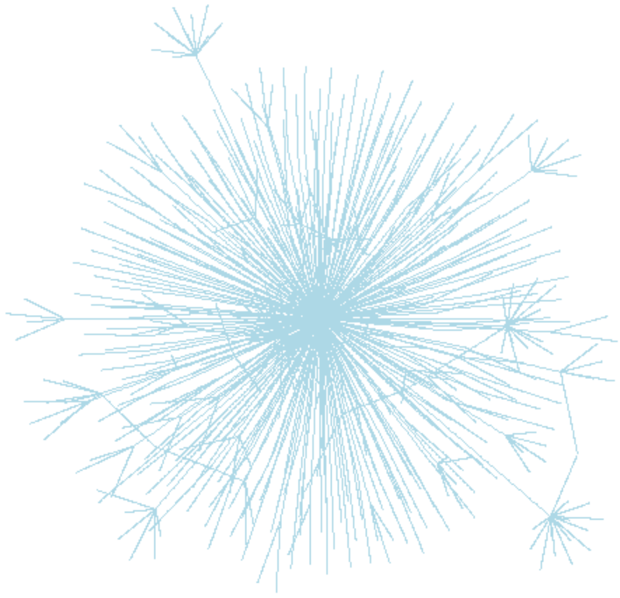
```
##
## [[2]]
```



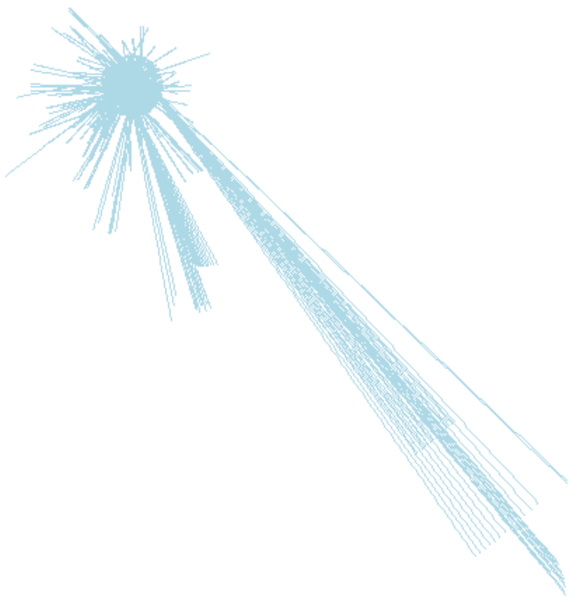
```
##
## [[3]]
```



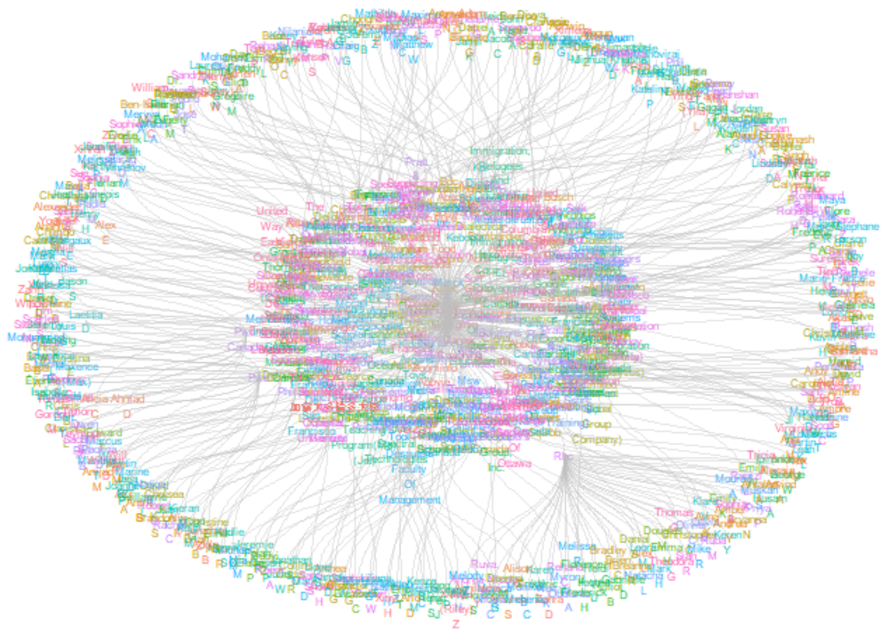
```
##
## [[4]]
```



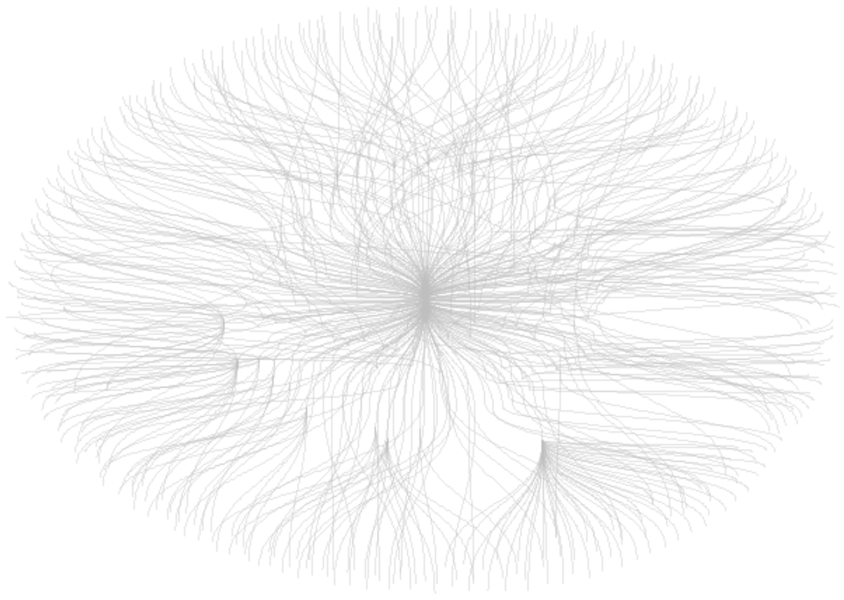
```
##
## [[5]]
```



```r
graph_connections %>%
