Social Network Analysis

MEENOWA Sarvesh

02/12/2021

```
## Load important libraries
library(igraph)
library(readr)
library(dplyr)
library(networkD3)
library(gganimate)
#library(gapminder)
library(ggplot2)
library(tidyverse)
library(circlize)
library(ggraph)
library(tidygraph)
library(corrr)
library(RColorBrewer)
```

```
#import dataframe
df <- read_csv("H:/Downloads/Datatsets/Auditions.db.comp (1).csv", locale = locale(encoding = "ISO-8859-
head(df)
```

```
## # A tibble: 6 x 12
##   Name   n_poste Level Section Role  status  year institutions ID    X     X.1
##   <chr>    <dbl> <chr> <chr>   <chr> <chr>  <dbl> <chr>        <chr> <lgl> <lgl>
## 1  Phil~     193 PR    <NA>    <NA>  inter~  2017 <NA>         8052~ NA    NA
## 2 A Bel~      20 <NA>  <NA>    <NA>  inter~  2020 CY Cergy Pa~ 1131~ NA    NA
## 3 A Bre~      62 <NA>  <NA>    <NA>  exter~  2020 <NA>         7054~ NA    NA
## 4 A evin    4279 <NA>  <NA>    <NA>  exter~  2020 <NA>         1590~ NA    NA
## 5 A Gom~    4647 <NA>  <NA>    <NA>  exter~  2020 <NA>         6993~ NA    NA
## 6 A Kuz~    4314 PR    7       <NA>  inter~  2017 Paris 7     3558~ NA    NA
## # ... with 1 more variable: Id.author.no <chr>
```

Simple SNA

Create a subset of the dataset and represent a bipartite and a multipartite graph of internal members, and the same thing external members (do not mix years) : 1. 2017, internal members 2. 2017, external members 3. 2018, internal members 4. 2018, external members 5. 2019, internal members 6. 2019, external members 7. 2020, internal members 8. 2020, external members

```
#2017 internal members
interne_2017 = df %>% filter(status == 'interne' & year == 2017)
#2017 external members
```

```
externe_2017 = df %>% filter(status == 'externe' & year == 2017)
#2018 internal members
interne_2018 = df %>% filter(status == 'interne' & year == 2018)
#2018 external members
externe_2018 = df %>% filter(status == 'externe' & year == 2018)
#2019 internal members
interne_2019 = df %>% filter(status == 'interne' & year == 2019)
#2019 external members
externe_2019 = df %>% filter(status == 'externe' & year == 2019)
#2020 internal members
interne_2020 = df %>% filter(status == 'interne' & year == 2020)
#2020 external members
externe_2020 = df %>% filter(status == 'externe' & year == 2020)
```

Use gganimate (in R, or any Python equivalent) to create a GIF-based succession of graphs for the years 2017 to 2020, for external members only(GGanimate doesn't work on networks) . Make an interactive graph of one year (internal members) with networkD3 (or the Python equivalent).

```
#create edgelist for every dataset
#extern 2017
externe_2017_edge_list_name <- externe_2017 %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(externe_2017_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1             Name2
##     <dbl> <chr>             <chr>
## 1      13 Caroline DESOMBRE Claire PERRIN
## 2      13 Caroline DESOMBRE Cyril CROZET
## 3      13 Caroline DESOMBRE Dominique BERGER
## 4      13 Caroline DESOMBRE Jeanine POMMIER
## 5      13 Caroline DESOMBRE Thierry PIOT
## 6      13 Claire PERRIN     Caroline DESOMBRE
```

```
#create node list

#  Get distinct source names
sources1 <-externe_2017_edge_list_name  %>% distinct(Name1) %>% rename(label = Name1)
# Get distinct destination names
destinations1 <- externe_2017_edge_list_name %>%
  distinct(Name2) %>%
  rename(label = Name2)
# Join the two data to create node
# Add unique ID for each country
nodes1 <- inner_join(sources1, destinations1, by = "label")
nodes1 <- nodes1 %>%
  mutate(id = 1:nrow(nodes1)) %>%
```

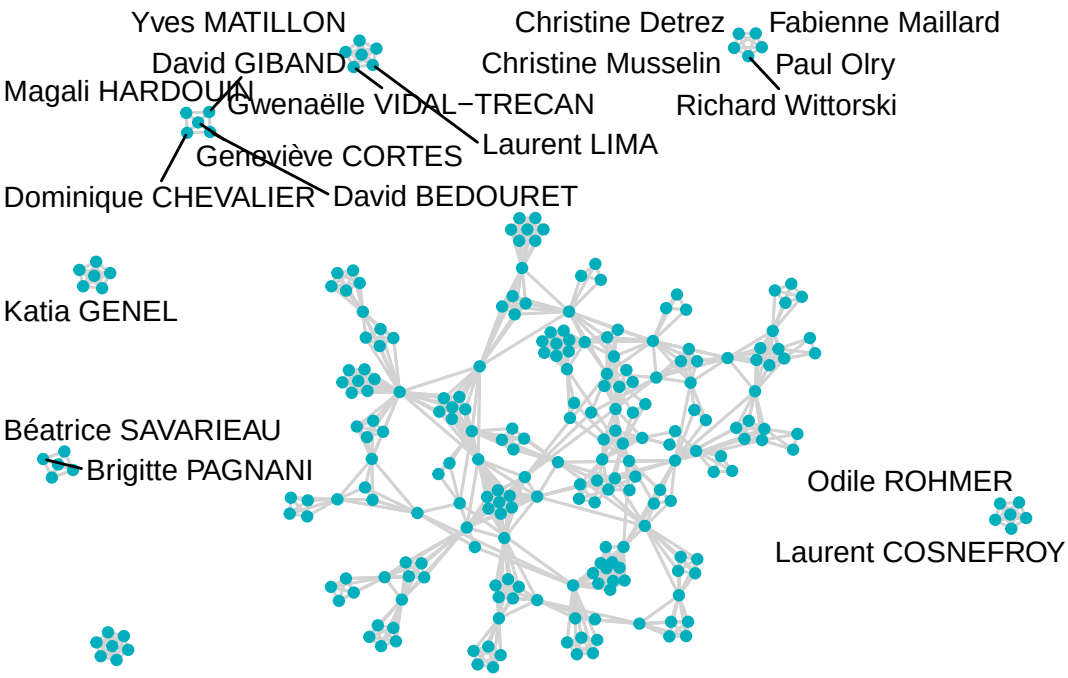
```
  select(id, everything())
head(nodes1, 3)
```

```
## # A tibble: 3 x 2
##      id label
##   <int> <chr>
## 1     1 Caroline DESOMBRE
## 2     2 Claire PERRIN
## 3     3 Cyril CROZET
```

```
# Rename the n.call column to weight
 externe_2017_edge_list_name <- externe_2017_edge_list_name %>%
  rename(weight = n_poste)
## (a) Join nodes id for source column
edges1 <-externe_2017_edge_list_name  %>%
  left_join(nodes1, by = c("Name1" = "label")) %>%
  rename(from = id)
# (b) Join nodes id for destination column
edges1 <- edges1 %>%
  left_join(nodes1, by = c("Name2" = "label")) %>%
  rename(to = id)
# (c) Select/keep only the columns from and to
edges1 <- select(edges1, from, to, weight)
head(edges1, 3)
```

```
## # A tibble: 3 x 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1     2     13
## 2     1     3     13
## 3     1     4     13
```

```
set.seed(4)
net.tidy1 <- tbl_graph(
  nodes = nodes1, edges = edges1, directed = TRUE
  )
ggraph(net.tidy1, layout = "graphopt") +
  geom_edge_link(width = 0.5, colour = "lightgray") +
  geom_node_point(size = 1.5, colour = "#00AFBB") +
  geom_node_text(aes(label = label), repel = TRUE)+
  theme_graph()
```




```
#interne 2017
interne_2017_edge_list_name <- interne_2017 %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(interne_2017_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1         Name2
##     <dbl> <chr>         <chr>
## 1      13 Ludovic MORGE Marc DAGUZON
## 2      13 Ludovic MORGE Marie-Christine TOCZEK-CAPELLE
## 3      13 Ludovic MORGE Nathalie GAL-PETITFAUX
## 4      13 Marc DAGUZON  Ludovic MORGE
## 5      13 Marc DAGUZON  Marie-Christine TOCZEK-CAPELLE
## 6      13 Marc DAGUZON  Nathalie GAL-PETITFAUX
```

```
#create node list

#  Get distinct source names
sources2 <-interne_2017_edge_list_name  %>% distinct(Name1) %>% rename(label = Name1)
# Get distinct destination names
```

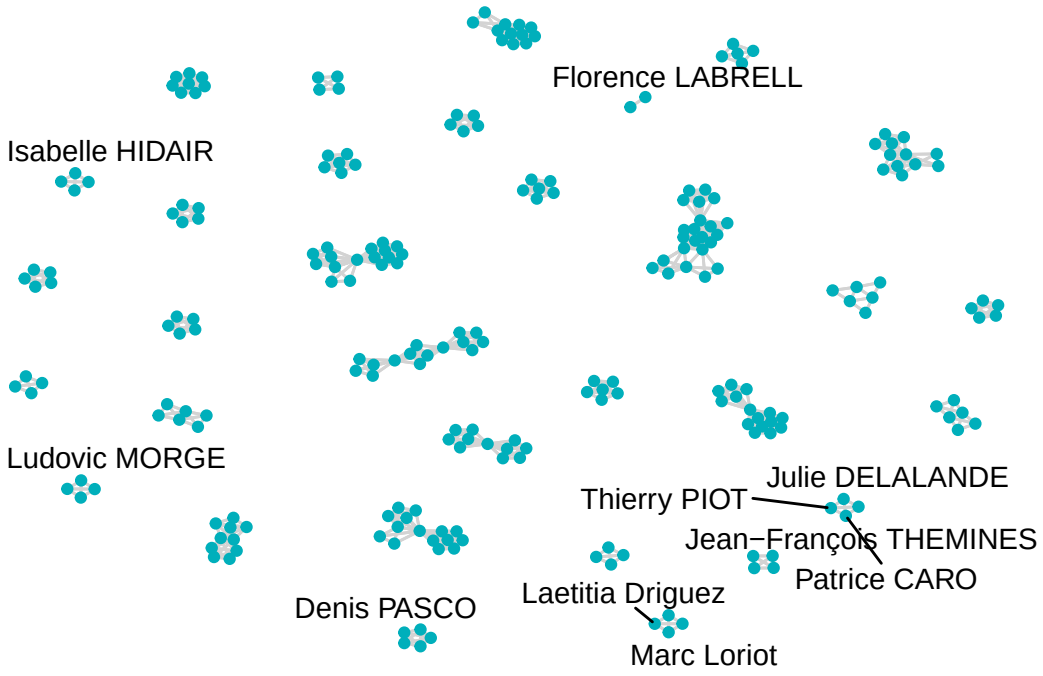
```r
destinations2 <- interne_2017_edge_list_name %>%
  distinct(Name2) %>%
  rename(label = Name2)
# Join the two data to create node
# Add unique ID for each country
nodes2 <- inner_join(sources2, destinations2, by = "label")
nodes2 <- nodes2 %>%
  mutate(id = 1:nrow(nodes2)) %>%
  select(id, everything())
head(nodes2, 3)
```

```
## # A tibble: 3 x 2
##      id label
##   <int> <chr>
## 1     1 Ludovic MORGE
## 2     2 Marc DAGUZON
## 3     3 Marie-Christine TOCZEK-CAPELLE
```

```r
# Rename the n.call column to weight
 interne_2017_edge_list_name <- interne_2017_edge_list_name %>%
  rename(weight = n_poste)
## (a) Join nodes id for source column
edges2 <-interne_2017_edge_list_name  %>%
  left_join(nodes2, by = c("Name1" = "label")) %>%
  rename(from = id)
# (b) Join nodes id for destination column
edges2 <- edges2 %>%
  left_join(nodes2, by = c("Name2" = "label")) %>%
  rename(to = id)
# (c) Select/keep only the columns from and to
edges2 <- select(edges2, from, to, weight)
head(edges2, 3)
```

```
## # A tibble: 3 x 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1     2     13
## 2     1     3     13
## 3     1     4     13
```

```r
set.seed(12)
net.tidy2 <- tbl_graph(
  nodes = nodes2, edges = edges2, directed = TRUE
  )
ggraph(net.tidy2, layout = "graphopt") +
  geom_edge_link(width = 0.5, colour = "lightgray") +
  geom_node_point(size = 1.5, colour = "#00AFBB") +
  geom_node_text(aes(label = label), repel = TRUE)+
  theme_graph()
```




```
#externe 2018
externe_2018_edge_list_name <- externe_2018 %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(externe_2018_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1               Name2
##     <dbl> <chr>               <chr>
## 1      28 BETRANCOURT Mireille CHARLIER Bernadette
## 2      28 BETRANCOURT Mireille Geneviève LAMEUL
## 3      28 BETRANCOURT Mireille Jérôme ENEAU
## 4      28 BETRANCOURT Mireille Philippe CARRÉ
## 5      28 CHARLIER Bernadette  BETRANCOURT Mireille
## 6      28 CHARLIER Bernadette  Geneviève LAMEUL
```

```
#create node list

#  Get distinct source names
sources3 <-externe_2018_edge_list_name  %>% distinct(Name1) %>% rename(label = Name1)
# Get distinct destination names
```

