SNA

Anh Thu

11/30/2021

```
# Load library
library(readr)
library(igraph)
```

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

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

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

```
library(dplyr)
```

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

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

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

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

```
library(ggplot2)
library(gganimate)
library(networkD3)
library(gapminder)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(patchwork)
library(hrbrthemes)
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.

##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and

##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```
library(circlize)
```

```
## ========================================
## circlize version 0.4.13
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================

##
## Attaching package: 'circlize'

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

```
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:viridis':
##
##     viridis_pal

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
Auditions_db_comp <- read_csv("dataset/Auditions.db.comp.csv",locale = locale(encoding = "ISO-8859-1"))
```

```
## Rows: 2430 Columns: 12
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (8): Name, Level, Section, Role, status, institutions, ID, Id.author.no
## dbl (2): n_poste, year
## lgl (2): X, X.1

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

#Simple SNA ##Create a subset of the dataset

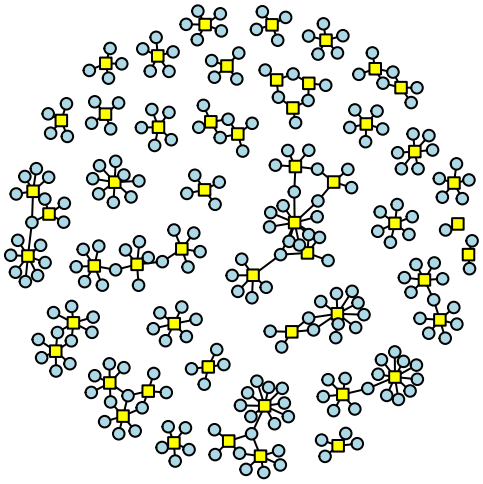
```
#2017
inter17 <- Auditions_db_comp[(Auditions_db_comp$year ==2017) & (Auditions_db_comp$status =="interne"),
exter17 <- Auditions_db_comp[(Auditions_db_comp$year ==2017) & (Auditions_db_comp$status =="externe"),

#2018
inter18 <- Auditions_db_comp[(Auditions_db_comp$year ==2018) & (Auditions_db_comp$status =="interne"),
exter18 <- Auditions_db_comp[(Auditions_db_comp$year ==2018) & (Auditions_db_comp$status =="externe"),

#2019
inter19 <- Auditions_db_comp[(Auditions_db_comp$year ==2019) & (Auditions_db_comp$status =="interne"),
exter19 <- Auditions_db_comp[(Auditions_db_comp$year ==2019) & (Auditions_db_comp$status =="externe"),

#2020
inter20 <- Auditions_db_comp[(Auditions_db_comp$year ==2020) & (Auditions_db_comp$status =="interne"),
exter20 <- Auditions_db_comp[(Auditions_db_comp$year ==2020) & (Auditions_db_comp$status =="externe"),
```

##Bipartite 2017

```
inter17_g <- inter17 %>% select(n_poste, Name)
# g_i17 <- graph.data.frame(inter17_g, directed = FALSE)
# plot(g_i17, vertex.label = NA, vertex.size = 5, vertex.color = 'red')
```

```
i17_g <- graph.data.frame(inter17_g, directed=FALSE)
i17_b <- bipartite.mapping(i17_g)
V(i17_g)$type <- bipartite_mapping(i17_g)$type
V(i17_g)$color <- ifelse(V(i17_g)$type, "lightblue", "yellow")
V(i17_g)$shape <- ifelse(V(i17_g)$type, "circle", "square")
E(i17_g)$color <- "black"
plot(i17_g,vertex.label = NA, vertex.size =5, vertex.size =5, main ='Internal Network in 2017')
```

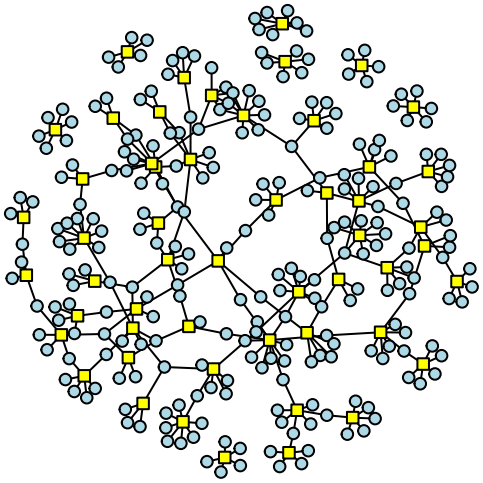
Internal Network in 2017



```
exter17_g <- exter17 %>% select(n_poste, Name)
# g_e17 <- graph.data.frame(exter17_g, directed = FALSE)
# plot(g_e17, vertex.label = NA, vertex.size = 5, vertex.color = 'red')
```

```
e17_g <- graph.data.frame(exter17_g, directed=FALSE)
e17_b <- bipartite.mapping(e17_g)
V(e17_g)$type <- bipartite_mapping(e17_g)$type
V(e17_g)$color <- ifelse(V(e17_g)$type, "lightblue", "yellow")
V(e17_g)$shape <- ifelse(V(e17_g)$type, "circle", "square")
E(e17_g)$color <- "black"
plot(e17_g,vertex.label = NA, vertex.size =5, main ='External Network in 2017')
```