  ggraph(layout = "stress") +
    geom_node_text(aes(label = label, color = name), size = 2) +
    geom_edge_diagonal(color = "gray", alpha = 0.4)
```

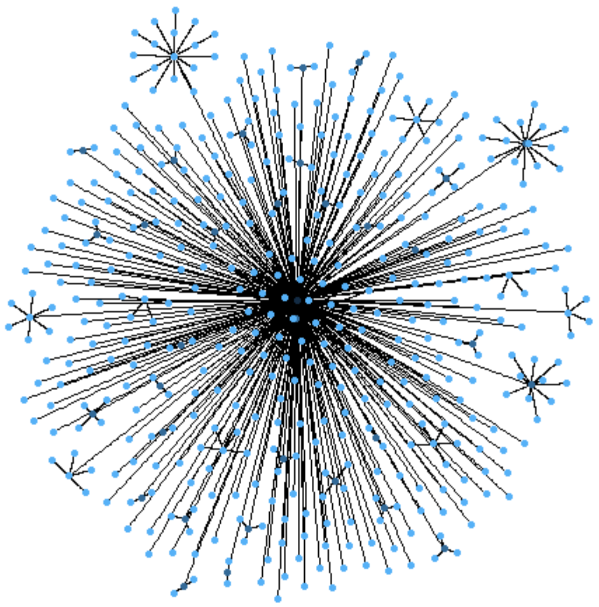


```r
graph_connections %>%
  ggraph(layout = "stress") +
    geom_edge_diagonal(color = "gray", alpha = 0.4)
```