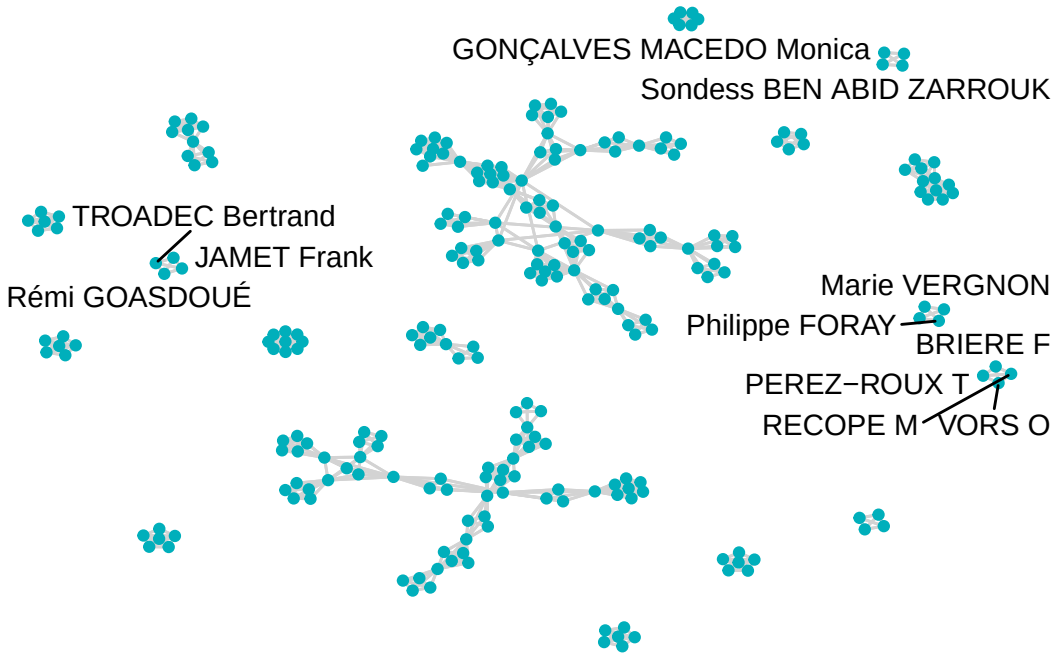
```r
destinations3 <- externe_2018_edge_list_name %>%
  distinct(Name2) %>%
  rename(label = Name2)
# Join the two data to create node
# Add unique ID for each country
nodes3 <- inner_join(sources3, destinations3, by = "label")
nodes3 <- nodes3 %>%
  mutate(id = 1:nrow(nodes3)) %>%
  select(id, everything())
head(nodes3, 3)
```

```
## # A tibble: 3 x 2
##      id label
##   <int> <chr>
## 1     1 BETRANCOURT Mireille
## 2     2 CHARLIER Bernadette
## 3     3 Geneviève LAMEUL
```

```r
# Rename the n.call column to weight
 externe_2018_edge_list_name <- externe_2018_edge_list_name %>%
  rename(weight = n_poste)
## (a) Join nodes id for source column
edges3 <-externe_2018_edge_list_name  %>%
  left_join(nodes3, by = c("Name1" = "label")) %>%
  rename(from = id)
# (b) Join nodes id for destination column
edges3 <- edges3 %>%
  left_join(nodes3, by = c("Name2" = "label")) %>%
  rename(to = id)
# (c) Select/keep only the columns from and to
edges3 <- select(edges3, from, to, weight)
head(edges3, 3)
```

```
## # A tibble: 3 x 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1     2     28
## 2     1     3     28
## 3     1     4     28
```

```r
set.seed(123)
net.tidy3 <- tbl_graph(
  nodes = nodes3, edges = edges3, directed = TRUE
  )
ggraph(net.tidy3, layout = "graphopt") +
  geom_edge_link(width = 0.5, colour = "lightgray") +
  geom_node_point(size = 1.5, colour = "#00AFBB") +
  geom_node_text(aes(label = label), repel = TRUE)+
  theme_graph()
```




```
#interne 2018
interne_2018_edge_list_name <- interne_2018 %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(interne_2018_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1           Name2
##     <dbl> <chr>           <chr>
## 1      28 Annie JEZEGOU   Gilles LECLERCQ
## 2      28 Annie JEZEGOU   Maria PAGONI
## 3      28 Annie JEZEGOU   Mokhtar KADDOURI
## 4      28 Annie JEZEGOU   Olivier LAS VERGNAS
## 5      28 Gilles LECLERCQ Annie JEZEGOU
## 6      28 Gilles LECLERCQ Maria PAGONI
```

```
#create node list

#  Get distinct source names
sources4 <-interne_2018_edge_list_name  %>% distinct(Name1) %>% rename(label = Name1)
# Get distinct destination names
```

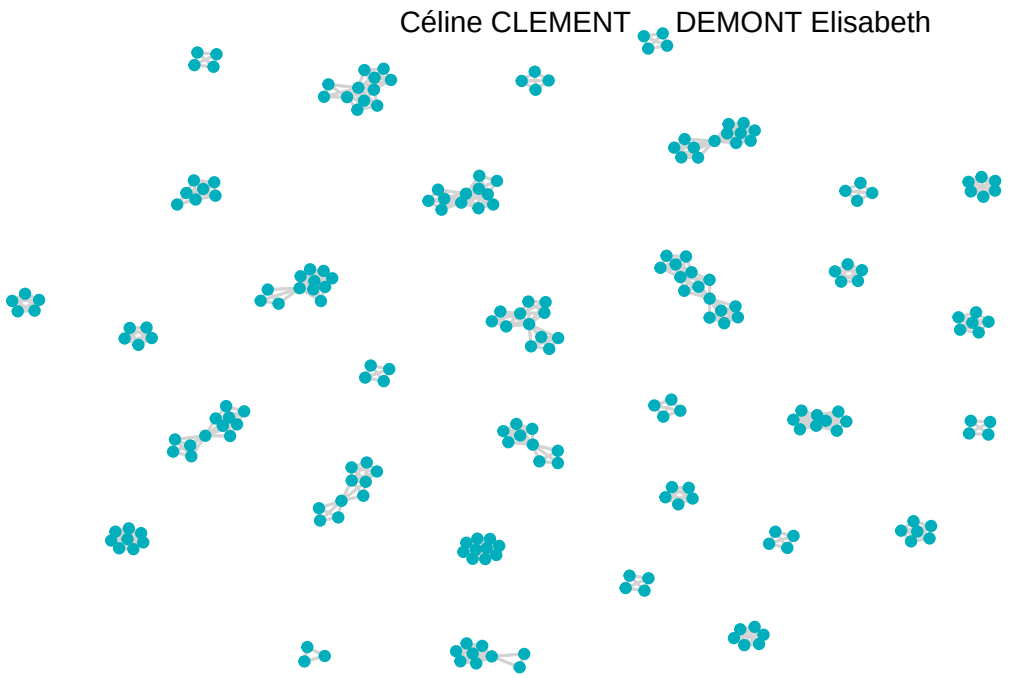
```r
destinations4 <- interne_2018_edge_list_name %>%
  distinct(Name2) %>%
  rename(label = Name2)
# Join the two data to create node
# Add unique ID for each country
nodes4 <- inner_join(sources4, destinations4, by = "label")
nodes4 <- nodes4 %>%
  mutate(id = 1:nrow(nodes4)) %>%
  select(id, everything())
head(nodes4, 3)
```

```
## # A tibble: 3 x 2
##      id label
##   <int> <chr>
## 1     1 Annie JEZEGOU
## 2     2 Gilles LECLERCQ
## 3     3 Maria PAGONI
```

```r
# Rename the n.call column to weight
 interne_2018_edge_list_name <- interne_2018_edge_list_name %>%
  rename(weight = n_poste)
## (a) Join nodes id for source column
edges4 <-interne_2018_edge_list_name  %>%
  left_join(nodes4, by = c("Name1" = "label")) %>%
  rename(from = id)
# (b) Join nodes id for destination column
edges4 <- edges4 %>%
  left_join(nodes4, by = c("Name2" = "label")) %>%
  rename(to = id)
# (c) Select/keep only the columns from and to
edges4 <- select(edges4, from, to, weight)
head(edges4, 3)
```

```
## # A tibble: 3 x 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1     2     28
## 2     1     3     28
## 3     1     4     28
```

```r
set.seed(154)
net.tidy4 <- tbl_graph(
  nodes = nodes4, edges = edges4, directed = TRUE
  )
ggraph(net.tidy4, layout = "graphopt") +
  geom_edge_link(width = 0.5, colour = "lightgray") +
  geom_node_point(size = 1.5, colour = "#00AFBB") +
  geom_node_text(aes(label = label), repel = TRUE)+
  theme_graph()
```




```
#interne 2019
interne_2019_edge_list_name <- interne_2019 %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(interne_2019_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1            Name2
##     <dbl> <chr>            <chr>
## 1      56 Abdelkarim ZAID  Brigitte Monfroy
## 2      56 Abdelkarim ZAID  Couturier Catherine
## 3      56 Abdelkarim ZAID  Jean-François GOUBET
## 4      56 Abdelkarim ZAID  Mierzejewski Stephan
## 5      56 Abdelkarim ZAID  Sylvain Broccolichi
## 6      56 Brigitte Monfroy Abdelkarim ZAID
```

```
#create node list

#  Get distinct source names
sources5 <-interne_2019_edge_list_name  %>% distinct(Name1) %>% rename(label = Name1)
# Get distinct destination names
```