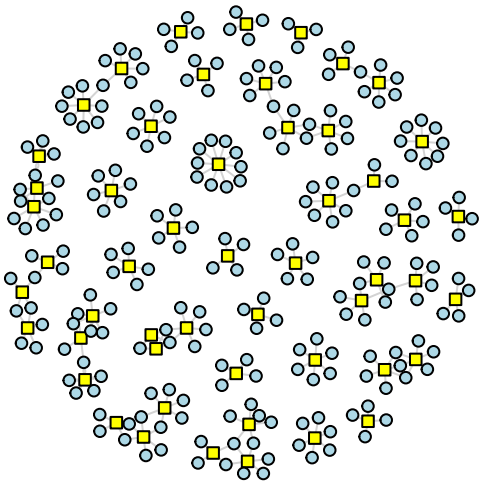
External Network in 2017



##Bipartite 2018

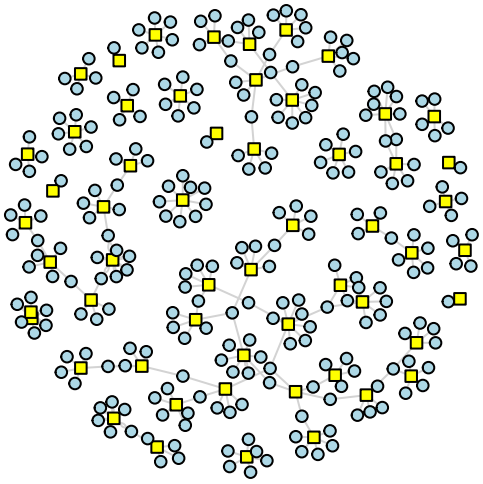
```
inter18_g <- inter18 %>% select(n_poste, Name)
# g_i18 <- graph.data.frame(inter18_g, directed = FALSE)
# plot(g_i18, vertex.label = NA, vertex.size = 5, vertex.color = 'red')
```

```
i18_g <- graph.data.frame(inter18_g, directed=FALSE)
i18_b <- bipartite.mapping(i18_g)
V(i18_g)$type <- bipartite_mapping(i18_g)$type
V(i18_g)$color <- ifelse(V(i18_g)$type, "lightblue", "yellow")
V(i18_g)$shape <- ifelse(V(i18_g)$type, "circle", "square")
E(i18_g)$color <- "lightgray"
plot(i18_g,vertex.label = NA, vertex.size =5)
```



```
exter18_g <- exter18 %>% select(n_poste, Name)
# g_e18 <- graph.data.frame(exter18_g, directed = FALSE)
# plot(g_e18, vertex.label = NA, vertex.size = 5, vertex.color = 'red')
```

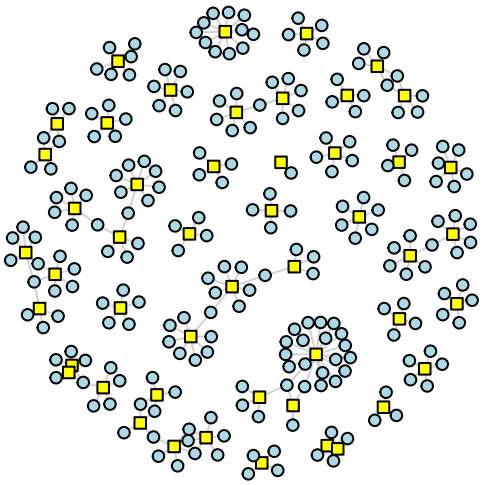
```
e18_g <- graph.data.frame(exter18_g, directed=FALSE)
e18_b <- bipartite.mapping(e18_g)
V(e18_g)$type <- bipartite_mapping(e18_g)$type
V(e18_g)$color <- ifelse(V(e18_g)$type, "lightblue", "yellow")
V(e18_g)$shape <- ifelse(V(e18_g)$type, "circle", "square")
E(e18_g)$color <- "lightgray"
plot(e18_g,vertex.label = NA, vertex.size =5)
```



##Bipartite 2019

```
inter19_g <- inter19 %>% select(n_poste, Name)
# g_i19 <- graph.data.frame(inter19_g, directed = FALSE)
# plot(g_i19, vertex.label = NA, vertex.size = 5, vertex.color = 'red')
```

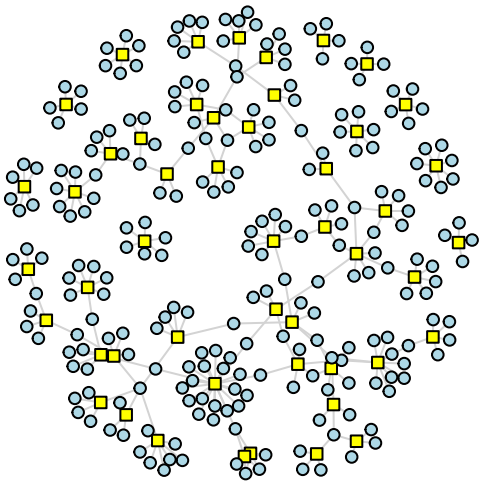
```
i19_g <- graph.data.frame(inter19_g, directed=FALSE)
i19_b <- bipartite.mapping(i19_g)
V(i19_g)$type <- bipartite_mapping(i19_g)$type
V(i19_g)$color <- ifelse(V(i19_g)$type, "lightblue", "yellow")
V(i19_g)$shape <- ifelse(V(i19_g)$type, "circle", "square")
E(i19_g)$color <- "lightgray"
plot(i19_g,vertex.label = NA, vertex.size =5)
```



```
exter19_g <- exter19 %>% select(n_poste, Name)
# g_e19 <- graph.data.frame(exter19_g, directed = FALSE)
# plot(g_e19, vertex.label = NA, vertex.size = 5, vertex.color = 'red')
```

```
e19_g <- graph.data.frame(exter19_g, directed=FALSE)
e19_b <- bipartite.mapping(e19_g)
V(e19_g)$type <- bipartite_mapping(e19_g)$type
```

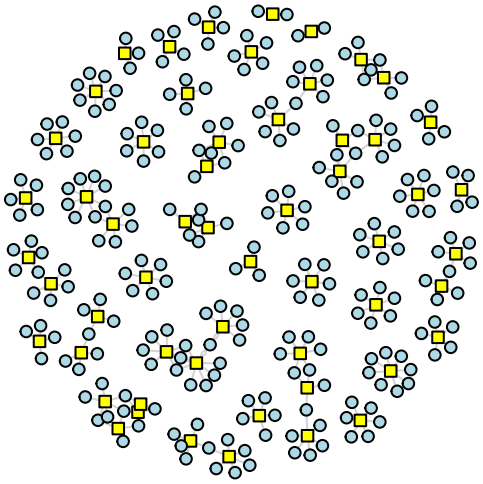
```
V(e19_g)$color <- ifelse(V(e19_g)$type, "lightblue", "yellow")
V(e19_g)$shape <- ifelse(V(e19_g)$type, "circle", "square")
E(e19_g)$color <- "lightgray"
plot(e19_g,vertex.label = NA, vertex.size =5)
```



