Assignment 1

Setup

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()
```

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

```
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","Connecte

#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_Detai

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

#Visulization ## Plot as a network unsing ggraph First we need a few transformations. Now set the network as a graph table for simplicity.

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

We will need to modify the table slightly to make sure that it is accepted by the code.

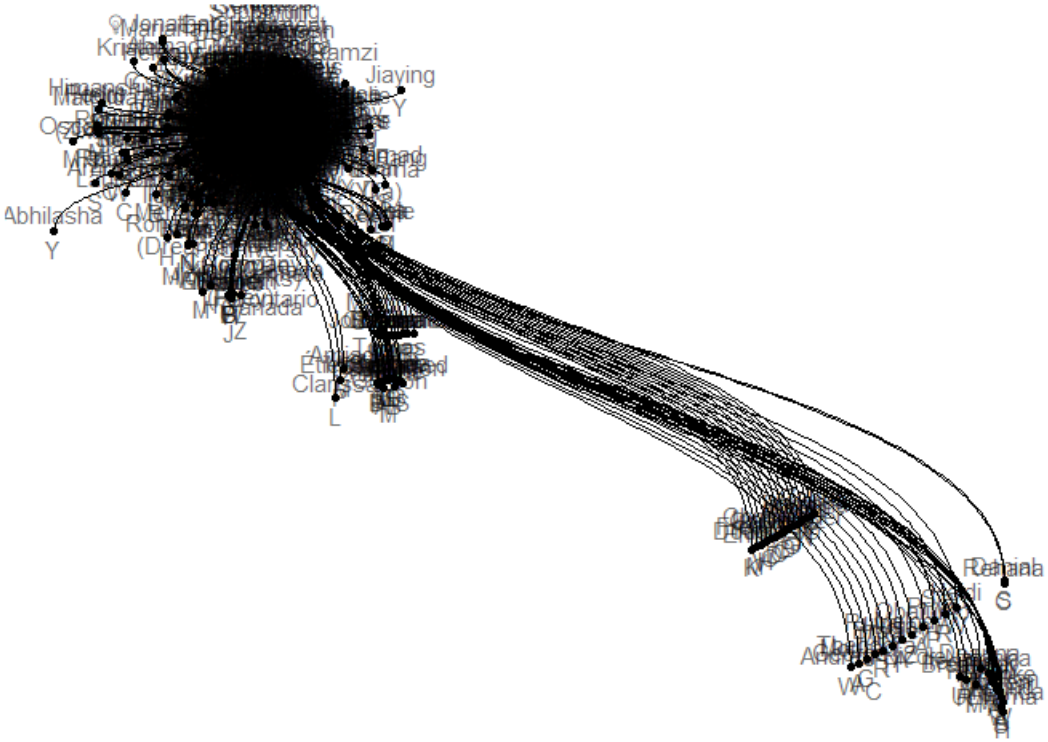
```
graph_connections <- graph_connections %>%
  activate(nodes) %>%
  mutate(
    title = str_to_title(name),
    label = str_replace_all(title, " ", "\n")
    )

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
```

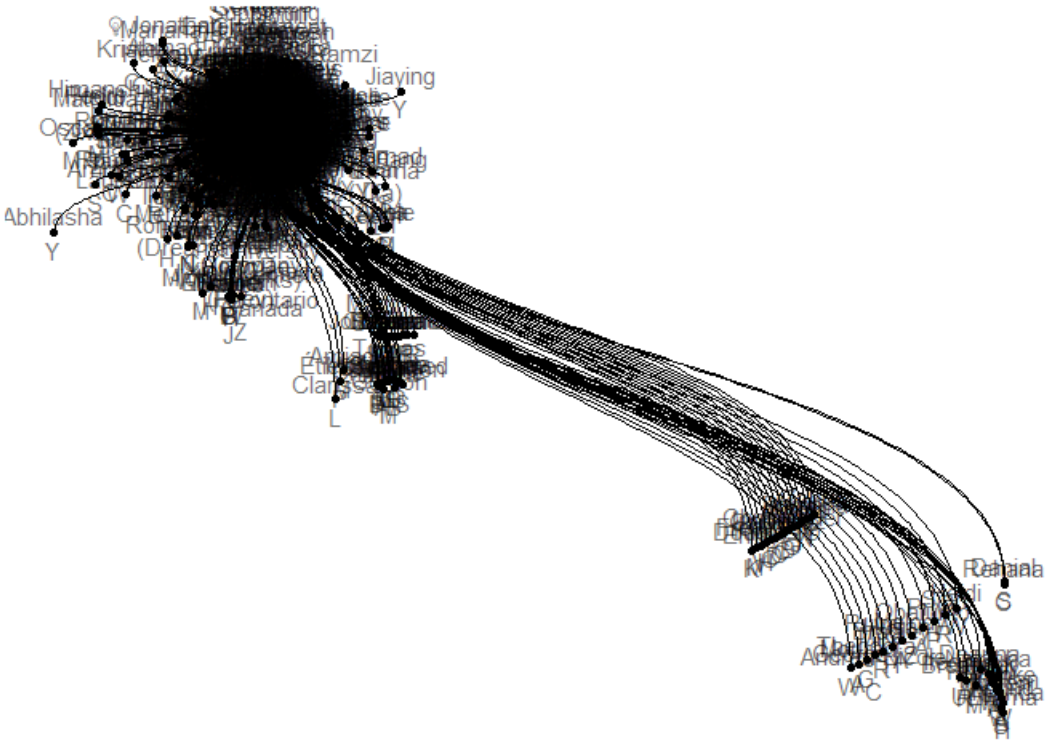
```
thm <- theme_minimal() +
  theme(
    legend.position = "none",
     axis.title = element_blank(),
     axis.text = element_blank(),
     panel.grid = element_blank(),
     panel.grid.major = element_blank(),
  )

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.
```