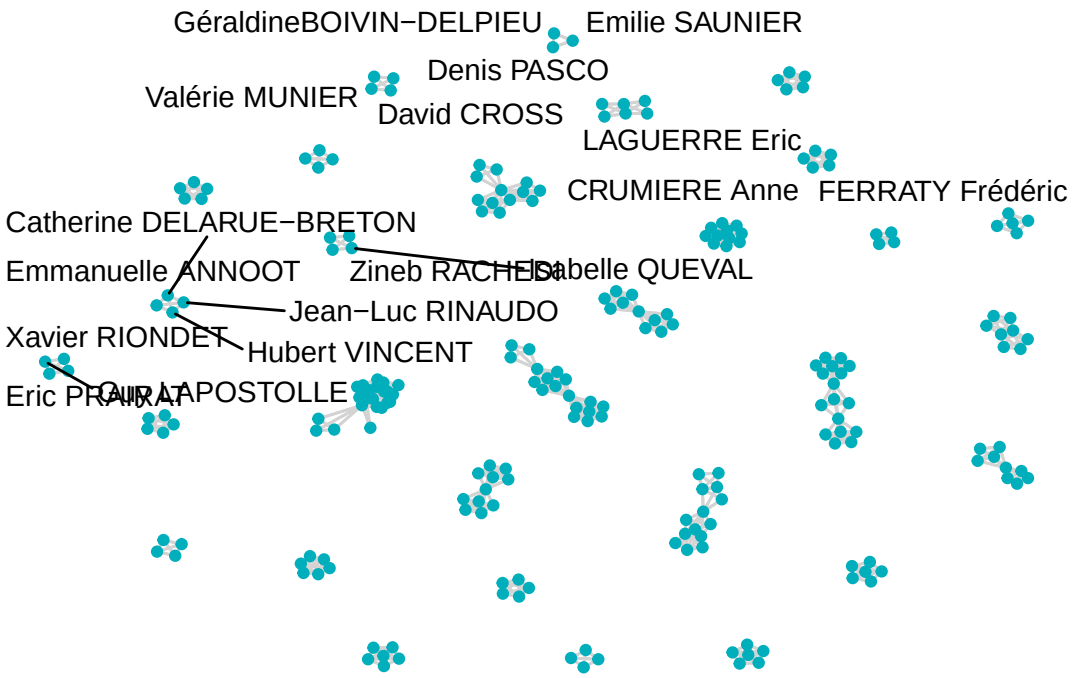
```r
destinations5 <- interne_2019_edge_list_name %>%
  distinct(Name2) %>%
  rename(label = Name2)
# Join the two data to create node
# Add unique ID for each country
nodes5 <- inner_join(sources5, destinations5, by = "label")
nodes5 <- nodes5 %>%
  mutate(id = 1:nrow(nodes5)) %>%
  select(id, everything())
head(nodes5, 3)
```

```
## # A tibble: 3 x 2
##      id label
##   <int> <chr>
## 1     1 Abdelkarim ZAID
## 2     2 Brigitte Monfroy
## 3     3 Couturier Catherine
```

```r
# Rename the n.call column to weight
 interne_2019_edge_list_name <- interne_2019_edge_list_name %>%
  rename(weight = n_poste)
## (a) Join nodes id for source column
edges5 <-interne_2019_edge_list_name  %>%
  left_join(nodes5, by = c("Name1" = "label")) %>%
  rename(from = id)
# (b) Join nodes id for destination column
edges5 <- edges5 %>%
  left_join(nodes5, by = c("Name2" = "label")) %>%
  rename(to = id)
# (c) Select/keep only the columns from and to
edges5 <- select(edges5, from, to, weight)
head(edges5, 3)
```

```
## # A tibble: 3 x 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1     2     56
## 2     1     3     56
## 3     1     4     56
```

```r
set.seed(154)
net.tidy5 <- tbl_graph(
  nodes = nodes5, edges = edges5, directed = TRUE
  )
ggraph(net.tidy5, layout = "graphopt") +
  geom_edge_link(width = 0.5, colour = "lightgray") +
  geom_node_point(size = 1.5, colour = "#00AFBB") +
  geom_node_text(aes(label = label), repel = TRUE)+
  theme_graph()
```




```
#externe 2019
externe_2019_edge_list_name <- externe_2019 %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(externe_2019_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1                     Name2
##     <dbl> <chr>                     <chr>
## 1      56 Anne-Claudine Oller-naudet Anne Barrère
## 2      56 Anne-Claudine Oller-naudet Farges Géraldine
## 3      56 Anne-Claudine Oller-naudet Garcia Sandrine
## 4      56 Anne-Claudine Oller-naudet Maillart Alain
## 5      56 Anne-Claudine Oller-naudet Netter Julien
## 6      56 Anne Barrère               Anne-Claudine Oller-naudet
```

```
#create node list

#  Get distinct source names
sources6 <-externe_2019_edge_list_name  %>% distinct(Name1) %>% rename(label = Name1)
# Get distinct destination names
```

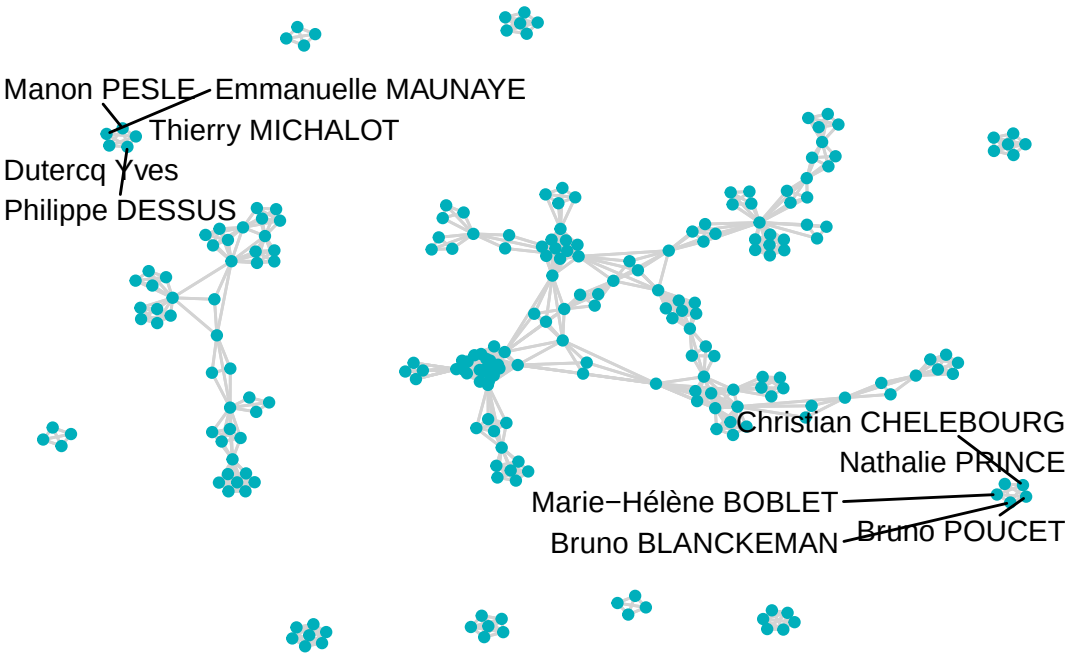
```r
destinations6 <- externe_2019_edge_list_name %>%
  distinct(Name2) %>%
  rename(label = Name2)
# Join the two data to create node
# Add unique ID for each country
nodes6 <- inner_join(sources6, destinations6, by = "label")
nodes6 <- nodes6 %>%
  mutate(id = 1:nrow(nodes6)) %>%
  select(id, everything())
head(nodes6, 3)
```

```
## # A tibble: 3 x 2
##      id label
##   <int> <chr>
## 1     1 Anne-Claudine Oller-naudet
## 2     2 Anne Barrère
## 3     3 Farges Géraldine
```

```r
# Rename the n.call column to weight
 externe_2019_edge_list_name <- externe_2019_edge_list_name %>%
  rename(weight = n_poste)
## (a) Join nodes id for source column
edges6 <-externe_2019_edge_list_name  %>%
  left_join(nodes6, by = c("Name1" = "label")) %>%
  rename(from = id)
# (b) Join nodes id for destination column
edges6 <- edges6 %>%
  left_join(nodes6, by = c("Name2" = "label")) %>%
  rename(to = id)
# (c) Select/keep only the columns from and to
edges6 <- select(edges6, from, to, weight)
head(edges6, 3)
```

```
## # A tibble: 3 x 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1     2     56
## 2     1     3     56
## 3     1     4     56
```

```r
set.seed(154)
net.tidy6 <- tbl_graph(
  nodes = nodes6, edges = edges6, directed = TRUE
  )
ggraph(net.tidy6, layout = "graphopt") +
  geom_edge_link(width = 0.5, colour = "lightgray") +
  geom_node_point(size = 1.5, colour = "#00AFBB") +
  geom_node_text(aes(label = label), repel = TRUE)+
  theme_graph()
```




```
#externe 2020
externe_2020_edge_list_name <- externe_2020 %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(externe_2020_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1          Name2
##     <dbl> <chr>          <chr>
## 1      19 Alain Frugière D Cross
## 2      19 Alain Frugière F Le Hebel
## 3      19 Alain Frugière Ludovic MORGE
## 4      19 Alain Frugière M Betrancourt
## 5      19 Alain Frugière M Gallezot
## 6      19 Alain Frugière S Fleck
```

```
#create node list

#  Get distinct source names
sources7<-externe_2020_edge_list_name  %>% distinct(Name1) %>% rename(label = Name1)
# Get distinct destination names
```