##Bipartite 2020

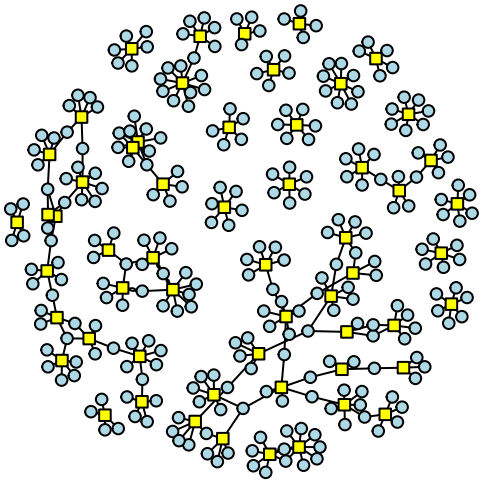
```
inter20_g <- inter20 %>% select(n_poste, Name)
# g_i20 <- graph.data.frame(inter20_g, directed = FALSE)
# plot(g_i20, vertex.label = NA, vertex.size = 5, vertex.color = 'red')
```

```
i20_g <- graph.data.frame(inter20_g, directed=FALSE)
i20_b <- bipartite.mapping(i20_g)
V(i20_g)$type <- bipartite_mapping(i20_g)$type
V(i20_g)$color <- ifelse(V(i20_g)$type, "lightblue", "yellow")
V(i20_g)$shape <- ifelse(V(i20_g)$type, "circle", "square")
E(i20_g)$color <- "lightgray"
plot(i20_g,vertex.label = NA, vertex.size =5)
```



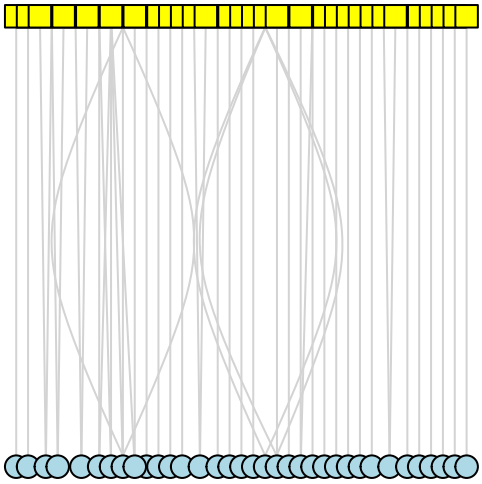
```
exter20_g <- exter20 %>% select(n_poste, Name)
# g_e20 <- graph.data.frame(exter20_g, directed = FALSE)
# col <- c("red", "yellow")
# plot(g_e20,vertex.label = NA, vertex.size = 5, vertex.color = col)
```

```
e20_g <- graph.data.frame(exter20_g, directed=FALSE)
e20_b <- bipartite.mapping(e20_g)
V(e20_g)$type <- bipartite_mapping(e20_g)$type
V(e20_g)$color <- ifelse(V(e20_g)$type, "lightblue", "yellow")
V(e20_g)$shape <- ifelse(V(e20_g)$type, "circle", "square")
E(e20_g)$color <- "black"
plot(e20_g, vertex.label = NA, vertex.size =5,edge.size =20)
```



```
b <- head(inter20, n=50)
```

```
b_g <- graph.data.frame(b, directed=FALSE)
b_b <- bipartite.mapping(b_g)
V(b_g)$type <- bipartite_mapping(b_g)$type
V(b_g)$color <- ifelse(V(b_g)$type, "lightblue", "yellow")
V(b_g)$shape <- ifelse(V(b_g)$type, "circle", "square")
E(b_g)$color <- "lightgray"
plot(b_g, layout=layout.bipartite, vertex.size=10, vertex.label= NA)
```



##Interactive graph

```
# Use igraph to make the graph and find membership
clwt <- cluster_walktrap(i17_g)
members <- membership(clwt)
# Convert to object suitable for networkD3
in17_d3 <- igraph_to_networkD3(i17_g, group = members)
# head(in17_d3)
```

```
# Create force directed network plot
forceNetwork(
Links = in17_d3$links,
Nodes = in17_d3$nodes,
Source = 'source',
Target = 'target',
NodeID = 'name',
Group = 'group'
)
```

```
exter17_edge_list <- exter17 %>% select(n_poste, ID) %>%
  inner_join(., select(., n_poste, ID), by = "n_poste") %>%
  rename(ID1 = ID.x, ID2 = ID.y) %>%
  filter(ID1 != ID2) %>%
  unique %>%
  arrange(n_poste)
head(exter17_edge_list)
```

```
## # A tibble: 6 x 3
##   n_poste ID1       ID2
##     <dbl> <chr>     <chr>
## 1      13 90444590  112729851
## 2      13 90444590  143374885
## 3      13 90444590  122818709
## 4      13 90444590  136684459
## 5      13 90444590  89759567
## 6      13 112729851 90444590
```

```
# Plot
simpleNetwork(exter17_edge_list[c('ID1', 'ID2')])  %>%
saveNetwork(file = 'graphs/exter17_edge_list_id.html')
```

```
exter17_edge_list_name <- exter17 %>% select(Level, n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
```

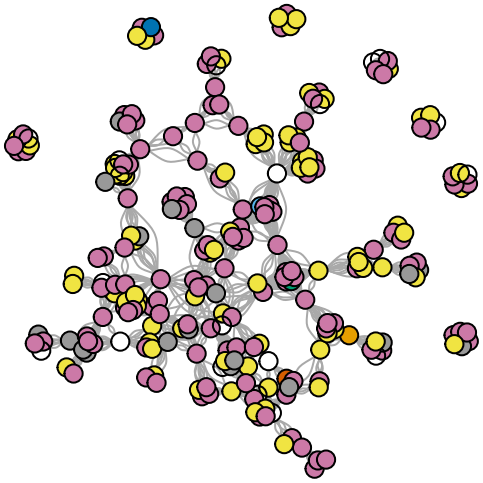
```
  unique %>%
  arrange(n_poste)
head(exter17_edge_list_name)
```

```
## # 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
```

```
# Plot
simpleNetwork(exter17_edge_list_name[c('Name1', 'Name2')])  %>%
saveNetwork(file = 'graphs/exter17_edge_list_name.html')
```

```
exter17_matrix_name <- as.matrix(exter17_edge_list_name[c('Name1', 'Name2')])
```

```
exter17$Level <- as.factor(exter17$Level)
g17 <- graph_from_edgelist(exter17_matrix_name, directed = FALSE)
plot(g17,
     vertex.label = NA, vertex.size = 8, vertex.color = exter17$Level)
```