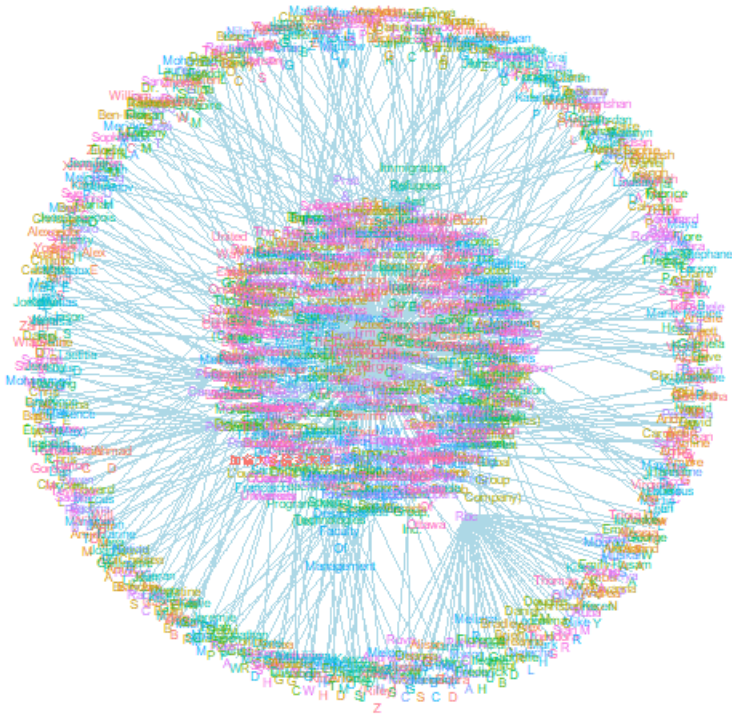



```
theme_set(thm)
graph_connections %>%
  ggraph(layout = "kk") +
    geom_node_point() +
    geom_edge_diagonal() +
  geom_node_text(aes(label = label, alpha=0.1))
```

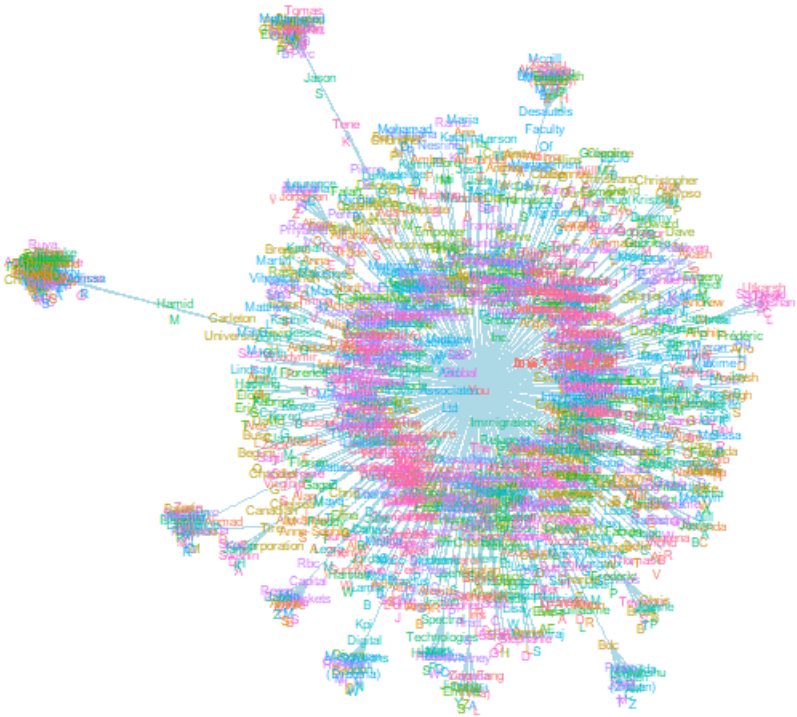



```
lapply(c('stress', 'fr', 'lgl', 'graphopt','kk'), function(layout) {
  graph_connections %>%
    ggraph(layout = layout) +
    geom_edge_fan(width = .2, color = 'lightblue') +
    geom_node_text(aes(label = label, color = name), size = 2) +
    coord_fixed()+
    scale_fill_brewer()
})
```