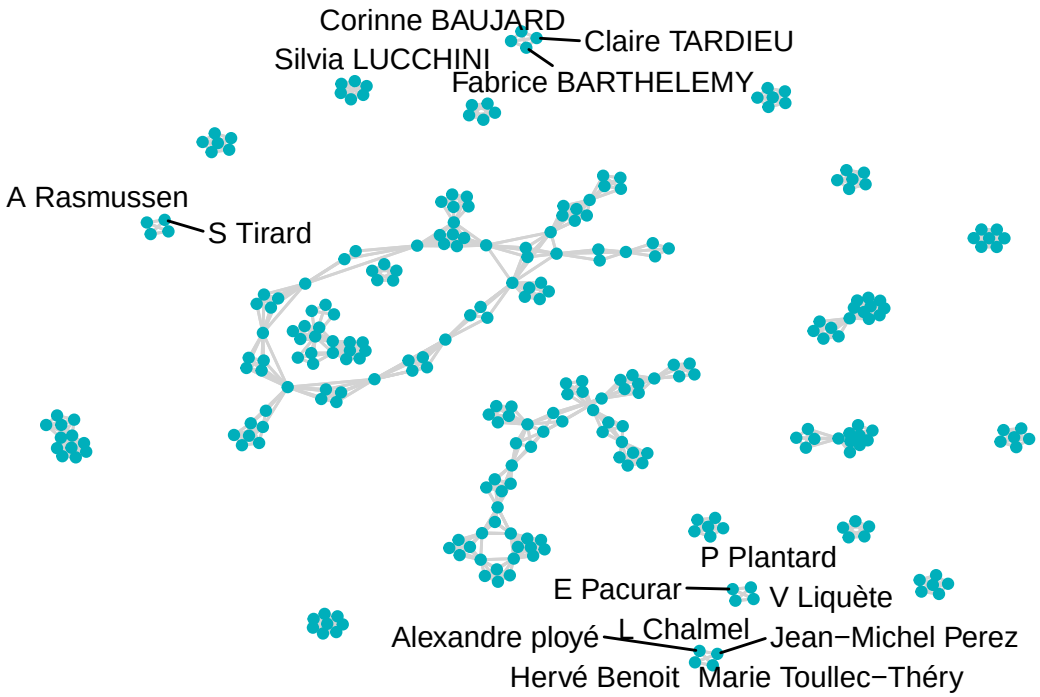
```r
destinations7 <- externe_2020_edge_list_name %>%
  distinct(Name2) %>%
  rename(label = Name2)
# Join the two data to create node
# Add unique ID for each country
nodes7<- inner_join(sources7, destinations7, by = "label")
nodes7 <- nodes7 %>%
  mutate(id = 1:nrow(nodes7)) %>%
  select(id, everything())
head(nodes7, 3)
```

```
## # A tibble: 3 x 2
##      id label
##   <int> <chr>
## 1     1 Alain Frugière
## 2     2 D Cross
## 3     3 F Le Hebel
```

```r
# Rename the n.call column to weight
 externe_2020_edge_list_name <- externe_2020_edge_list_name %>%
  rename(weight = n_poste)
## (a) Join nodes id for source column
edges7 <-externe_2020_edge_list_name  %>%
  left_join(nodes7, by = c("Name1" = "label")) %>%
  rename(from = id)
# (b) Join nodes id for destination column
edges7 <- edges7 %>%
  left_join(nodes7, by = c("Name2" = "label")) %>%
  rename(to = id)
# (c) Select/keep only the columns from and to
edges7 <- select(edges7, from, to, weight)
head(edges7, 3)
```

```
## # A tibble: 3 x 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1     2     19
## 2     1     3     19
## 3     1     4     19
```

```r
set.seed(123)
net.tidy7 <- tbl_graph(
  nodes = nodes7, edges = edges7, directed = TRUE
  )
ggraph(net.tidy7, layout = "graphopt") +
  geom_edge_link(width = 0.5, colour = "lightgray") +
  geom_node_point(size = 1.5, colour = "#00AFBB") +
  geom_node_text(aes(label = label), repel = TRUE)+
  theme_graph()
```




```
#interne 2020
interne_2020_edge_list_name <- interne_2020 %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(interne_2020_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1          Name2
##     <dbl> <chr>          <chr>
## 1      19 B Debu         Érica DE VRIES
## 2      19 B Debu         Hamid Chaachoua
## 3      19 B Debu         I Girault
## 4      19 B Debu         P Flore
## 5      19 Érica DE VRIES B Debu
## 6      19 Érica DE VRIES Hamid Chaachoua
```

```
#create node list

#  Get distinct source names
sources8 <-interne_2020_edge_list_name  %>% distinct(Name1) %>% rename(label = Name1)
# Get distinct destination names
```

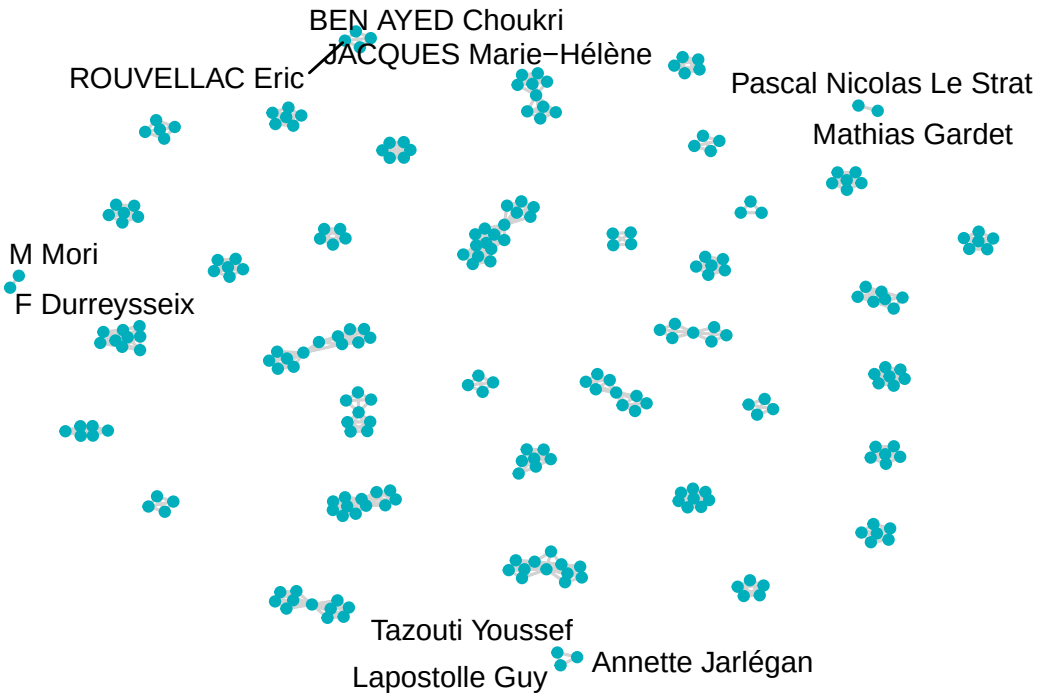
```r
destinations8 <- interne_2020_edge_list_name %>%
  distinct(Name2) %>%
  rename(label = Name2)
# Join the two data to create node
# Add unique ID for each country
nodes8<- inner_join(sources8, destinations8, by = "label")
nodes8 <- nodes8 %>%
  mutate(id = 1:nrow(nodes8)) %>%
  select(id, everything())
head(nodes8, 3)
```

```
## # A tibble: 3 x 2
##      id label
##   <int> <chr>
## 1     1 B Debu
## 2     2 Érica DE VRIES
## 3     3 Hamid Chaachoua
```

```r
# Rename the n.call column to weight
 interne_2020_edge_list_name <- interne_2020_edge_list_name %>%
  rename(weight = n_poste)
## (a) Join nodes id for source column
edges8 <-interne_2020_edge_list_name  %>%
  left_join(nodes8, by = c("Name1" = "label")) %>%
  rename(from = id)
# (b) Join nodes id for destination column
edges8 <- edges8 %>%
  left_join(nodes8, by = c("Name2" = "label")) %>%
  rename(to = id)
# (c) Select/keep only the columns from and to
edges8 <- select(edges8, from, to, weight)
head(edges8, 3)
```

```
## # A tibble: 3 x 3
##    from    to weight
##   <int> <int>  <dbl>
## 1     1     2     19
## 2     1     3     19
## 3     1     4     19
```

```r
set.seed(123)
net.tidy8 <- tbl_graph(
  nodes = nodes8, edges = edges8, directed = TRUE
  )
ggraph(net.tidy8, layout = "graphopt") +
  geom_edge_link(width = 0.5, colour = "lightgray") +
  geom_node_point(size = 1.5, colour = "#00AFBB") +
  geom_node_text(aes(label = label), repel = TRUE)+
  theme_graph()
```



Make an interactive graph of one year (internal members) with networkD3.

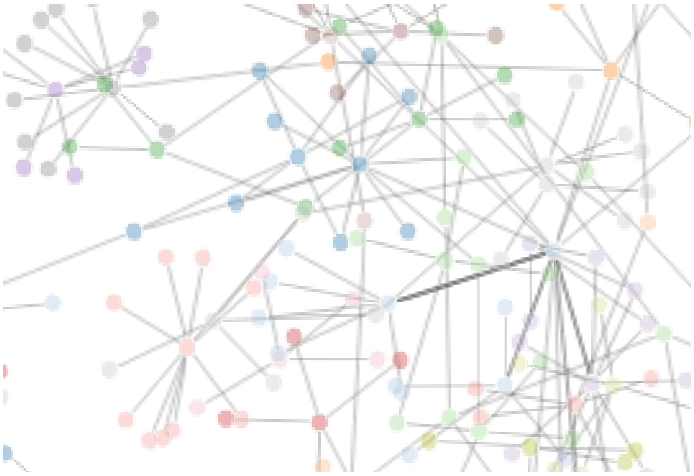
```
#subset interne 2017 to select n_poste and ID
interne_2017_2 = interne_2017 %>% select(n_poste, ID)
head(interne_2017_2)
```

```
## # A tibble: 6 x 2
##   n_poste ID
##     <dbl> <chr>
## 1     193 80526470
## 2    4314 35587865
## 3    4314 <NA>
## 4      54 130387304
## 5     193 <NA>
## 6    4380 7957047X
```

```
#create igraph ibject
graph_interne_2017 <- graph_from_data_frame(interne_2017_2, directed = FALSE)
V(graph_interne_2017)$label <- V(graph_interne_2017)$name
```

```
# Use igraph to make the graph and find membership
wc <- cluster_walktrap(graph_interne_2017)
members <- membership(wc)
#create graph object for network D3
graph_interne_2017_d3 <- igraph_to_networkD3(graph_interne_2017, group = members)
#head(graph_interne_2017_d3)
```

```
# Create force directed network plot
forceNetwork(
Links = graph_interne_2017_d3$links,
Nodes = graph_interne_2017_d3$nodes,
Source = 'source',
Target = 'target',
NodeID = 'name',
Group = 'group'
)
```



Plot a network graph where the vertices are colored by levels of the juries(additional i.e not in documentation)

```
#filter for 2017
interne_2017 = df %>% filter(status == 'interne' & year == 2017)
externe_2017 = df %>% filter(status == 'externe' & year == 2017)
#recode ?MCF to MCF and PUPH to PU-PH
externe_2017$Level[externe_2017$Level == "?MCF"]  <- "MCF"
externe_2017$Level[externe_2017$Level == "PUPH"]  <- "PU-PH"
interne_2017$Level[interne_2017$Level == "?MCF"]  <- "MCF"
interne_2017$Level[interne_2017$Level == "PUPH"]  <- "PU-PH"
```