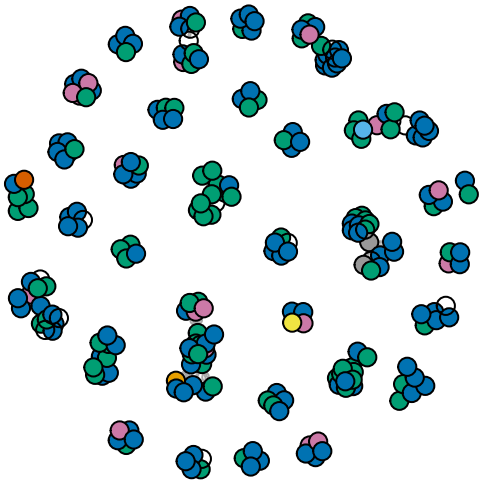
```
inter17_edge_list_name <- inter17 %>% select(Level, n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)
head(inter17_edge_list_name)
```

```
## # A tibble: 6 x 4
##   Level n_poste Name1         Name2
```

```
##   <chr>   <dbl> <chr>         <chr>
## 1 PU         13 Ludovic MORGE Marc DAGUZON
## 2 PU         13 Ludovic MORGE Marie-Christine TOCZEK-CAPELLE
## 3 PU         13 Ludovic MORGE Nathalie GAL-PETITFAUX
## 4 MCF        13 Marc DAGUZON  Ludovic MORGE
## 5 MCF        13 Marc DAGUZON  Marie-Christine TOCZEK-CAPELLE
## 6 MCF        13 Marc DAGUZON  Nathalie GAL-PETITFAUX
```

```
inter17_matrix_name <- as.matrix(inter17_edge_list_name[c('Name1', 'Name2')])
```

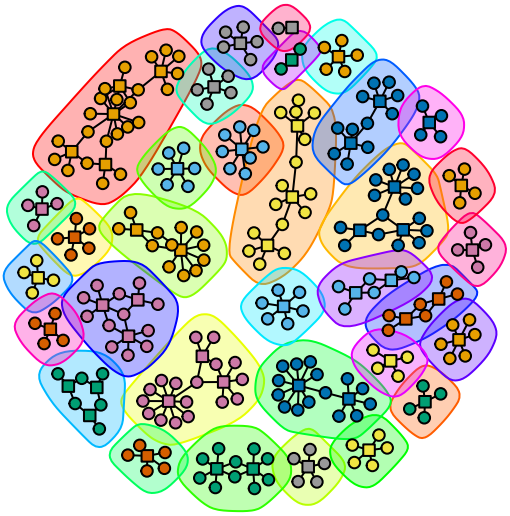
```
inter17$Level <- as.factor(inter17$Level)
g17 <- graph_from_edgelist(inter17_matrix_name, directed = FALSE)
plot(g17,
     vertex.label = NA, vertex.size = 8, vertex.color = inter17$Level)
```



Community Detection - 2017

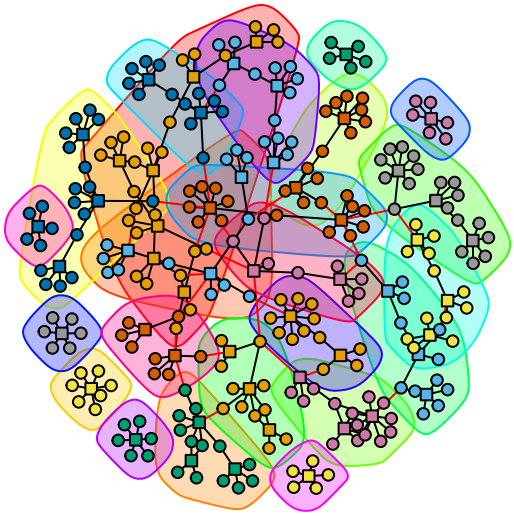
```
set.seed(2680)
cl_i17 <- cluster_louvain(i17_g )
communities = as.data.frame(communities(cl_i17))[,1]
vector <- character(length(communities))
for (i in 1:length(communities)) {
    vector[i] = unlist(communities[i])[1]
}

plot(cl_i17, i17_g,
     vertex.size = 5,
     vertex.label.cex = 1,
     vertex.label = NA,
     main = "Community Detection - Thesis Internal Network in 2017",
    )
```

Community Detection − Thesis Internal Network in 2017



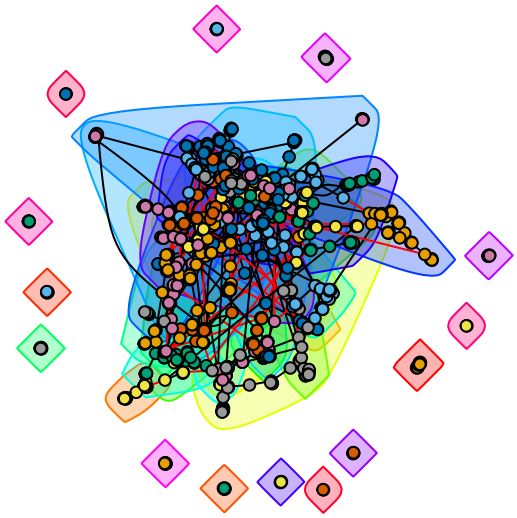
```
set.seed(2680)
cl_e17 <- cluster_louvain(e17_g )
communities = as.data.frame(communities(cl_e17))[,1]
vector <- character(length(communities))
for (i in 1:length(communities)) {
    vector[i] = unlist(communities[i])[1]
}

plot(cl_e17, e17_g,
     vertex.size = 5,
     vertex.label.cex = 1,
     vertex.label = NA,
     main = "Community Detection - Thesis External Network in 2017",
    )
```

Community Detection − Thesis External Network in 2017



Community Detection - External all the years

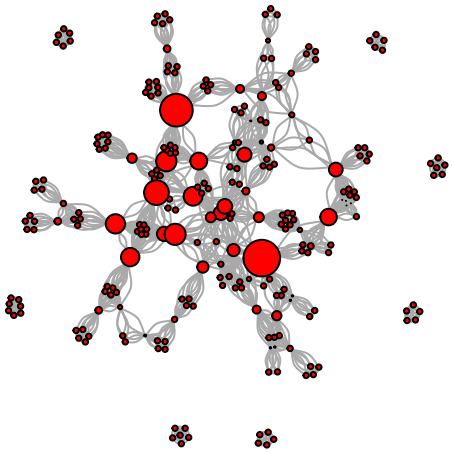
```
exter <- Auditions_db_comp[(Auditions_db_comp$status =="externe"), ]
ex <- exter %>% select(n_poste, Name, year)
ex_g <- graph.data.frame(ex, directed=FALSE)
set.seed(2680)
cl_ex <- cluster_louvain(ex_g )
communities = as.data.frame(communities(cl_ex))[,1]
vector <- character(length(communities))
for (i in 1:length(communities)) {
    vector[i] = unlist(communities[i])[1]
}

plot(cl_ex, ex_g,
     vertex.size = 5,
     vertex.label.cex = 1,
     vertex.label = NA,
     main = "Thesis External Network",
    )
```