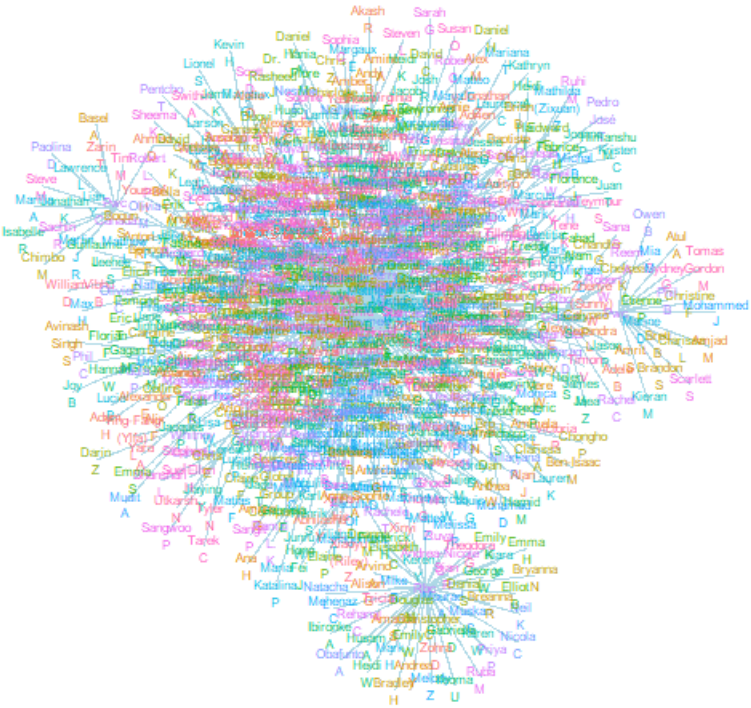
```
## [[1]]
```



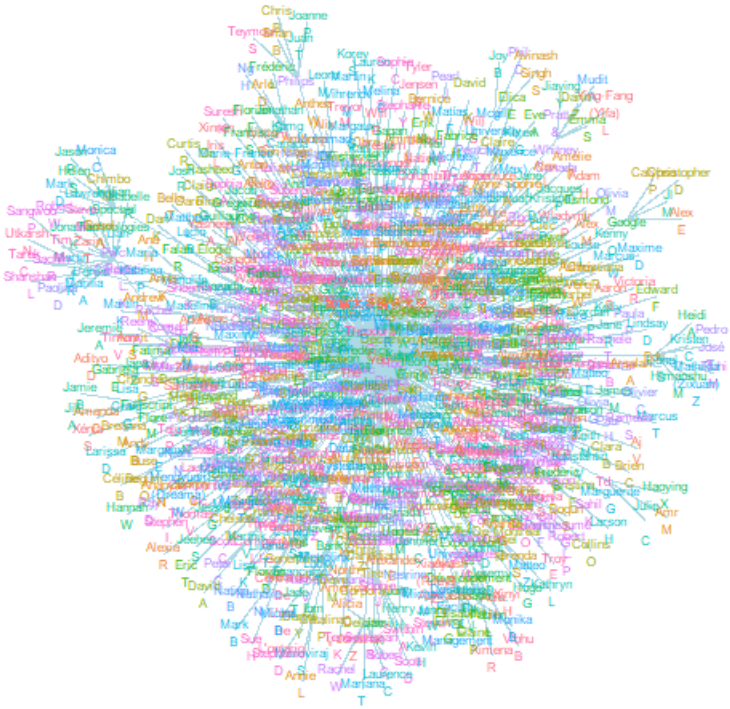
```
##
## [[2]]
```



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



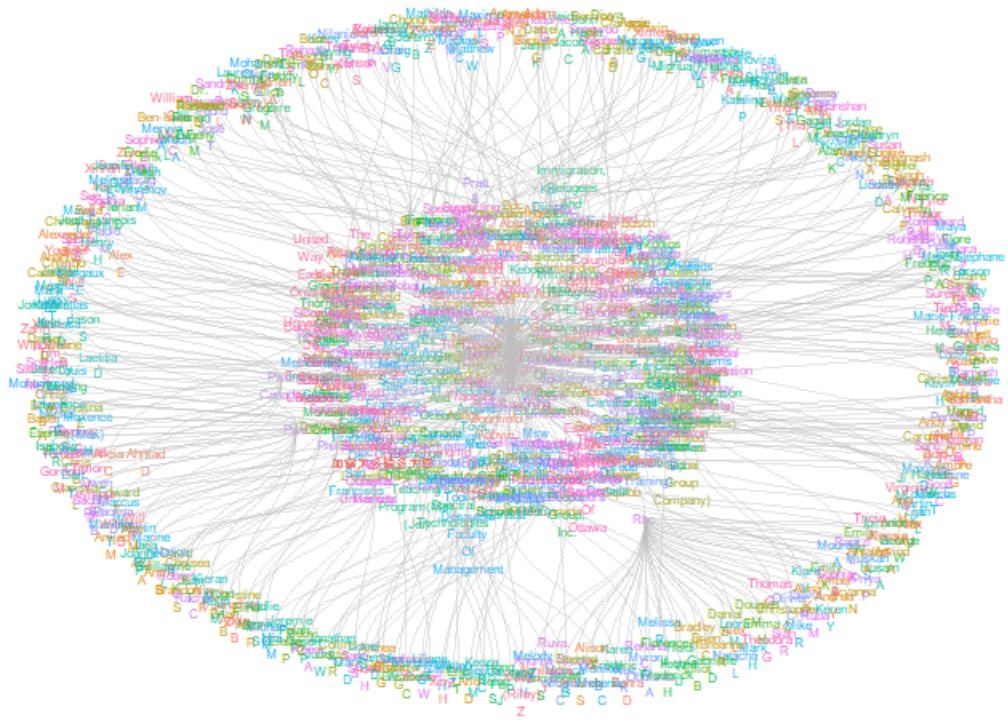