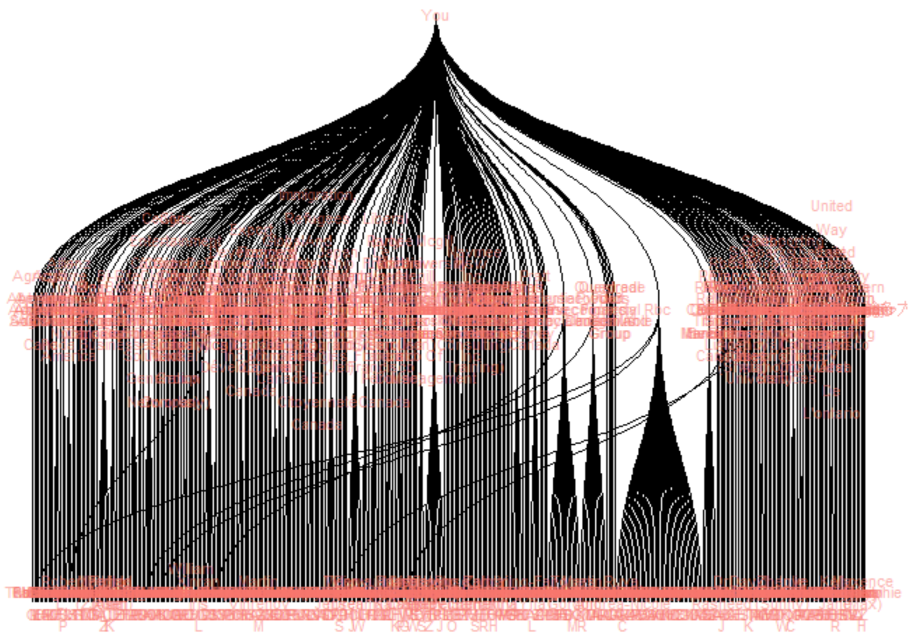
```
ggraph(graph_connections, 'circlepack') +
  geom_edge_link() +
  geom_node_point(aes(colour = depth)) +
  coord_fixed()
```

```
## Multiple parents. Unfolding graph
```



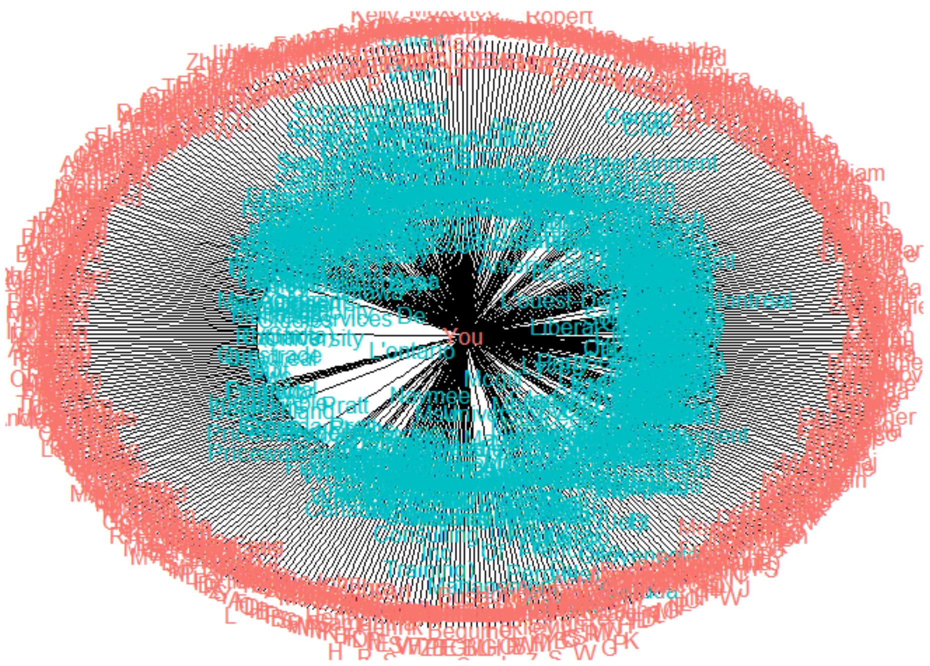
We can also use a hierarchical or tree typ view

```
ggraph(graph_connections, 'tree') +
  geom_edge_diagonal()+
  geom_node_text(aes(label = label,alpha=0.2, color='blue'),size=3)
```