##Betweenness

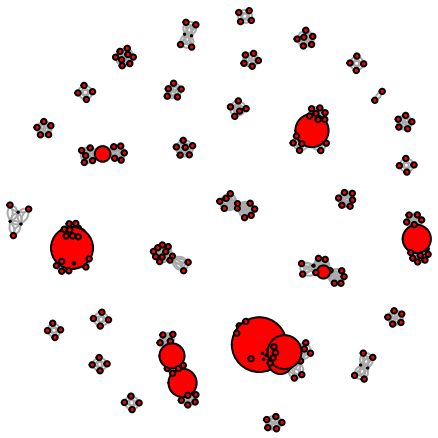
```
exter17_edge_list_name <- exter17 %>% select(Level, n_poste, Name) %>%
  inner_join(., select(., n_poste, Name), by = "n_poste") %>%
  rename(Name1 = Name.x, Name2 = Name.y) %>%
  filter(Name1 != Name2) %>%
  unique %>%
  arrange(n_poste)

exter17_matrix_name <- as.matrix(exter17_edge_list_name[c('Name1', 'Name2')])
g17 <- graph_from_edgelist(exter17_matrix_name, directed = FALSE)
plot(g17,vertex.label = NA, vertex.size = betweenness(g17)*0.003,edge.size=1, vertex.color = "red", mai
```

Betweenness centrality of external juries in 2017



```
inter17_edge_list_name <- inter17 %>% 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)

inter17_matrix_name <- as.matrix(inter17_edge_list_name[c('Name1', 'Name2')])
g17_i <- graph_from_edgelist(inter17_matrix_name, directed = FALSE)
plot(g17_i,vertex.label = NA, vertex.size = betweenness(g17_i)*0.3,edge.size=1, vertex.color = "red", ma
```

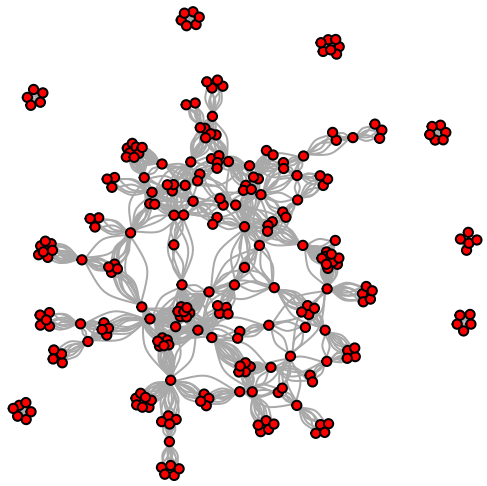
Betweenness centrality of internal juries in 2017



Strength

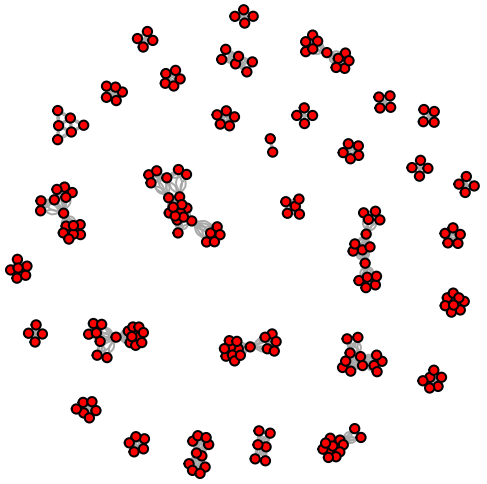
```
plot(g17,vertex.label = NA, vertex.size = 4,edge.size=strength(g17)*0.5, vertex.color = "red", main = "
```

Edge's strength centrality of external juries in 2017



```
plot(g17_i,vertex.label = NA, vertex.size = 4,edge.size=strength(g17_i)*0.5, vertex.color = "red", main
```

Edge's strength centrality of internal juries in 2017



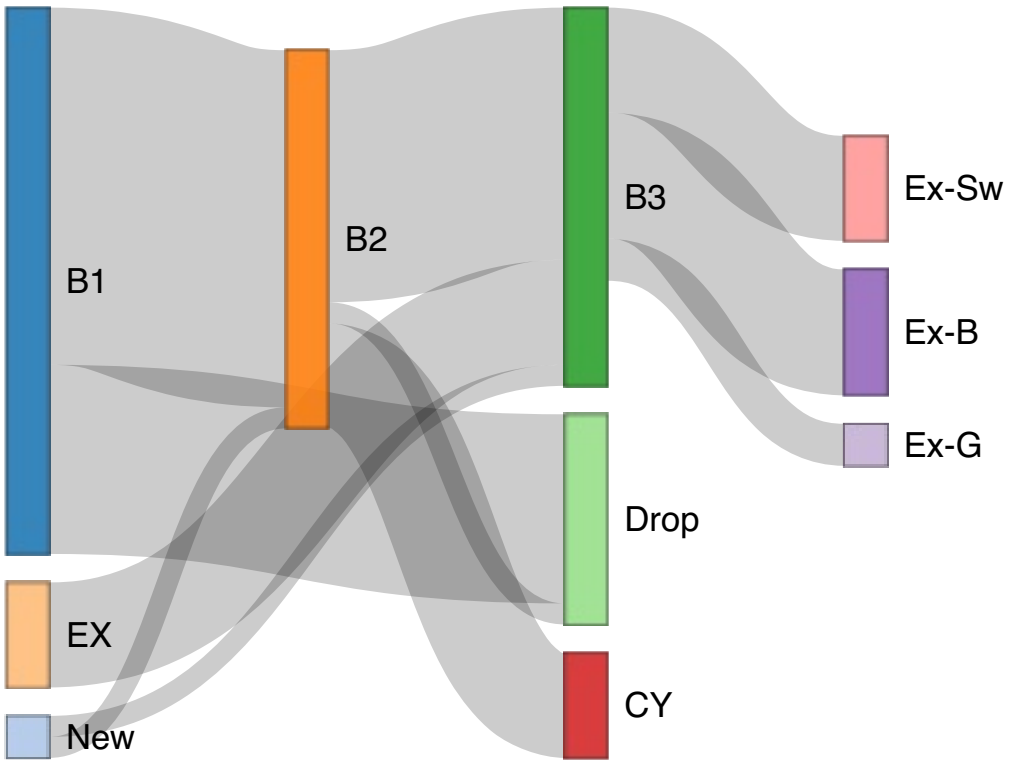
#Sankey diagram and Chord plot

```
links <- data.frame(
  source = c("B1","B1","New","B2","B2","B2","New","EX","B3","B3","B3"),
  target = c("B2","Drop","B2","B3","CY","Drop","B3","B3","Ex-Sw","Ex-B","Ex-G"),
  value = c(17,9,1,12,5,1,1,5,5,6,2)
)
```

```
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
```