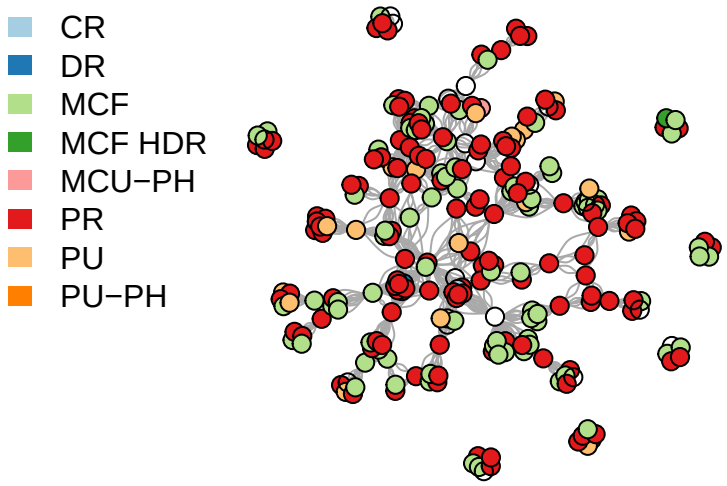
Checking for external 2017 and coloring by level

```r
#self join to create edge list for external 2017
externe_2017_edge_list_name <- externe_2017 %>% select(Level, n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(externe_2017_edge_list_name)
```

```
## # A tibble: 6 x 4
##   Level n_poste Name1             Name2
##   <chr>   <dbl> <chr>             <chr>
## 1 MCF        13 Caroline DESOMBRE Claire PERRIN
## 2 MCF        13 Caroline DESOMBRE Cyril CROZET
## 3 MCF        13 Caroline DESOMBRE Dominique BERGER
## 4 MCF        13 Caroline DESOMBRE Jeanine POMMIER
## 5 MCF        13 Caroline DESOMBRE Thierry PIOT
## 6 MCF        13 Claire PERRIN     Caroline DESOMBRE
```

```r
externe_2017_matrix_name <- as.matrix(externe_2017_edge_list_name[c('Name1', 'Name2')])
```

```r
set.seed(123)


externe_2017$Level <- as.factor(externe_2017$Level)
#generate palette of 10 colours

pal <- brewer.pal(n = 10, name = 'Paired')

vertex.col <- pal[(externe_2017$Level)]


#plot external 2017 graph colored by level of jury
graph_externe_2017 <- graph_from_edgelist(externe_2017_matrix_name, directed = FALSE)



plot(graph_externe_2017,
     vertex.label = NA, vertex.size = 8, vertex.color = vertex.col)

legend("topleft",bty = "n",
       legend=levels((externe_2017$Level)),
       fill=pal, border=NA)
```



Color by institutions

```
#check how many institutions there are
length(unique(externe_2017$institutions))
```

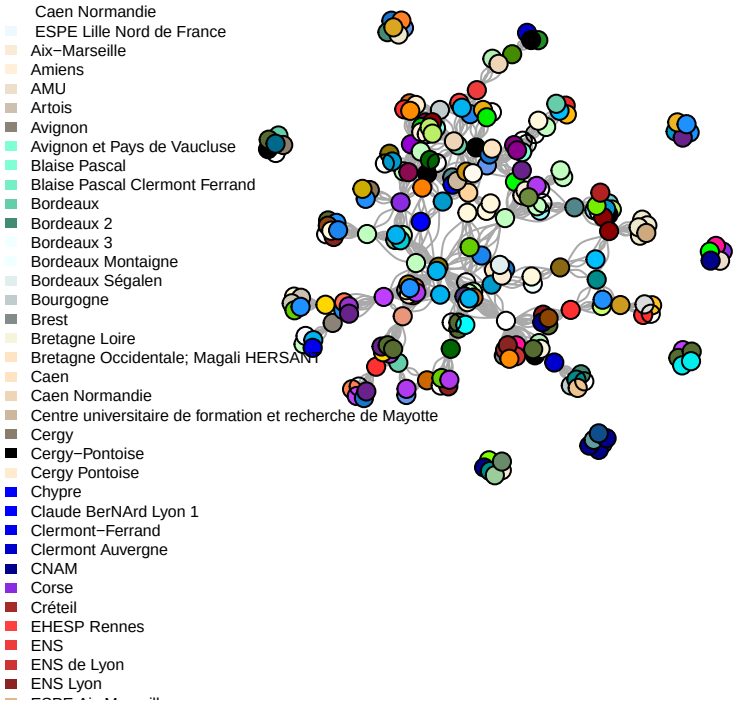
```
## [1] 148
```

```
#self join to create external graph but this time with institutions
externe_2017_edge_list_name <- externe_2017 %>% select(institutions, n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(externe_2017_edge_list_name)
```

```
## # A tibble: 6 x 4
##   institutions n_poste Name1             Name2
##   <chr>          <dbl> <chr>             <chr>
## 1 ESPE Lille        13 Caroline DESOMBRE Claire PERRIN
## 2 ESPE Lille        13 Caroline DESOMBRE Cyril CROZET
## 3 ESPE Lille        13 Caroline DESOMBRE Dominique BERGER
## 4 ESPE Lille        13 Caroline DESOMBRE Jeanine POMMIER
## 5 ESPE Lille        13 Caroline DESOMBRE Thierry PIOT
## 6 Lyon 1            13 Claire PERRIN     Caroline DESOMBRE
```

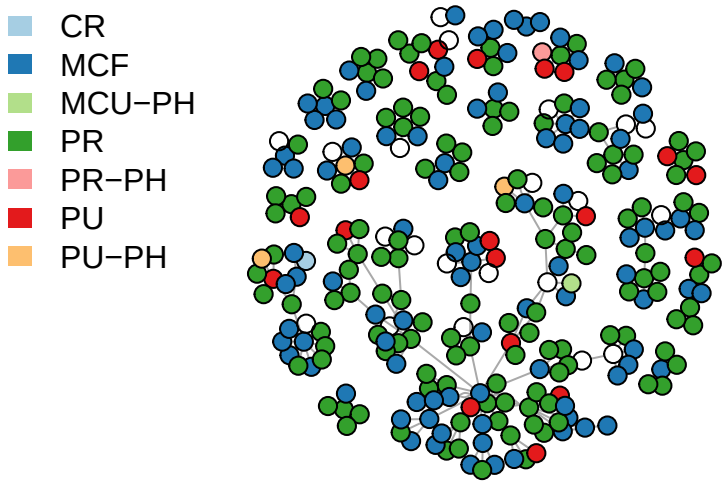
```
externe_2017_matrix_name <- as.matrix(externe_2017_edge_list_name[c('Name1', 'Name2')])
```

```
set.seed(123)


externe_2017$institutions <- as.factor(externe_2017$institutions)
#generate palette of colors
pal = grDevices::colors()[grep('gr(a|e)y', grDevices::colors(), invert = T)]

#pal <- brewer.pal(n = 10, name = 'Paired')
vertex.col <- pal[(externe_2017$institutions)]



graph_externe_2017 <- graph_from_edgelist(externe_2017_matrix_name, directed = FALSE)

#plot external jury members for 2017 colored by institutions

plot(graph_externe_2017,
     vertex.label = NA, vertex.size = 8, vertex.color = vertex.col)

legend("topleft",bty = "n",
       legend=levels((externe_2017$institutions)),
       fill=pal, border=NA,cex=0.5)
```



- A lot of links » since juries travel/move from different institutions to select candidates

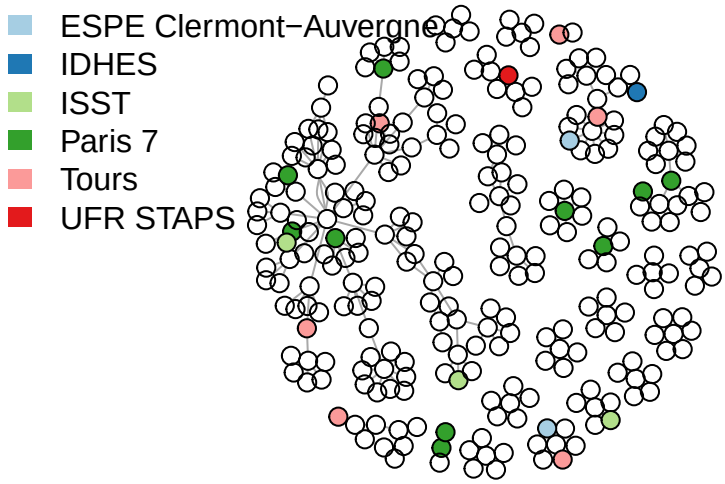
```
set.seed(123)
pal <- brewer.pal(n = 10, name = 'Paired')
interne_2017$Level <- as.factor(interne_2017$Level)


#internal 2017 colored by level
vertex.col <- pal[(interne_2017$Level)]

plot(graph_interne_2017,
     vertex.label = NA, vertex.size = 8, vertex.color = vertex.col)

legend("topleft",bty = "n",
       legend=levels(interne_2017$Level),
       fill=pal, border=NA,cex=1)
```



- Some networks are uncolored because the rows have N/A as institutions - Each small network represent an institution or institutions with their partners -(From the figure colored by level), we can see that in each small network there is atleast one PR(Professeurs des universités),each small network represent an institution and if there are multiple PR , it could be different departments of an institution or partner institutions

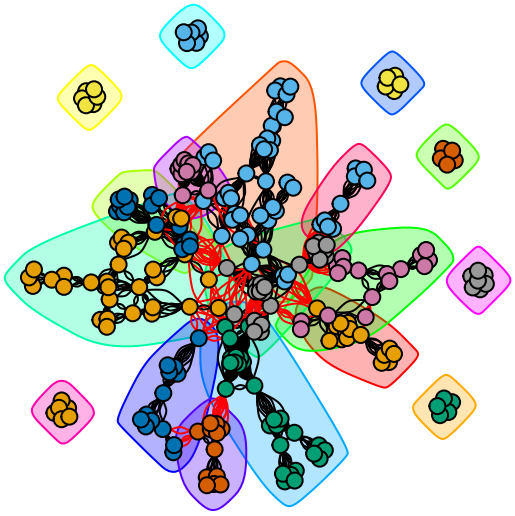
```
pal <- brewer.pal(n = 10, name = 'Paired')

interne_2017$institutions <- as.factor(interne_2017$institutions)

# interne_2017$Level[interne_2017$Level == "?MCF"]  <- "MCF"
```

```r
# interne_2017$Level[interne_2017$Level == "PUPH"]  <- "PU-PH"

#internal 2017 colored by institutions

vertex.col <- pal[(interne_2017$institutions)]

plot(graph_interne_2017,
     vertex.label = NA, vertex.size = 8, vertex.color = vertex.col)

legend("topleft",bty = "n",
       legend=levels(interne_2017$institutions),
       fill=pal, border=NA)
```