```
##
## [[3]]
```

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



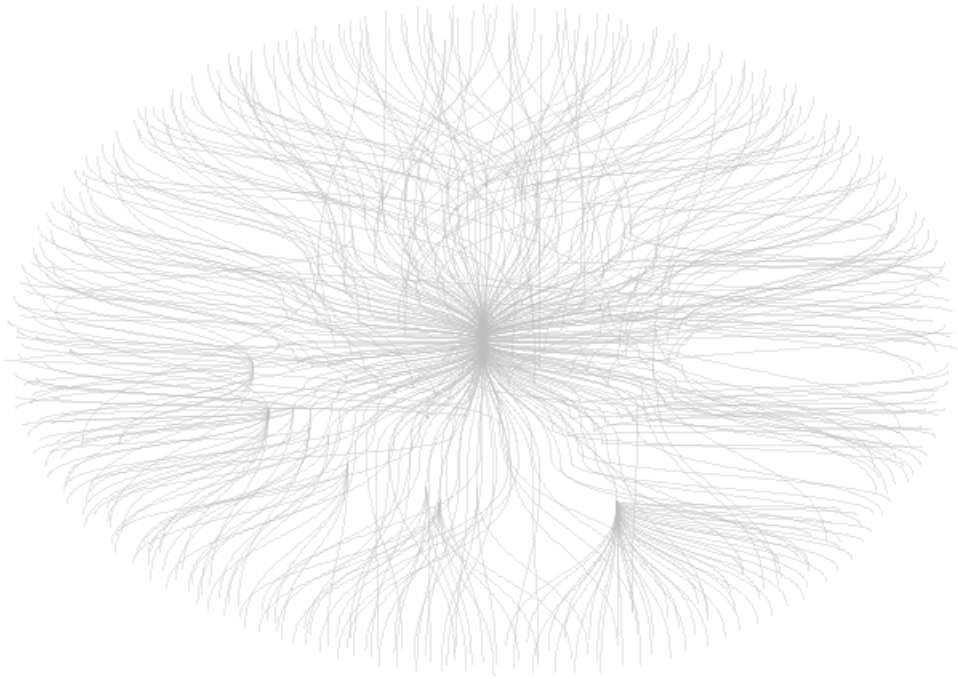
```
##
## [[5]]
```

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

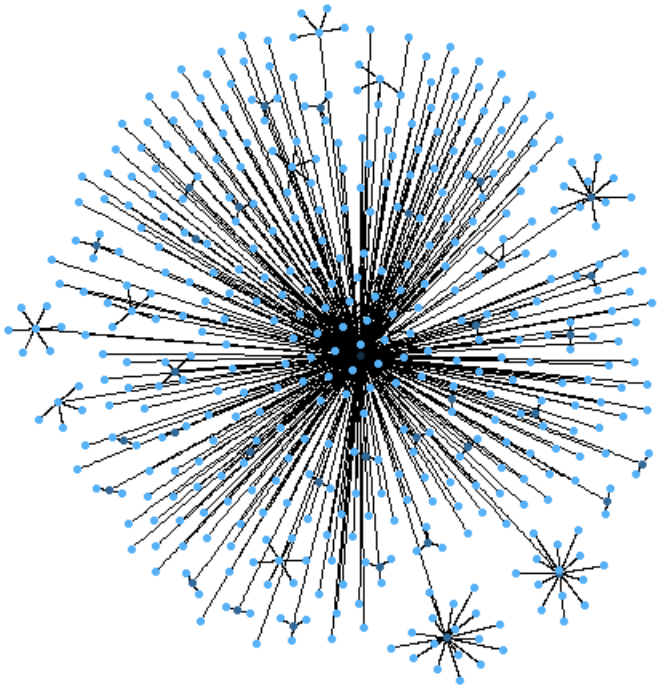


```
graph_connections %>%
  ggraph(layout = "stress") +
```

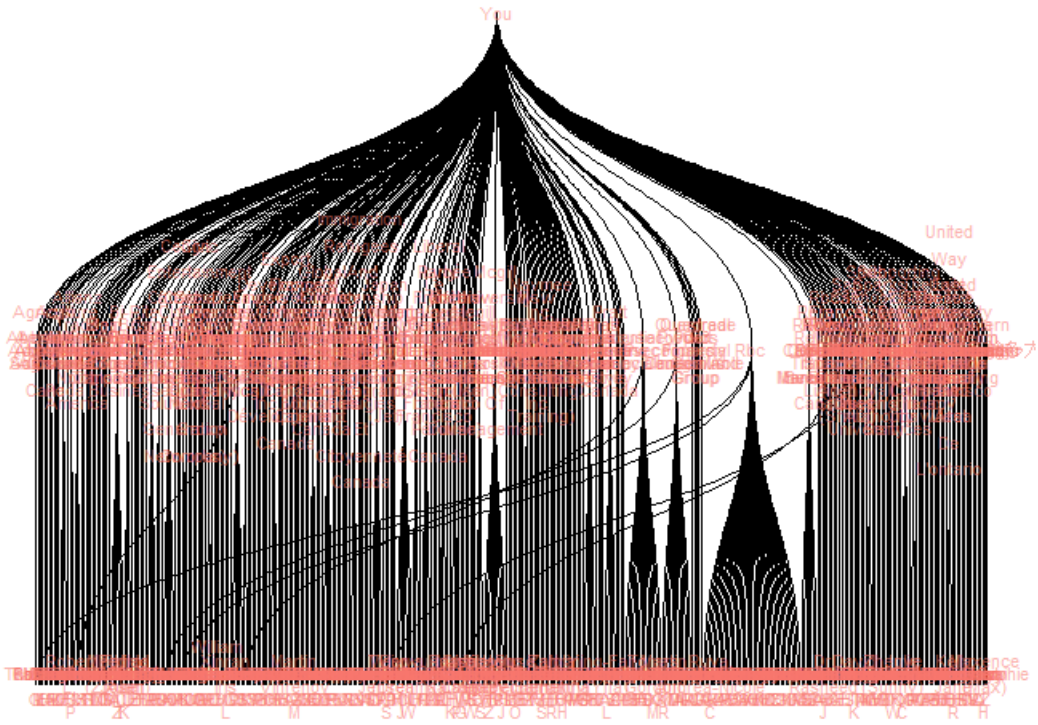
```
    geom_edge_diagonal(color = "gray", alpha = 0.4)
```



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


## Multiple parents. Unfolding graph
```



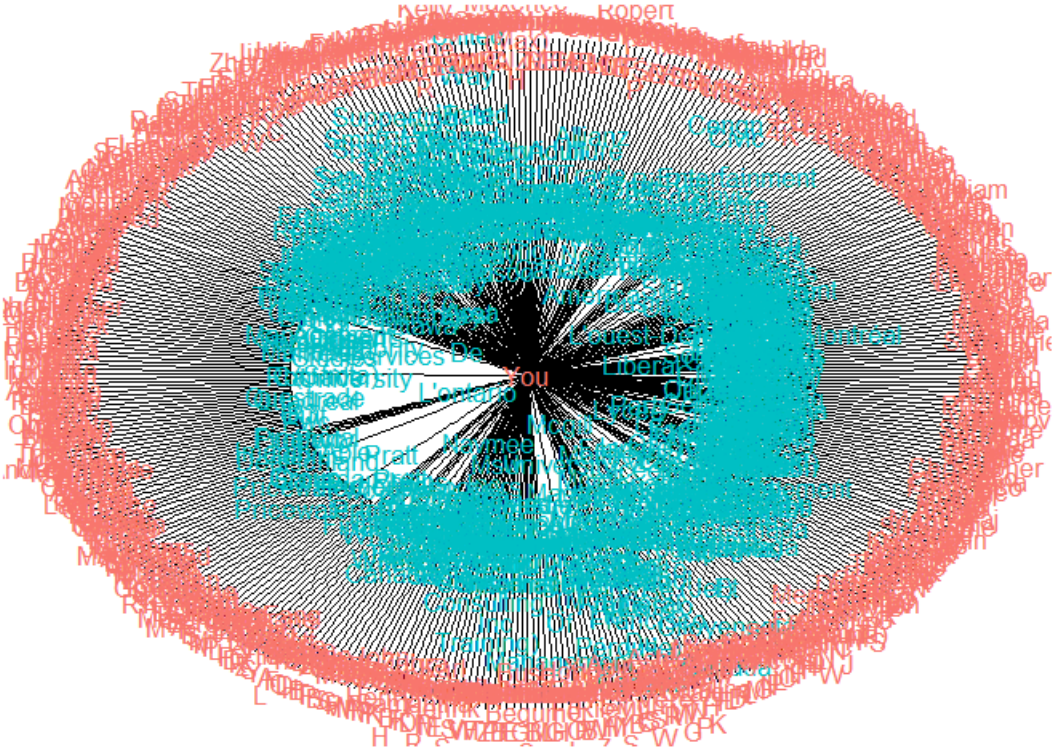
```
ggraph(graph_connections, 'tree') +
  geom_edge_diagonal()+
  geom_node_text(aes(label = label,alpha=0.2, color='blue'),size=3)
```




```
ggraph(graph_connections, 'dendrogram', circular = TRUE ) +
  geom_edge_elbow() +
  #coord_fixed()+
```

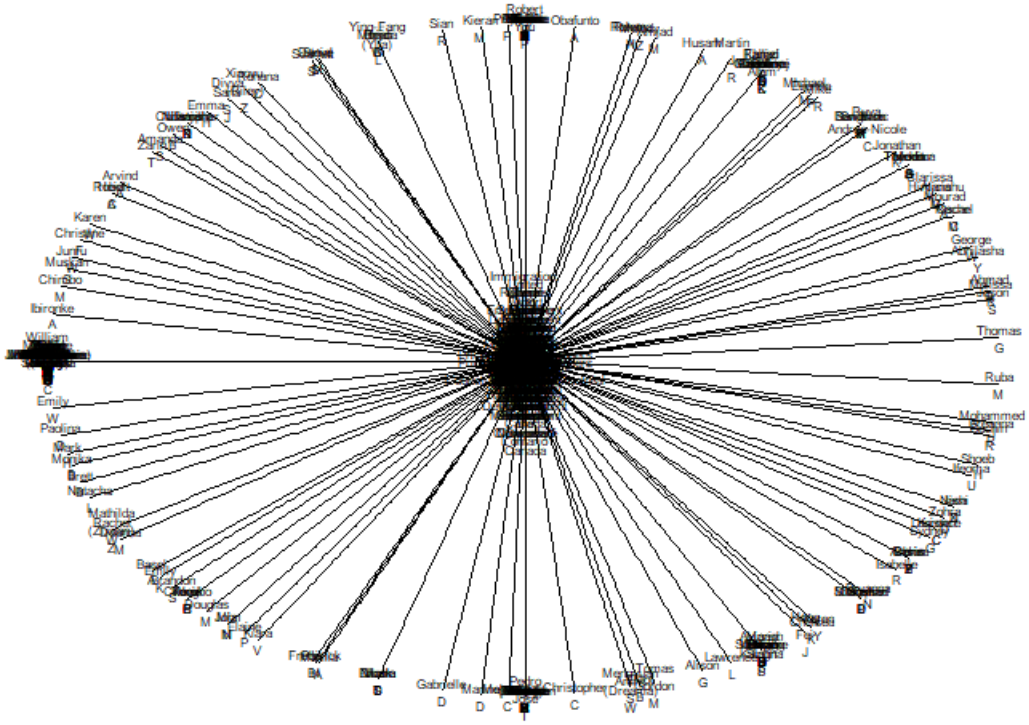
```
  #geom_node_text(aes(label = label , color = name), size = 2)+
  geom_node_text(aes(label = label , color = ifelse(name %in% companies, "red","black"), size = 2))
```