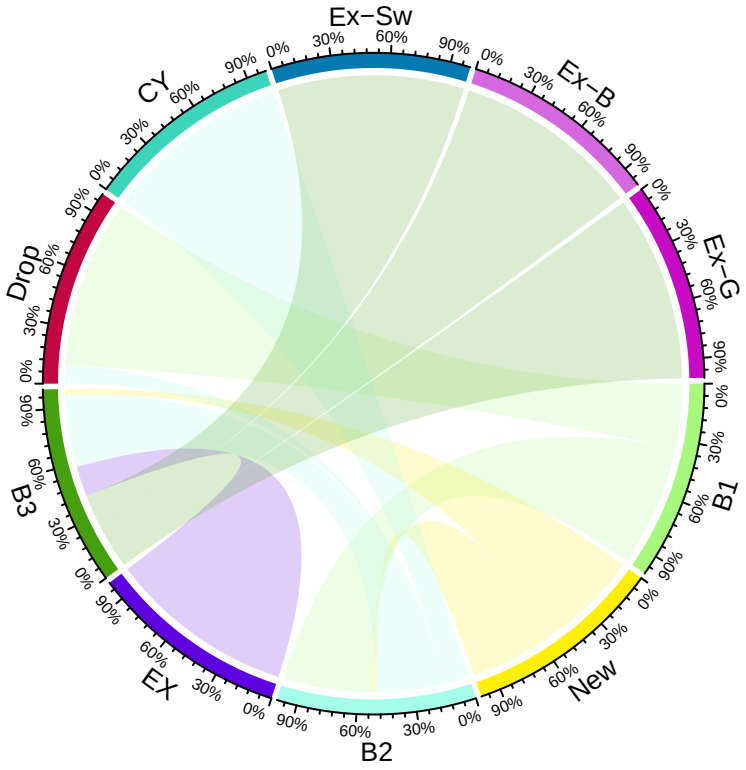
##Sankey

```
sankeyNetwork(Links = links, Nodes = nodes, Source = "ID.source", Target = "ID.target", Value = "value"
```



##Chord plot

```
c <- graph.data.frame(links)
adjaceny.matrix <- get.adjacency(c, sparse = FALSE, attr= "value")
chordDiagram(adjaceny.matrix, transparency = 0.80, scale = TRUE, small.gap = 100)
```


Ex−Sw
Ex−B
Ex−G
B1
New
B2
EX
B3
Drop
CY

Association Rule Mining

December 3, 2021

[65]:
```
import pandas as pd
import numpy as np
import networkx as nx
#from NetworkxD3 import simpleNetworkx
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
from apyori import apriori
import random
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
import math
from arulesviz import Arulesviz
```

[66]:
```
df = pd.read_csv('dataset/tel_samp_rec.csv', encoding="latin-1")
df.head()
```

[66]:

```
  Defence.date                                          Domains  \
0   2010/09/23  Sciences du Vivant [q-bio] / Ecologie, Environ…
1   2009/11/02                    Sciences de l'Homme et Société
2   1996/05/30  Sciences du Vivant [q-bio] / Alimentation et N…
3   2018/02/02  Informatique [cs] / Autre [cs.OH]  \r\n\r\nInf…
4   2015/07/08  Informatique [cs] / Automatique  \r\n\r\nInfor…

  Full.Text.Language  def.date  n.disc        these.id  \
0             French    2010.0       1  tel-00662843v1
1             French    2009.0       1  tel-00491490v1
2             French    1996.0       1  tel-01776364v1
3             French    2018.0       1  tel-02437294v1
4             French    2015.0       1  tel-01245100v1

                      disc1.lev1                disc1.lev2   disc1.lev3  \
0      Sciences du Vivant [q-bio]   Ecologie, Environnement  Ecosystèmes
1  Sciences de l'Homme et Société                       NaN          NaN
2      Sciences du Vivant [q-bio]  Alimentation et Nutrition          NaN
3                Informatique [cs]             Autre [cs.OH]          NaN
4                Informatique [cs]               Automatique          NaN

                   disc2.lev1  … n.tag disc1.rec.lev1  \
```

```
0                        NaN  …     1              X
1                        NaN  …     1             IV
2  Sciences du Vivant [q-bio]  …     2              X
3           Informatique [cs]  …     2              V
4           Informatique [cs]  …     2              V

                              disc1.rec.lev2             disc1.rec.lev3  \
0  67 - Biologie des populations et écologie    Ecologie, Environnement
1                                        NaN                        NaN
2               68 - Biologie des organismes  Alimentation et Nutrition
3                          27 - Informatique                        NaN
4                          27 - Informatique                        NaN

  disc2.rec.lev1     disc2.rec.lev2 disc2.rec.lev3 disc3.rec.lev1  \
0            NaN                NaN            NaN            NaN
1            NaN                NaN            NaN            NaN
2            NaN                NaN            NaN            NaN
3              V  27 - Informatique            NaN            NaN
4              V  27 - Informatique            NaN            NaN

  disc3.rec.lev2 disc3.rec.lev3
0            NaN            NaN
1            NaN            NaN
2            NaN            NaN
3            NaN            NaN
4            NaN            NaN

[5 rows x 25 columns]
```

[67]:
```
disc = df[["disc1.rec.lev1", "disc2.rec.lev1", "disc3.rec.lev1"]]
disc.head()
```

[67]:
```
  disc1.rec.lev1 disc2.rec.lev1 disc3.rec.lev1
0              X            NaN            NaN
1             IV            NaN            NaN
2              X            NaN            NaN
3              V              V            NaN
4              V              V            NaN
```

[68]:
```
transactions = disc.values.tolist()

temp = list()
type(temp)
for x in transactions:
    transaction = [y for y in x if str(y) != 'nan']
    temp.append(transaction)
```

```
transactions = [x for x in temp if len(x) > 1]
```

[69]:
```
encoder = TransactionEncoder().fit(transactions)

# One-hot encode itemsets by applying fit and transform
disc2 = encoder.transform(transactions)
# Convert one-hot encoded data to DataFrame
disc2 = pd.DataFrame(disc2, columns = encoder.columns_)
disc2.head()
```

[69]:

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

[70]:
```
disc2 = disc2[(disc2 == True).sum(axis = 1) >= 2]
```

[71]:
```
disc3 = apriori(disc2, min_support = 0.0003, use_colnames = True).
 ↪sort_values('support', ascending = False).reset_index(drop = True)
disc3['length'] = disc3["itemsets"].apply(lambda x: len(x))
disc3.head()
```