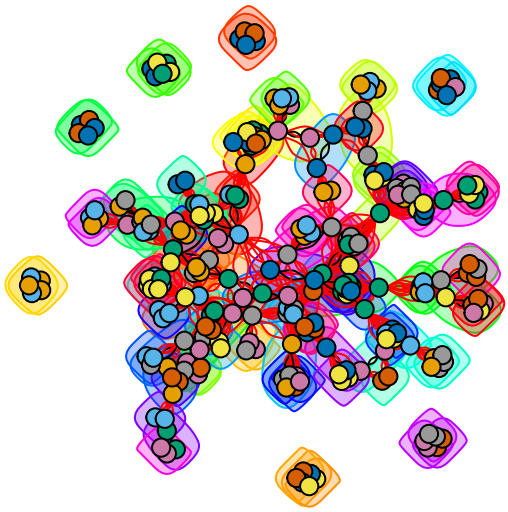


Apply a community detection approach to detect communities in one of the graphs with only external members

```r
# Community Detection ------------------------------------------------------
# Louvain
#apply community detection on external members 2017
lc_externe_2017 <- cluster_louvain(graph_externe_2017)
# membership(lc_externe_2017)
# communities(lc_externe_2017)
plot(lc_externe_2017, graph_externe_2017, vertex.label = NA, vertex.size = 7)
```



```
#infomap method - community detection
imc_externe_2017 <- cluster_infomap(graph_externe_2017)

plot(imc_externe_2017, graph_externe_2017, vertex.label = NA, vertex.size = 8)
```

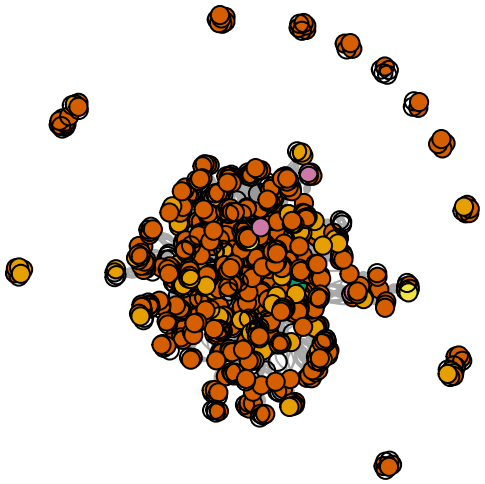


Create a last graph where all years are pooled together, but only for external members. What is the problem with this graph ?

```
#filter df for externe only
externe = df %>% filter(status == "externe")
#self join for externe
externe_edge_list_name <- externe %>% select(n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

head(externe_edge_list_name)
```

```
## # A tibble: 6 x 3
##   n_poste Name1             Name2
##     <dbl> <chr>             <chr>
## 1      13 Caroline DESOMBRE Claire PERRIN
## 2      13 Caroline DESOMBRE Cyril CROZET
## 3      13 Caroline DESOMBRE Dominique BERGER
## 4      13 Caroline DESOMBRE Jeanine POMMIER
## 5      13 Caroline DESOMBRE Thierry PIOT
## 6      13 Claire PERRIN     Caroline DESOMBRE
```

```r
#create matrix from dataframe
externe_matrix_name <- as.matrix(externe_edge_list_name[c('Name1', 'Name2')])
#color by level
externe$Level <- as.factor(externe$Level)
graph_externe <- graph_from_edgelist(externe_matrix_name, directed = FALSE)
plot(graph_externe, vertex.label = NA, vertex.size = 8, vertex.color = externe$Level)
```



Centrality measures

Choose the relevant dataset to compute the following indicators. Identify the top ten members (create a table per indicator) with the highest values for the following indicators : degree, strength, betweenness.(https://dshizuka.github.io/networkanalysis/04_measuring.html) https://medium.com/@615162020004/social-network-analysis-with-r-centrality-measure-86d7fa273574

Create the plots but this time, the width of the nodes should depend upon the value of the centrality indicators.

```r
#create sub dataframe for external 2017
externe_2017_2 = externe_2017 %>% select(n_poste, Name)
head(externe_2017_2)
```

```
## # A tibble: 6 x 2
##   n_poste Name
##     <dbl> <chr>
```

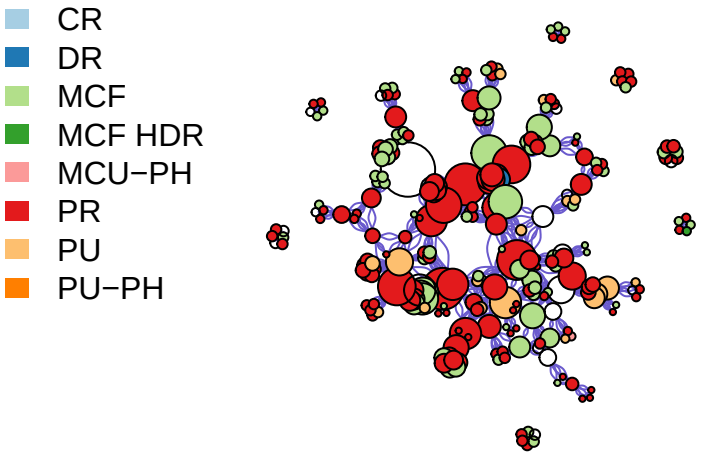
```
## 1    4283 Abdelkarim ZAID
## 2     544 Alain JEAN
## 3    4061 Alain VULBEAU
## 4    4372 Alain VULBEAU
## 5    4265 Anne-Claire Husser
## 6     901 Anne BARRERE
```

```
#get adjacency matrix from edge list
graph_2017_ex <- graph_from_edgelist(as.matrix(externe_2017_edge_list_name[c("Name1","Name2")]), directe
```

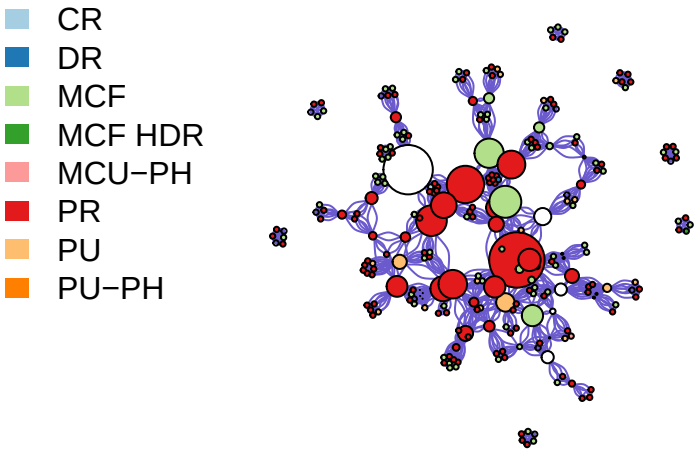
```
#show igraph object
graph_2017_ex
```

```
## IGRAPH d8e12ac UN-- 253 1744 --
## + attr: name (v/c)
## + edges from d8e12ac (vertex names):
##  [1] Caroline DESOMBRE--Claire PERRIN    Caroline DESOMBRE--Cyril CROZET
##  [3] Caroline DESOMBRE--Dominique BERGER Caroline DESOMBRE--Jeanine POMMIER
##  [5] Caroline DESOMBRE--Thierry PIOT     Caroline DESOMBRE--Claire PERRIN
##  [7] Claire PERRIN    --Cyril CROZET     Claire PERRIN    --Dominique BERGER
##  [9] Claire PERRIN    --Jeanine POMMIER  Claire PERRIN    --Thierry PIOT
## [11] Caroline DESOMBRE--Cyril CROZET     Claire PERRIN    --Cyril CROZET
## [13] Cyril CROZET     --Dominique BERGER Cyril CROZET     --Jeanine POMMIER
## [15] Cyril CROZET     --Thierry PIOT     Caroline DESOMBRE--Dominique BERGER
## + ... omitted several edges
```

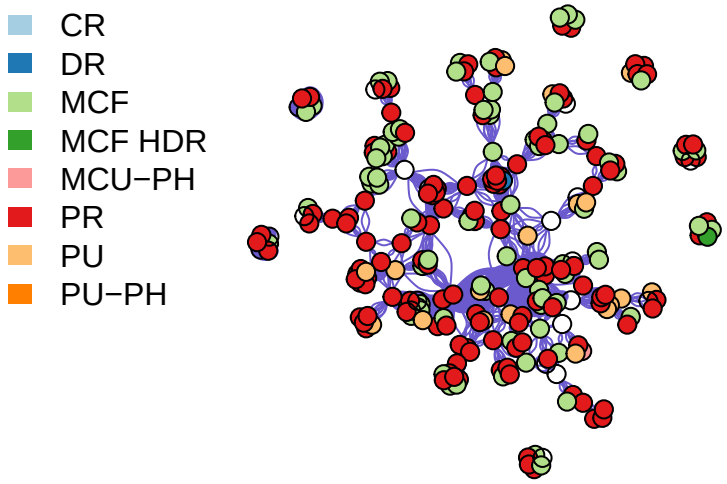
```
#graph - node size depends on degree centrality
set.seed(10)


externe_2017$Level <- as.factor(externe_2017$Level)

pal <- brewer.pal(n = 10, name = 'Paired')

vertex.col <- pal[(externe_2017$Level)]
#calculate degree
de=igraph::degree(graph_2017_ex)
plot(graph_2017_ex, vertex.label="", vertex.color=vertex.col, edge.color="slateblue", vertex.size=de*0.5
legend("topleft",bty = "n",
       legend=levels((externe_2017$Level)),
       fill=pal, border=NA,cex=1)
```




```
set.seed(10)

#graph - node size depends on betweeness centrality colored by levels


externe_2017$Level <- as.factor(externe_2017$Level)

pal <- brewer.pal(n = 10, name = 'Paired')

vertex.col <- pal[(externe_2017$Level)]

be=betweenness(graph_2017_ex, normalized=T)

plot(graph_2017_ex, vertex.label="", vertex.color=vertex.col, edge.color="slateblue", vertex.size=be*150

legend("topleft",bty = "n",
       legend=levels((externe_2017$Level)),
       fill=pal, border=NA,cex=1)
```




```
set.seed(10)


#graph - edge width depends onstrength centrality

externe_2017$Level <- as.factor(externe_2017$Level)

pal <- brewer.pal(n = 10, name = 'Paired')

vertex.col <- pal[(externe_2017$Level)]

st=graph.strength(graph_2017_ex)

plot(graph_2017_ex, vertex.label="", vertex.color=vertex.col, edge.color="slateblue", vertex.size=8,edg

legend("topleft",bty = "n",
       legend=levels((externe_2017$Level)),
       fill=pal, border=NA)
```




```
#Table for top 10 centrality
#sort dataframe by 10 highest degree
df_deg <- as.data.frame(de) %>% arrange(desc(as.data.frame(de)))
#add name column
df_deg$Name <- rownames(df_deg)
#rename rownames to empty
rownames(df_deg) <- c()
df_deg <- head(df_deg,10)
df_deg
```

```
##    de                           Name
## 1  52 Denise ORANGE RAVACHOL RAVACHOL
## 2  40               Caroline DESOMBRE
## 3  40         Fabienne BRIERE-GUENOUN
## 4  38                Dominique BERGER
## 5  36                 Nicole BIAGIOLI
## 6  36                   Bruno GARNIER
## 7  34                    Eric FLAVIER
## 8  34                   Pierre Périer
## 9  32              Nassira HEDJERASSI
## 10 30                Line NUMA BOCAGE
```

```
#merge degree dataframe to get institution and level

df_merged1<-merge(x = df_deg, y = externe_2017[, c("institutions", "Level","Name")], by = "Name", all.x=
```

```r
head(df_merged1[!duplicated(df_merged1),])
```

```
##                               Name de               institutions Level
## 1                   Bruno GARNIER 36                      Corse    PR
## 3               Caroline DESOMBRE 40                  ESPE Lille   MCF
## 4               Caroline DESOMBRE 40  ESPE Lille Nord de France   MCF
## 5               Caroline DESOMBRE 40  ESPE Lille Nord de France   MCF
## 6 Denise ORANGE RAVACHOL RAVACHOL 52                    Lille 3    PR
## 7 Denise ORANGE RAVACHOL RAVACHOL 52                    Lille 3    PU
```

```r
#Table for top 10 centrality
#sort dataframe by 10 highest degree
df_bet <- as.data.frame(be) %>% arrange(desc(as.data.frame(be)))
#add name column
df_bet$Name <- rownames(df_bet)
#rename rownames to empty
rownames(df_bet) <- c()
df_bet <- head(df_bet,10)
df_bet
```

```
##            be                            Name
## 1  0.17694863                Dominique BERGER
## 2  0.15771004 Denise ORANGE RAVACHOL RAVACHOL
## 3  0.11932955         Fabienne BRIERE-GUENOUN
## 4  0.10210809              Nassira HEDJERASSI
## 5  0.09813727                 Martine JAUBERT
## 6  0.09343578                    Eric FLAVIER
## 7  0.09125400           Pascale BRANDT POMARES
## 8  0.08900287                   Bruno GARNIER
## 9  0.08197679                   Pierre Périer
## 10 0.08003937                Caroline DESOMBRE
```