```
## 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 implementation

Credit to:http://users.dimi.uniud.it/~massimo.franceschet/ns/syllabus/make/tidygraph/tidygraph.html

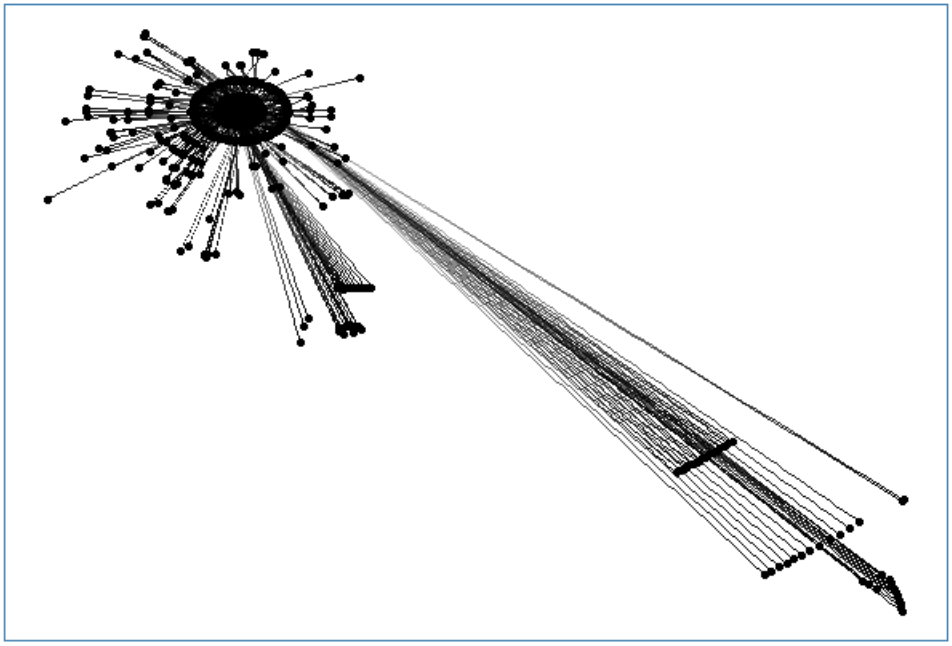
```
# graph analysis and visualziation
library(tidygraph)
library(ggraph)
library(igraph)
```

```
##
## 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')
```