[71]:

	support	itemsets	length
0	0.535092	(IX)	1
1	0.299491	(V)	1
2	0.240289	(VI)	1
3	0.230512	(X)	1
4	0.193812	(IV)	1

[72]:
```
rules = association_rules(disc3, metric = "lift", min_threshold = 1).
 ↪sort_values("lift", ascending = False).reset_index(drop = True)
rules.head()
```

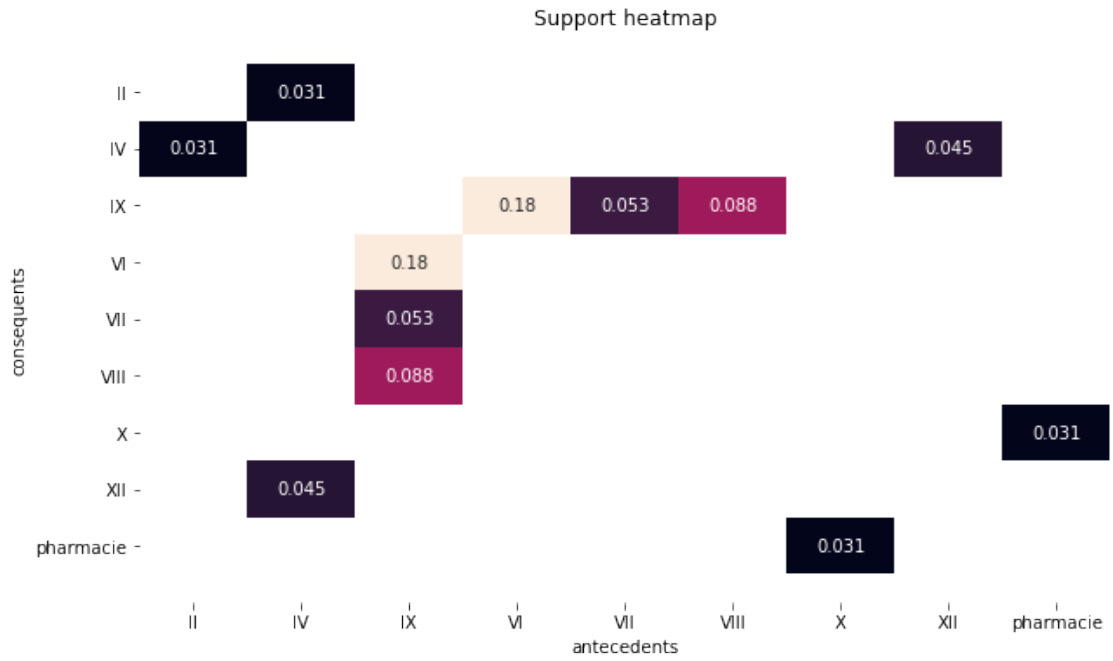
[72]:

	antecedents	consequents	antecedent support	consequent support	support
0	(I)	(I - Droit)	0.027860	0.011117	0.002947
1	(I - Droit)	(I)	0.011117	0.027860	0.002947
2	(I)	(IV)	0.027860	0.193812	0.022636
3	(IV)	(I)	0.193812	0.027860	0.022636

```
4  (pharmacie)          (X)            0.036834            0.230512  0.030672

   confidence      lift  leverage  conviction
0    0.105769  9.514133  0.002637    1.105848
1    0.265060  9.514133  0.002637    1.322748
2    0.812500  4.192208  0.017236    4.299670
3    0.116793  4.192208  0.017236    1.100694
4    0.832727  3.612517  0.022182    4.600202
```

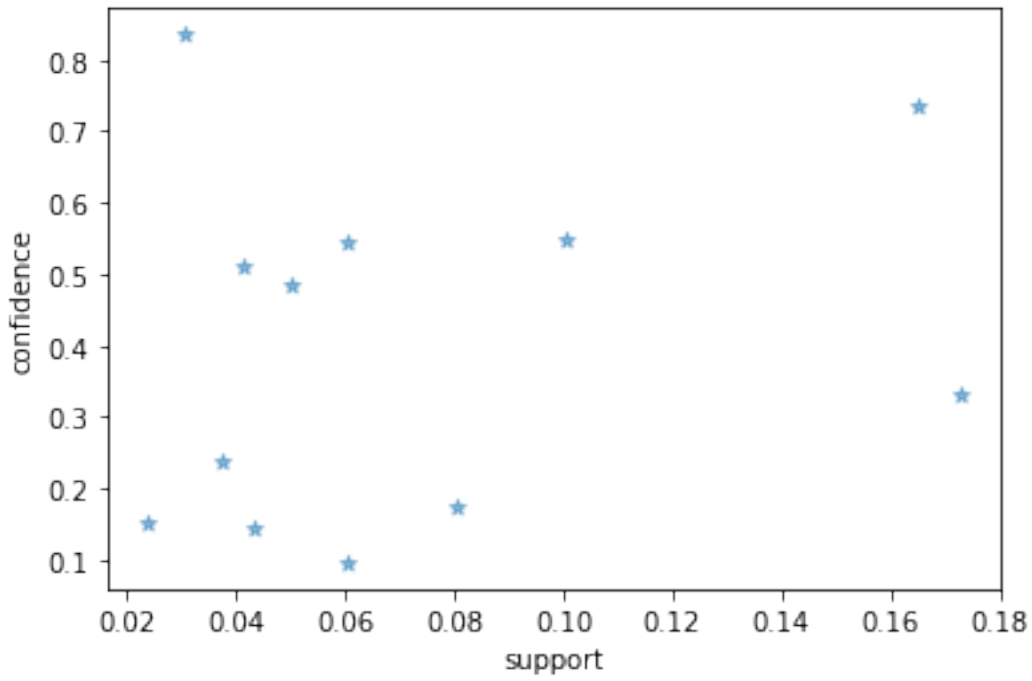
[83]:
```
# Convert antecedents and consequents into strings
rules_copy = rules
rules_copy['antecedents'] = rules['antecedents'].apply(lambda a: ','.
 ↪join(list(a)))
rules_copy['consequents'] = rules['consequents'].apply(lambda a: ','.
 ↪join(list(a)))
```

[84]:
```
# Transform antecedent, consequent, and support columns into matrix
support_table = rules_copy.pivot(index='consequents', columns='antecedents',␣
 ↪values='support')
plt.figure(figsize=(10,6))
sns.heatmap(support_table, annot=True, cbar=False)
b, t = plt.ylim()
b += 0.5
t -= 0.5
plt.ylim(b, t)
plt.yticks(rotation=0)
# Add title and axis names
plt.title('Support heatmap')
plt.show()
```