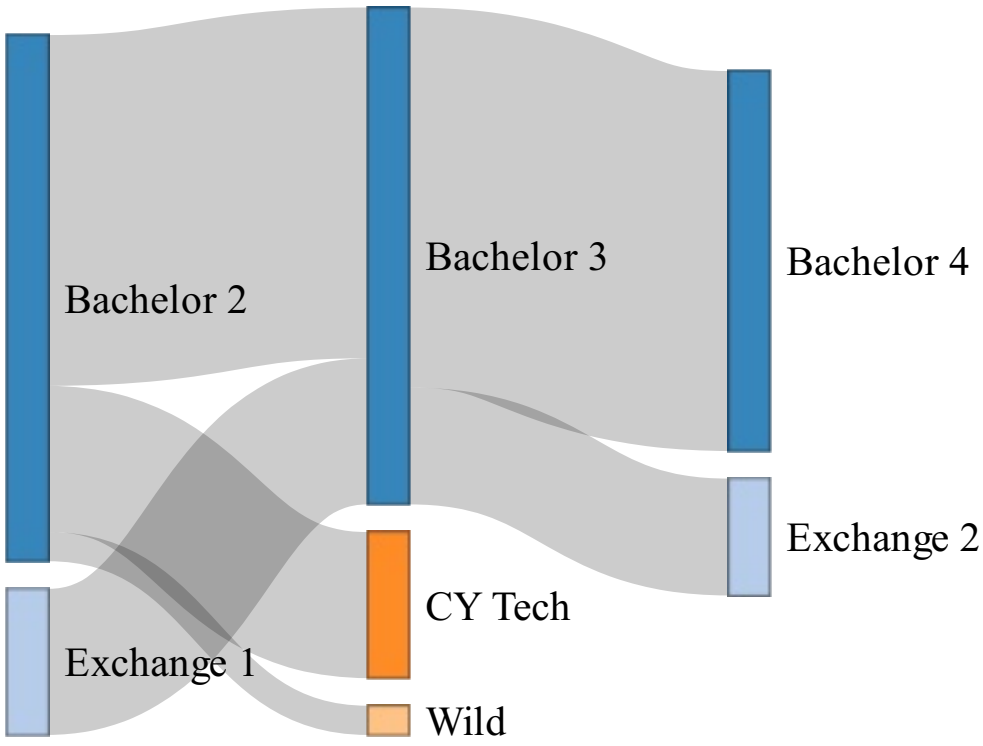
```r
#merge degree dataframe to get institution and level

df_merged2<-merge(x = df_bet, y = externe_2017[, c("institutions", "Level","Name")], by = "Name", all.x=

head(df_merged2[!duplicated(df_merged2),])
```

```
##                               Name         be               institutions Level
## 1                   Bruno GARNIER 0.08900287                      Corse    PR
## 3               Caroline DESOMBRE 0.08003937                  ESPE Lille   MCF
## 4               Caroline DESOMBRE 0.08003937  ESPE Lille Nord de France   MCF
## 5               Caroline DESOMBRE 0.08003937  ESPE Lille Nord de France   MCF
## 6 Denise ORANGE RAVACHOL RAVACHOL 0.15771004                    Lille 3    PR
## 9 Denise ORANGE RAVACHOL RAVACHOL 0.15771004                    Lille 3    PU
```

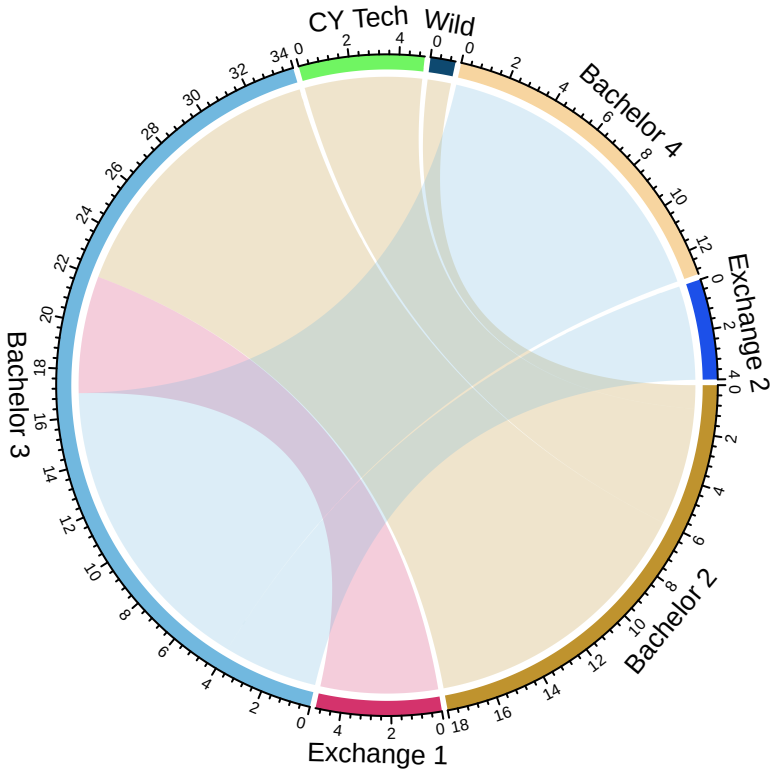
Sankey diagram and chord plot

```r
links <- data.frame(
source = c("Bachelor 2", "Bachelor 2", "Bachelor 2", "Exchange 1", "Bachelor 3", "Bachelor 3"),
target = c("Bachelor 3", "CY Tech", "Wild", "Bachelor 3", "Bachelor 4", "Exchange 2"),
value = c(12, 5, 1, 5, 13, 4)
)



nodes <- data.frame(name = c(as.character(links$source), as.character(links$target)) %>% unique())



links$ID.source <- match(links$source, nodes$name) - 1
links$ID.target <- match(links$target, nodes$name) - 1
```

```r
p <- sankeyNetwork(Links = links, Nodes = nodes, Source = "ID.source", Target = "ID.target", Value = "va



p
```



```
temp_graph <- graph.data.frame(links)
adjaceny.matrix <- get.adjacency(temp_graph, sparse = FALSE, attr= "value")



chordDiagram(adjaceny.matrix, transparency = 0.75)
```



In [39]:

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import numpy as np
import matplotlib.pyplot as plt
from itertools import permutations
import seaborn as sns

from pandas.plotting import parallel_coordinates

```

In [2]:

```
# !pip install mlxtend
#https://goldinlocks.github.io/Market-Basket-Analysis-in-Python/
```

In [10]:

```
df = pd.read_csv("tel_samp_rec.csv",encoding="latin-1")
```

In [11]:

```
df.head()
```

Out[11]:

	Defence.date	Domains	Full.Text.Language	def.date	n.disc	these.id	disc1.lev1	
0	2010/09/23	Sciences du Vivant [q-bio] / Ecologie, Environ...	French	2010.0	1	tel-00662843v1	Sciences du Vivant [q-bio]	Env
1	2009/11/02	Sciences de l'Homme et Société	French	2009.0	1	tel-00491490v1	Sciences de l'Homme et Société	
2	1996/05/30	Sciences du Vivant [q-bio] / Alimentation et N...	French	1996.0	1	tel-01776364v1	Sciences du Vivant [q-bio]	Alin
3	2018/02/02	Informatique [cs] / Autre [cs.OH] \r\n\r\nInf...	French	2018.0	1	tel-02437294v1	Informatique [cs]	Au
4	2015/07/08	Informatique [cs] / Automatique \r\n\r\nInfor...	French	2015.0	1	tel-01245100v1	Informatique [cs]	A

5 rows × 25 columns

In [12]:

```
cols = ['disc1.rec.lev1','disc2.rec.lev1','disc3.rec.lev1']
#subset columns shown above and take columns where all the 3 columns are not null
df_sub = df[df[cols].notnull().all(axis=1)]
```

In [13]:

```
df_sub.head()
```

Out[13]:

	Defence.date	Domains	Full.Text.Language	def.date	n.disc	these.id	disc1.lev1	dis
53	1985/10/28	Planète et Univers [physics] / Sciences de la ...	French	1985.0	2	tel-00711880v1	Planète et Univers [physics]	S de
104	2018/12/17	Sciences de l'ingénieur [physics] / Génie civi...	English	2018.0	2	tel-02182014v1	Sciences de l'ingénieur [physics]	Gé
113	2003/06/17	Sciences de l'ingénieur [physics] / Traitement...	French	2003.0	3	tel-00130932v1	Sciences de l'ingénieur [physics]	Tra d [e
193	1997/10/24	Planète et Univers [physics] / Sciences de la ...	French	1997.0	2	tel-00675418v1	Planète et Univers [physics]	S de
212	2002/12/13	Sciences du Vivant [q-bio] / Autre [q-bio.OT] ...	French	2002.0	2	tel-00008546v1	Sciences du Vivant [q-bio]	A

5 rows × 25 columns

In [14]:

```
df_sub = df_sub[cols]
```

In [15]:

```
df_sub.head()
```

Out[15]:

	disc1.rec.lev1	disc2.rec.lev1	disc3.rec.lev1
53	VIII	VIII	VIII
104	IX	IX	V
113	IX	VI	V
193	VIII	VIII	VIII
212	X	IX	IX

In [16]:

```
# Getting the list of transactions from the dataset
transactions = []
for i in range(0, len(df_sub)):
    transactions.append([str(df_sub.values[i,j]) for j in range(0, len(df_sub.columns))
```

In [17]:

```
#check transactions
transactions[:1]

```

Out[17]:

```
[['VIII', 'VIII', 'VIII']]
```

In [19]:

```

# Extract unique items.
flattened = [item for transaction in transactions for item in transaction]
items = list(set(flattened))
```

In [20]:

```
print('# of items:',len(items))
print(list(items))
```

```
# of items: 13
['IV', 'pharmacie', 'VI', 'I', 'III', 'XII', 'II', 'VIII', 'X', 'V', 'I - Dr
oit', 'VII', 'IX']
```

In [21]:

```
#remove nan if present in list
if 'nan' in items: items.remove('nan')
print(list(items))
```

```
['IV', 'pharmacie', 'VI', 'I', 'III', 'XII', 'II', 'VIII', 'X', 'V', 'I - Dr
oit', 'VII', 'IX']
```

In [22]:

```
# Compute and print rules.
rules = list(permutations(items, 2))
print('# of rules:',len(rules))
print(rules[:5])
```

```
# of rules: 156
[('IV', 'pharmacie'), ('IV', 'VI'), ('IV', 'I'), ('IV', 'III'), ('IV', 'XI
I')]
```

In [23]:

```
# Import the transaction encoder function from mlxtend
from mlxtend.preprocessing import TransactionEncoder

# Instantiate transaction encoder and identify unique items
encoder = TransactionEncoder().fit(transactions)

# One-hot encode transactions
onehot = encoder.transform(transactions)

# Convert one-hot encoded data to DataFrame
onehot = pd.DataFrame(onehot, columns = encoder.columns_)

# Print the one-hot encoded transaction dataset
onehot.head()
```

Out[23]:

	I	I - Droit	II	III	IV	IX	V	VI	VII	VIII	X	XII	pharmacie
0	False	False	False	False	False	False	False	False	False	True	False	False	False
1	False	False	False	False	False	True	True	False	False	False	False	False	False
2	False	False	False	False	False	True	True	True	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	True	False	False	False
4	False	False	False	False	False	True	False	False	False	False	True	False	False

In [24]:

```
def leverage(antecedent, consequent):
    # Compute support for antecedent AND consequent
    supportAB = np.logical_and(antecedent, consequent).mean()

    # Compute support for antecedent
    supportA = antecedent.mean()

    # Compute support for consequent
    supportB = consequent.mean()

    # Return leverage
    return supportAB - supportB * supportA

# Define a function to compute Zhang's metric
def zhang(antecedent, consequent):
    # Compute the support of each book
    supportA = antecedent.mean()
    supportC = consequent.mean()

    # Compute the support of both books
    supportAC = np.logical_and(antecedent, consequent).mean()

    # Complete the expressions for the numerator and denominator
    numerator = supportAC - supportA*supportC
    denominator = max(supportAC*(1-supportA), supportA*(supportC-supportAC))

    # Return Zhang's metric
    return numerator / denominator

def conviction(antecedent, consequent):
    # Compute support for antecedent AND consequent
    supportAC = np.logical_and(antecedent, consequent).mean()

    # Compute support for antecedent
    supportA = antecedent.mean()

    # Compute support for NOT consequent
    supportnC = 1.0 - consequent.mean()

    # Compute support for antecedent and NOT consequent
    supportAnC = supportA - supportAC

    # Return conviction
    return supportA * supportnC / supportAnC

```

In [25]:

```
# Create rules DataFrame
rules_ = pd.DataFrame(rules, columns=['antecedents','consequents'])

# Define an empty list for metrics
zhangs, conv, lev, antec_supp, cons_supp, suppt, conf, lft = [], [], [], [], [], [], [], []

# Loop over lists in itemsets
for itemset in rules:
    # Extract the antecedent and consequent columns
    antecedent = onehot[itemset[0]]
    consequent = onehot[itemset[1]]

    antecedent_support = onehot[itemset[0]].mean()
    consequent_support = onehot[itemset[1]].mean()
    support = np.logical_and(onehot[itemset[0]], onehot[itemset[1]]).mean()
    confidence = support / antecedent_support
    lift = support / (antecedent_support * consequent_support)

    # Complete metrics and append it to the list
    antec_supp.append(antecedent_support)
    cons_supp.append(consequent_support)
    suppt.append(support)
    conf.append(confidence)
    lft.append(lift)
    lev.append(leverage(antecedent, consequent))
    conv.append(conviction(antecedent, consequent))
    zhangs.append(zhang(antecedent, consequent))

# Store results
rules_['antecedent support'] = antec_supp
rules_['consequent support'] = cons_supp
rules_['support'] = suppt
rules_['confidence'] = conf
rules_['lift'] = lft
rules_['leverage'] = lev
rules_['conviction'] = conv
rules_['zhang'] = zhangs

# Print results
rules_.sort_values('zhang',ascending=False).head()
```

Out[25]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
2	IV	I	0.143959	0.021994	0.020566	0.142857	6.495362	0.01
9	IV	I - Droit	0.143959	0.004856	0.003999	0.027778	5.720588	0.00
97	X	pharmacie	0.199372	0.019709	0.016281	0.081662	4.143453	0.01
45	I	I - Droit	0.021994	0.004856	0.001428	0.064935	13.372804	0.00
123	I - Droit	I	0.004856	0.021994	0.001428	0.294118	13.372804	0.00

In [26]:

```
# Function to convert rules to coordinates.
def rules_to_coordinates(rules):
    rules['antecedent'] = rules['antecedents'].apply(lambda antecedent: list(antecedent
    rules['consequent'] = rules['consequents'].apply(lambda consequent: list(consequent
    rules['rule'] = rules.index
    return rules[['antecedent','consequent','rule']]
```

In [ ]:

```

```

In [32]:

```
rules_.head()
```

Out[32]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
0	IV	pharmacie	0.143959	0.019709	0.000857	0.005952	0.302019	-0.00198
1	IV	VI	0.143959	0.162239	0.000857	0.005952	0.036689	-0.02249
2	IV	I	0.143959	0.021994	0.020566	0.142857	6.495362	0.01739
3	IV	III	0.143959	0.032562	0.022279	0.154762	4.752820	0.01759
4	IV	XII	0.143959	0.049414	0.034276	0.238095	4.818332	0.02716

In [33]:

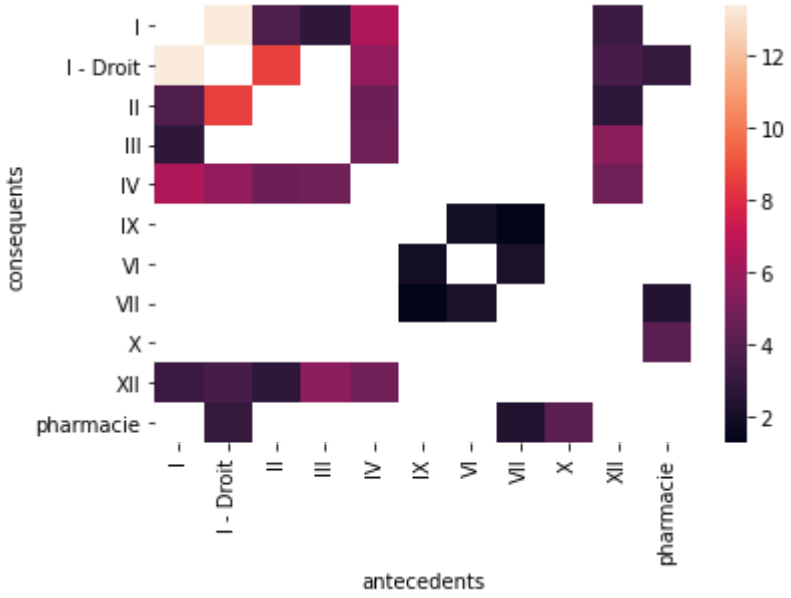
```
#remove rows where antecedent = consequent
rules_ = rules_[rules_['antecedents'] != rules_['consequents']]
```

In [35]:

```
#filter rules with lift > 1
rules_ = rules_.query("lift>1")
#create support table based on lift values > 1
support_table = rules_.pivot(index='consequents', columns='antecedents',
values='lift')



sns.heatmap(support_table)
```

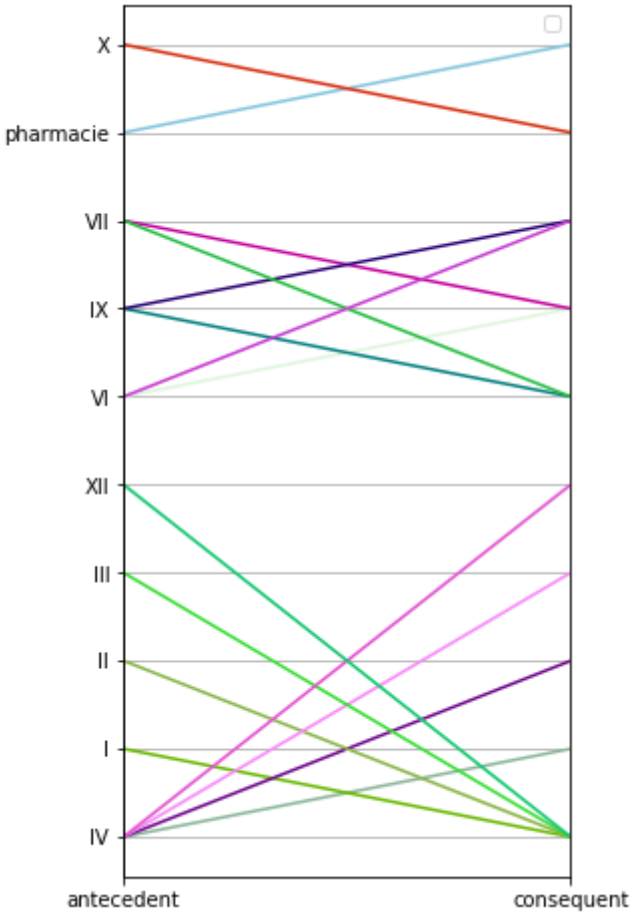
Out[35]:

```
<AxesSubplot:xlabel='antecedents', ylabel='consequents'>
```



In [41]:

```
# Generate frequent itemsets
frequent_itemsets = apriori(onehot, min_support = 0.01, use_colnames = True, max_len =
# Generate association rules
rules = association_rules(frequent_itemsets, metric = 'lift', min_threshold = 1.00)
# Generate coordinates and print example
coords = rules_to_coordinates(rules)
# Generate parallel coordinates plot

plt.figure(figsize=(4,8))
parallel_coordinates(coords, 'rule')
plt.legend([])
plt.grid(True)
plt.show()
```



https://www.galaxie.enseignementsup-recherche.gouv.fr/ensup/pdf/qualification/sections.pdf (https://www.galaxie.enseignementsup-recherche.gouv.fr/ensup/pdf/qualification/sections.pdf)

> LINK : Reference what I , II etc means