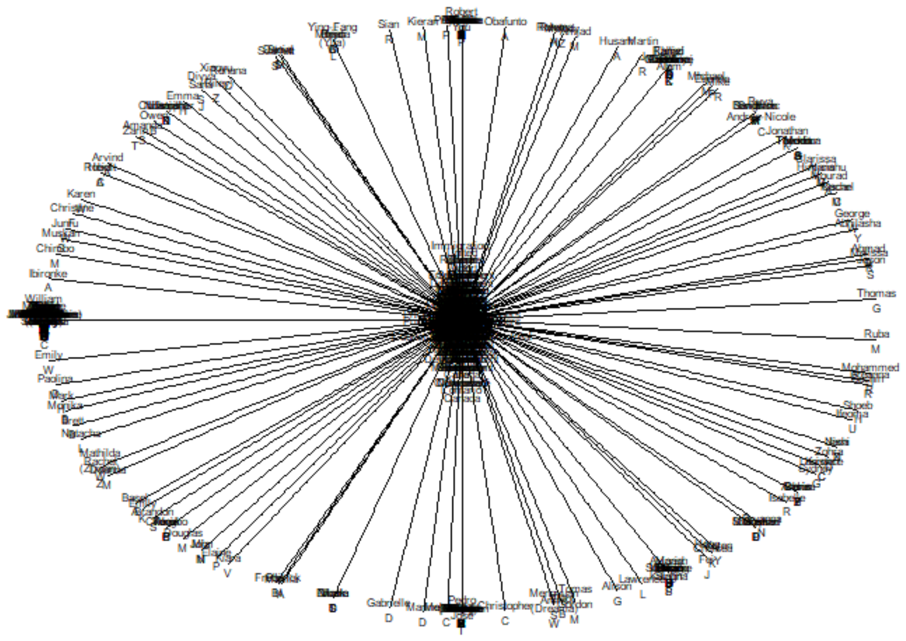
```
ggraph(graph_connections, 'dendrogram', circular = TRUE ) +
  geom_edge_elbow() +
  #coord_fixed()+
  #geom_node_text(aes(label = label , color = name), size = 2)+
  geom_node_text(aes(label = label , color = ifelse(name %in% companies, "red","black"), size = 2))
```

```
## Multiple parents. Unfolding graph
```



Attempted a branching or unrooted tree but it was not effective

```
ggraph(graph_connections, 'unrooted') +
  geom_edge_link()+
  geom_node_text(aes(label = label), size = 2)
```



Tidy Graph

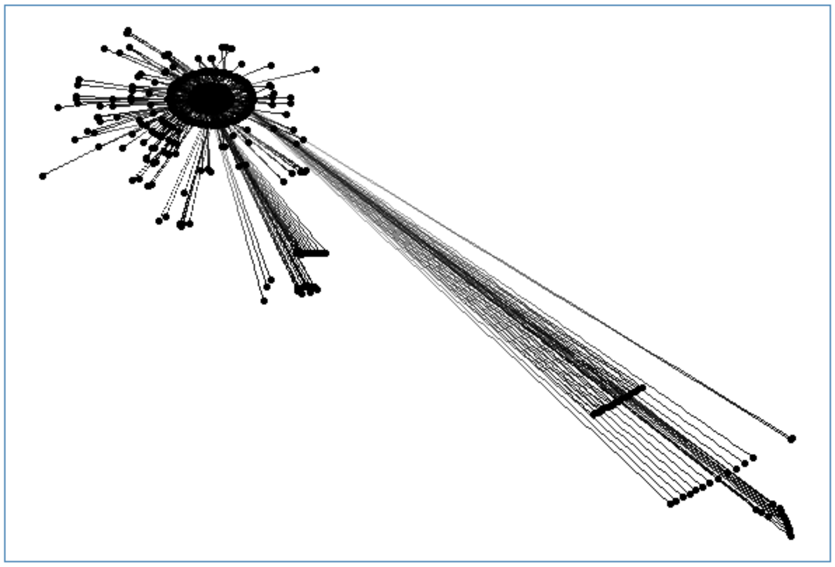
We will now use a package called tidy graphy to visualize the netwrok. **Credit to:http://users.dimi.uniud.it/~massimo.franceschet/ns/syllabus/make/tidygraph/tidygraph.html**

```
# graph analysis and visualziation
library(tidygraph)
library(ggraph)
library(igraph)
```

```
##
## Attaching package: 'igraph'

## 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 object is masked from 'package:tidygraph':
##
##     groups

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
# tidy data analysis and visualziation
library(readr)
library(dplyr)
```

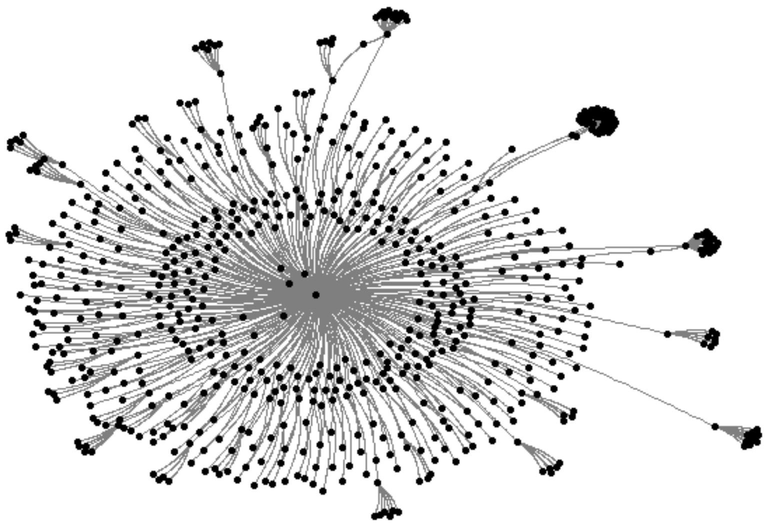
```
graph <- as_tbl_graph(Connections_Detail)

# plot using ggraph
ggraph(graph, layout = 'kk') +
    geom_edge_fan(aes(alpha = after_stat(index)), show.legend = FALSE) +
    geom_node_point() +
    theme_graph(foreground = 'steelblue', fg_text_colour = 'white')
```



Visualizing the data is tricky but we can do some data manipulations to show more clearly the network.

```
# edge size shows frequency of co-occurrence
graph %>%
   ggraph(layout = "fr") +
   geom_edge_arc(colour= "gray50",
                 lineend = "round",
                strength = .1) +
   geom_node_point() +
   geom_node_text(aes(label = name),
                 repel = TRUE,
                 point.padding = unit(0.2, "lines"),
                 colour="gray10") +
  scale_edge_width(range = c(0, 2.5)) +
  scale_edge_alpha(range = c(0, .3)) +
  theme_graph(background = "white") +
  guides(edge_width = FALSE,
         edge_alpha = FALSE)
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.

## Warning: ggrepel: 674 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



Unforuntely the sizing functions are not currently working but would help determine the strength of network ties.

Vsiualization II

Visulize the results in python’s plotly

Let’s try some python pacakges to enhance what we have

```
import pandas as pd
import numpy as np
from pyvis.network import Network
import plotly.express as px
import networkx as nx
import matplotlib.pyplot as plt
df = pd.read_csv('Connections.csv',skiprows=3)
df['Full Name']= df.apply(
    lambda x: str(x['First Name']) +" "+ str(x['Last Name']) ,
    axis=1)
    #lambda x: str(x['First Name']) +" "+ str(x['Last Name'])[0] , axis=1)
df=df.dropna(subset=['Company', 'Position'])
df=df.groupby("Company").filter(lambda x: len(x) > 1)

df['Count_Company'] = df.groupby('Company')['Company'].transform('size')
df['Count_Company']=df['Count_Company'].astype(int)

df['Position_Count'] = df.groupby('Position')['Position'].transform('size')
df['Position_Count']=df['Position_Count'].astype(int)



net = Network(height="750px", width="100%", bgcolor="#222222", font_color="white",notebook=True)

# set the physics layout of the network



## Local cdn resources have problems on chrome/safari when used in jupyter-notebook.



net.barnes_hut()
```

```
sources = df['Company']
targets = df['Full Name']
weights = df['Count_Company']

edge_data = zip(sources, targets, weights)

for e in edge_data:
                src = e[0]
                dst = e[1]
                w = e[2]
                net.add_node(src, src, title=src)
                net.add_node(dst, dst, title=dst)
                net.add_edge(src, dst, value=w)

neighbor_map = net.get_adj_list()

# add neighbor data to node hover data
for node in net.nodes:
                node["title"] += " Neighbors:<br>" + "<br>".join(neighbor_map[node["id"]])
                node["value"] = len(neighbor_map[node["id"]])

net.show("network.html")
```

```
## <IPython.lib.display.IFrame object at 0x0000015DA80F32E0>
```