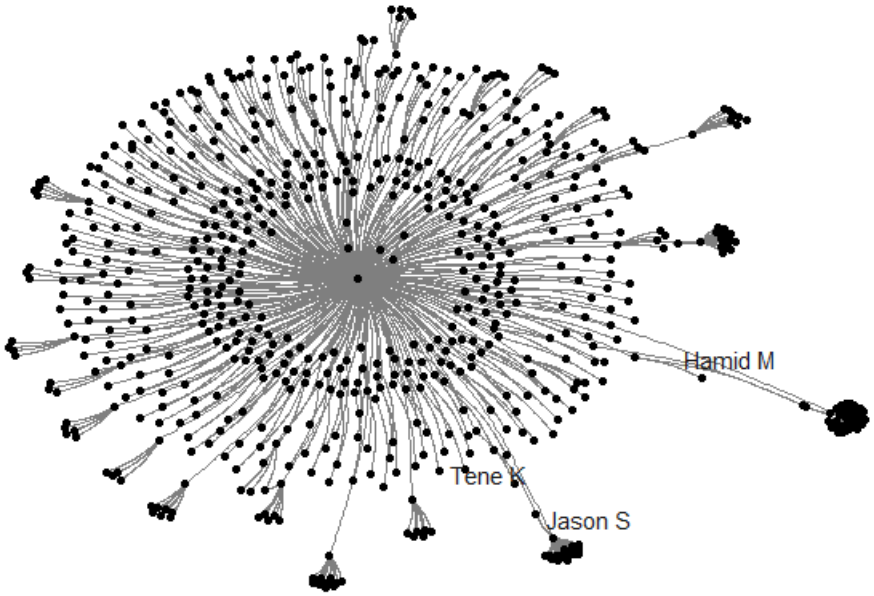


Visualizing the table is tricky but we can do some data manipulations to show what is happening. **help from:** **https://stackoverflow.com/questions/63997659/get-edge-data-from-the-tidygraph-package**

```
# 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: 671 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

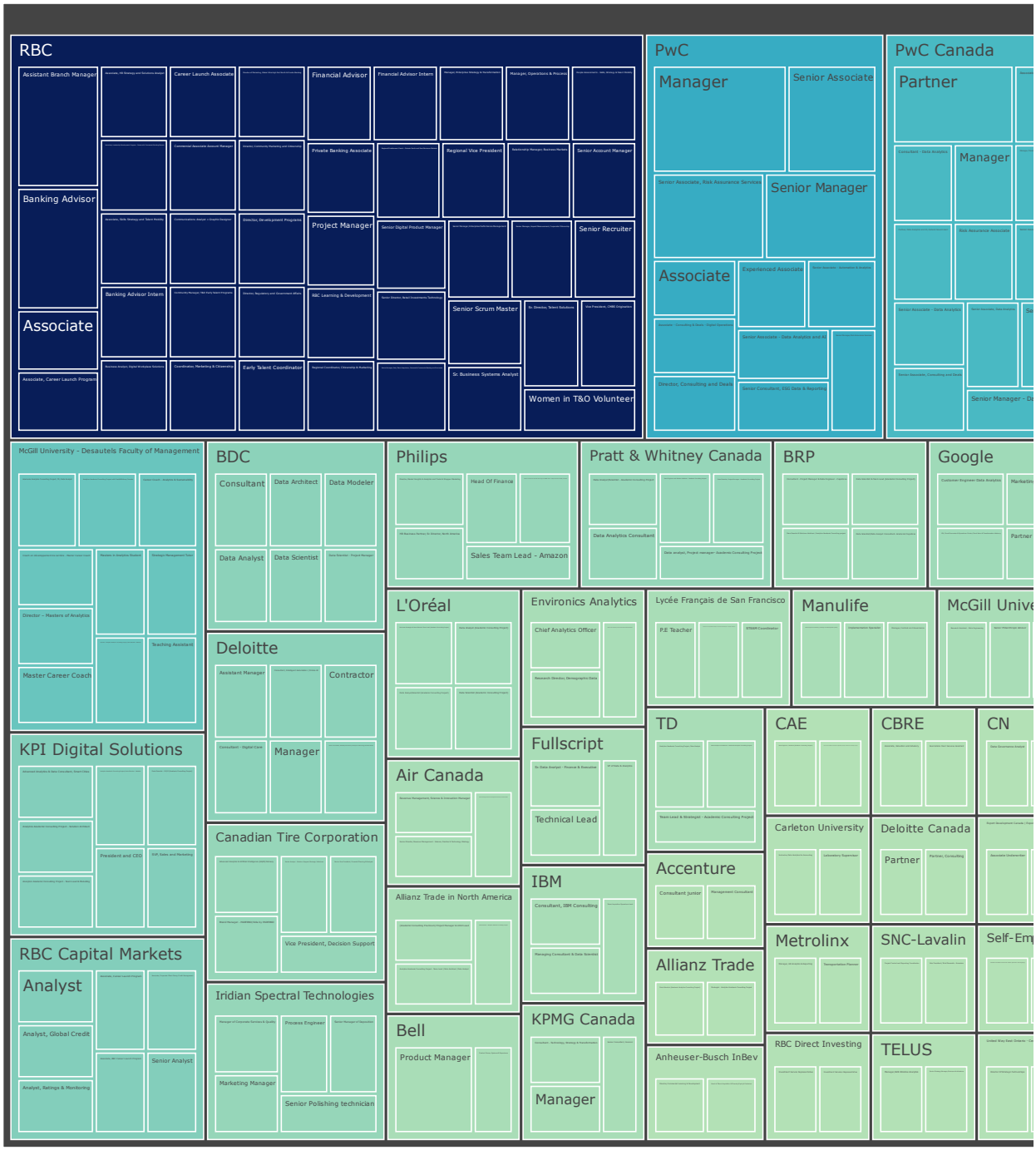


Vsiualization

Visulize the results in python's plotly

```
import pandas as pd
import numpy as np
df = pd.read_csv('Connections.csv',skiprows=3)
df1=df.groupby("Company").filter(lambda x: len(x) > 1)
df1['Count'] = df1.groupby('Company')['Company'].transform('size')
df1['Count']=df1['Count'].astype(int)
import plotly.express as px
```

```
df1=df1.dropna(subset=['Company', 'Position'])
px.treemap(df1, path=['Company', 'Position'],
           width=1000,
           height=1000,
           color='Count',
           color_continuous_scale='YlGnBu',
           color_continuous_midpoint=np.average(df1['Count']))
```



```
df1=df.groupby("Position").filter(lambda x: len(x) > 1)
df1['Count'] = df1.groupby('Position')['Position'].transform('size')
df1['Count']=df1['Count'].astype(int)
px.treemap(df1, path=['Position','Company'],
           width=1000,
           height=1000,
           color='Count',
           color_continuous_scale='Inferno',
           color_continuous_midpoint=np.average(df1['Count']))
```