[85]:
```
support=rules['support'].to_numpy()
confidence=rules['confidence'].to_numpy()
```

[86]:
```
for i in range (len(support)):
   support[i] = support[i] + 0.0025 * (random.randint(1,10) - 5)
   confidence[i] = confidence[i] + 0.0025 * (random.randint(1,10) - 5)

plt.scatter(support, confidence, alpha=0.5, marker="*")
plt.xlabel('support')
plt.ylabel('confidence')
plt.show()
```



[87]:
```
rules_copy.head()
```

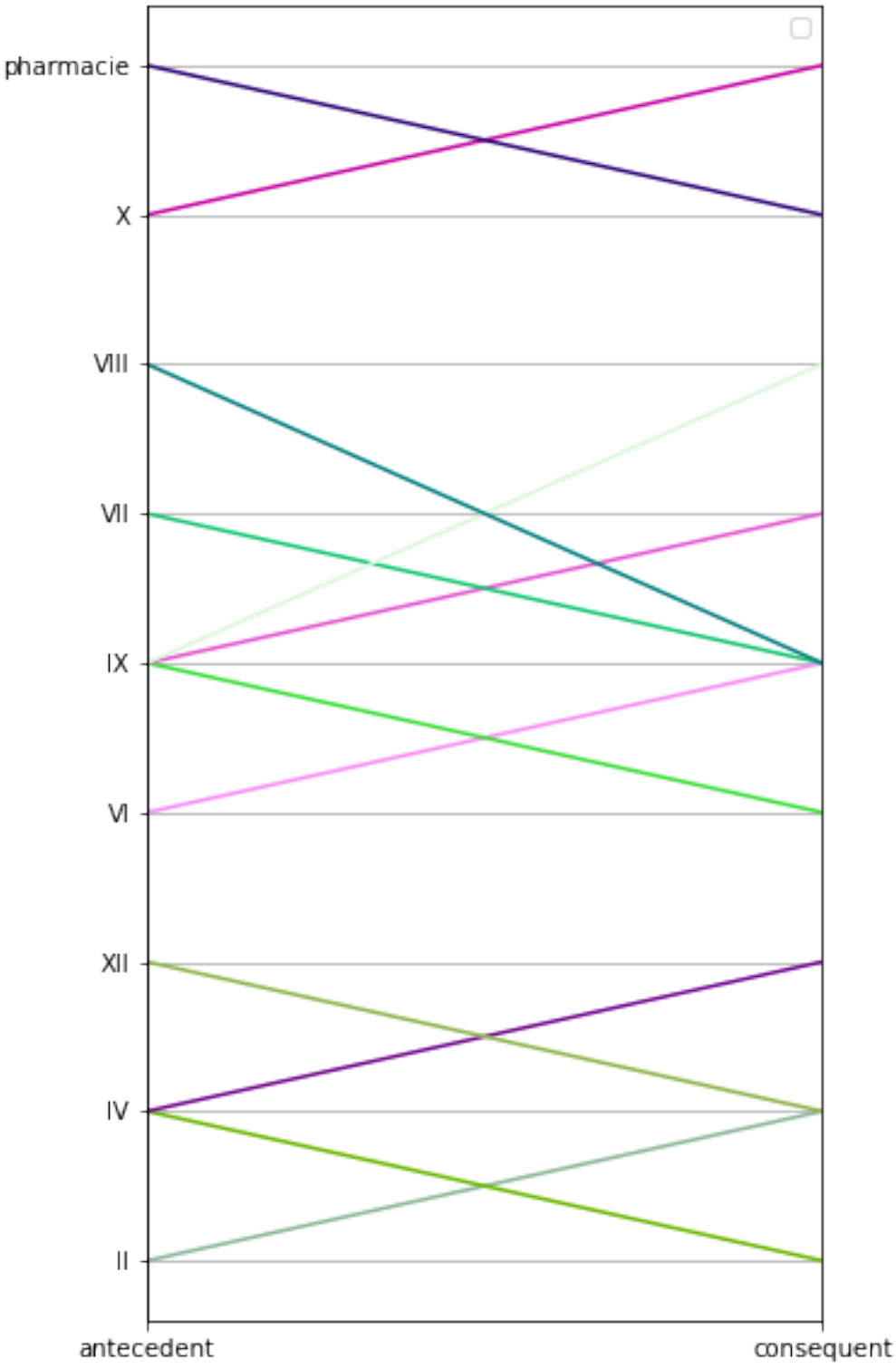
[87]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent	consequent	rule
0	II	IV	0.062952	0.193812	0.041342	0.510372	2.568842	0.019141	1.605544	II	IV	0
1	IV	II	0.193812	0.062952	0.023842	0.154214	2.568842	0.019141	1.117814	IV	II	1
2	IV	XII	0.193812	0.092285	0.037370	0.236513	2.508679	0.026984	1.181172	IV	XII	2
3	XII	IV	0.092285	0.193812	0.049870	0.483712	2.508679	0.026984	1.569106	XII	IV	3
4	VI	IX	0.240289	0.535092	0.165194	0.734097	1.362563	0.046617	1.716140	VI	IX	4

[88]:
```
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

disc2_sets = disc2.applymap(encode_units) # Encode the basket
disc2_sets = disc2_sets[basket_sets.sum(axis=1) > 1] # Only keep transaction␣
↪that has more than 1 item
```

[89]:
```
# Function to convert rules to coordinates.
def rules_to_coordinates(rules):
    rules['antecedent'] = rules['antecedents'].apply(lambda antecedent:␣
 ↪list(antecedent)[0])
    rules['consequent'] = rules['consequents'].apply(lambda consequent:␣
 ↪list(consequent)[0])
    rules['rule'] = rules.index
    return rules[['antecedent','consequent','rule']]
```

[90]:
```
from pandas.plotting import parallel_coordinates

# Compute the frequent itemsets
frequent_itemsets = apriori(disc2_sets, min_support = 0.03,
                            use_colnames = True)

# Compute rules from the frequent itemsets
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

# Convert rules into coordinates suitable for use in a parallel coordinates plot
coords = rules_to_coordinates(rules)

# Generate parallel coordinates plot
plt.figure(figsize=(5,10))
parallel_coordinates(coords, 'rule')
plt.legend([])
plt.grid(True)
plt.show()
```


pharmacie
X
VIII
VII
IX
VI
XII
IV
II
antecedent
consequent

[ ]:

[ ]: