UMONS
Université de Mons

« Automatiser l'analyse de graphes d'hyperliens avec igraph »

Dr Ir Robert Viseur
Assistant Professor

11èmes Rencontres R
Session « Reporting »

Mardi 20 mai 2025 11h50–12h10 (Auditoire Gutenberg))

POLYTECH MONS

Résumé

Spygraph est un crawler configurable (focused crawler en anglais) développé en Python (Viseur, 2022). Il permet d'explorer un ensemble de sites web, de manière plus ou moins ciblée, puis d'exporter les hyperliens et les domaines découverts. L'outil facilite l'exploration itérative d'écosystèmes d'affaires, au travers des sites web des membres, dans un premier temps de manière divergente (par exemple pour découvrir de nouveaux acteurs jusqu'à alors inconnus), dans un second temps de manière convergente (pour concentrer l'analyse sur un ensemble validé de sites web). L'exportation des résultats de l'exploration peut se faire sous la forme d'un fichier de tableur (CSV) et d'un fichier de graphe (DOT, GML). Une première version du logiciel exportait les graphes au format DOT uniquement. Les formats CSV et DOT permettaient notamment une analyse dans le logiciel libre Gephi. Une version ultérieure permet l'exportation au format GML. Ce dernier est exploitable au sein du logiciel igraph. igraph est un ensemble d'outils d'analyse de réseaux, libre et gratuit, programmable en R, Python, Mathematica et C/C++. Deux cas d'utilisation sont présentés : d'une part, l'utilisation, avec Spygraph, à l'aide du langage Python, d'autre part, l'analyse d'un fichier au format GML, exporté depuis Spygraph, directement dans R. L'objectif est de générer, après le crawl, premièrement, un ensemble de métriques (p. ex. degree, betweeness centrality et pagerank), deuxièmement, une prévisualisation des communautés. Une comparaison des différentes approches d'analyse est proposée en guise de conclusion.

Analyse des réseaux sociaux

Un réseau social est défini comme « *une représentation structurée des acteurs sociaux (nœuds) et de leurs interconnexions (liens)* » (Arif, 2015). L'ensemble forme un graphe.

Les nœuds peuvent être des personnes, des lieux, des organisations ou des pages web (Arif, 2015). Les interconnexions sont basées sur l'amitié, le lien familial, la co-rédaction ou les hyperliens (Arif, 2015).

Les méthodes et outils pour l'analyse de réseaux sociaux sont rassemblées au sein d'une discipline appelée « *Social Network Analysis* » (SNA).

Il existe de nombreux exemples de réseaux sociaux sur base d'interactions sociales :

- les citations entre papiers scientifiques (Arif, 2015),
- les interactions dans un salon de discussion (Viseur, Charleux & Fally, 2023),
- les liens hypertextes entre pages web (Viseur, 2022 ; Viseur, 2014).

Utilisation de métriques

L'analyse de réseaux sociaux peut s'appuyer sur des métriques, en particulier les métriques de centralité (« *centrality* »).

La centralité permet de mesurer « *l'importance relative des nœuds et des arêtes dans un graphe* » (Arif, 2015).

Elle se décline en plusieurs mesures, incluant :

- la centralité de degré,
  (« *Degree Centrality* »)
- la centralité de proximité,
  (« *Closeness Centrality* »)
- la centralité d'intermédiarité,
  (« *Betweenness Centrality* »)
- la centralité des vecteurs propres.
  (« *Eigenvector Centrality* »)

Mesures de centralité

La « *Degree Centrality* » d'un nœud fournit « *le nombre de liens qui y sont rattachés, utilisé pour identifier les nœuds ayant le plus grand nombre de connexions dans le réseau* » (Arif, 2015).

La « *Eigenvector Centrality* » améliore la « *Degree Centrality* », car « *elle dépend non seulement du nombre de liens rattachés, mais aussi de la qualité de ces liens* » (Arf, 2015). Sont proches dans le principe : le « *Pagerank* », popularisé par Google, et la « *Katz Centrality* ».

La « *Closeness Centrality* » fournit « *le degré de proximité (directe ou indirecte) entre un nœud et le reste des nœuds du réseau* » (Arif, 2015).

La « *Betweenness Centrality* » calcule « *la fraction de tous les chemins les plus courts qui passent par un nœud donné, ou, en termes simples, le nombre de fois où un nœud agit comme un pont le long du chemin le plus court entre deux autres nœuds* » (Arif, 2015).

Logiciels pour l'analyse de graphes

Deux logiciels libres ressortent pour l'analyse de graphe : Gephi (Bastian, Heymann & Jacomy, 2009) et igraph (Csardi & Nepusz, 2006).

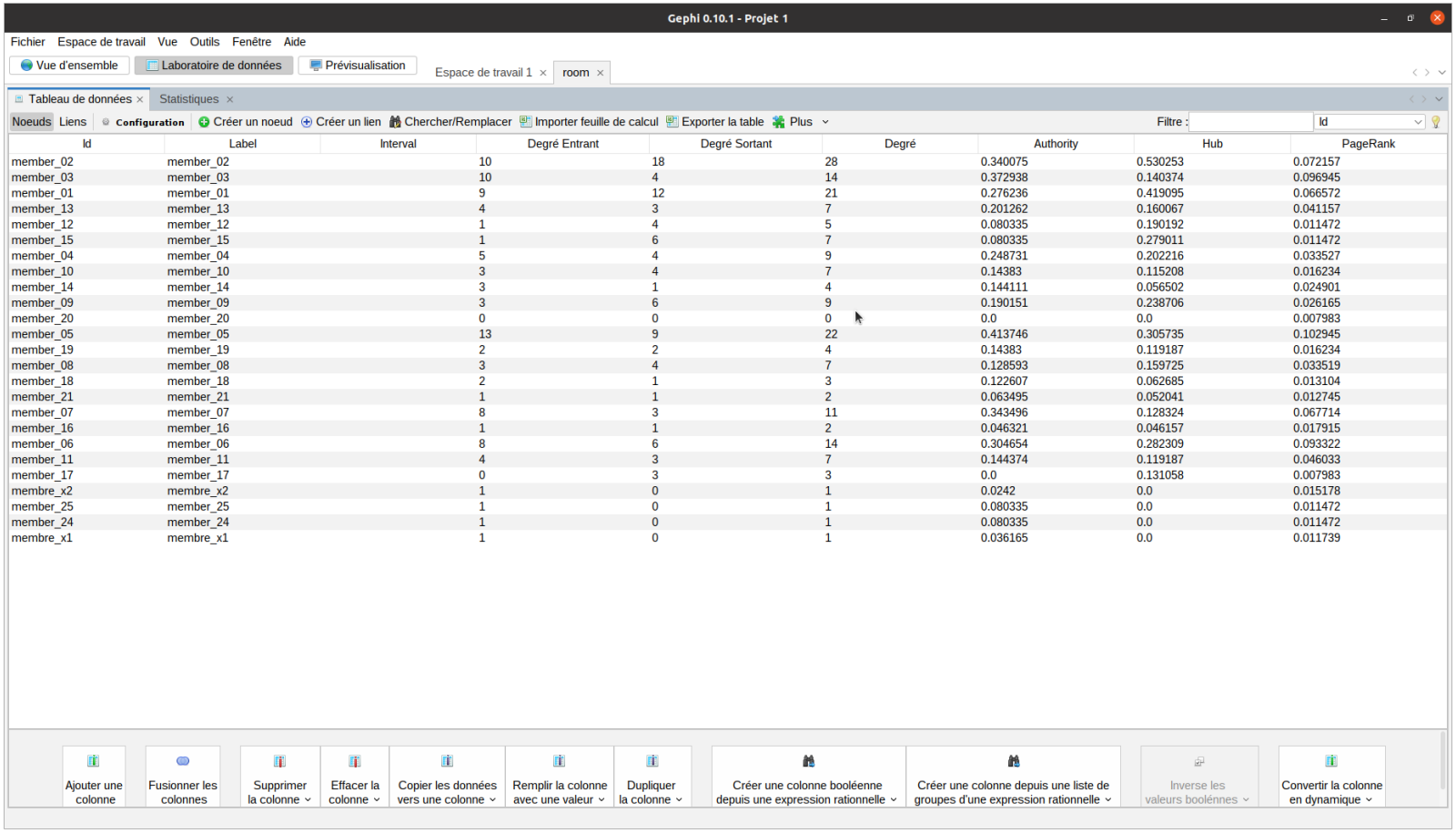
Le logiciel igraph constitue une collection d'outils d'analyse de réseaux pouvant être programmé en R, Python, Mathematica et C/C++.

Ces logiciels permettent :

- le calcul de mesures de centralité,
- l'identification de communautés,
- la visualisation des graphes.

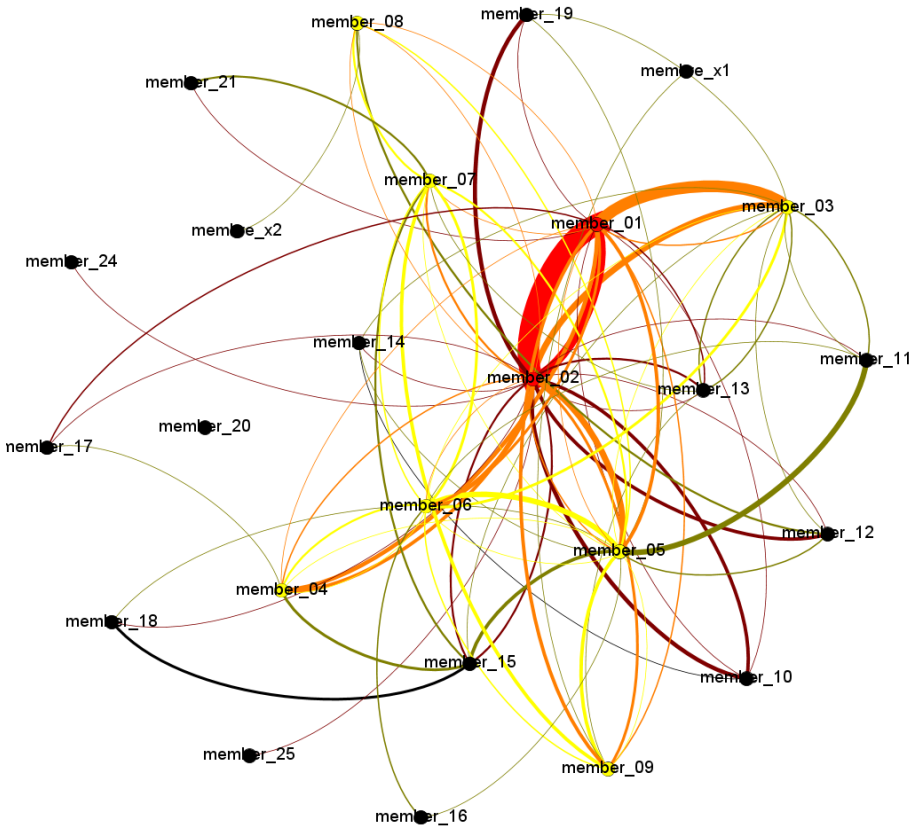
Plusieurs formats de fichiers permettent d'importer un graphe dont DOT (Gephi) et GML (Gephi, igraph).

Analyse d'une communauté en ligne (Viseur *et al.*, 2023) #1



Id	Label	Interval	Degré Entrant	Degré Sortant	Degré	Authority	Hub	PageRank
member_02	member_02		10	18	28	0.340075	0.530253	0.072157
member_03	member_03		10	4	14	0.372938	0.140374	0.096945
member_01	member_01		9	12	21	0.276236	0.419095	0.066572
member_13	member_13		4	3	7	0.201262	0.160067	0.041157
member_12	member_12		1	4	5	0.080335	0.190192	0.011472
member_15	member_15		1	6	7	0.080335	0.279011	0.011472
member_04	member_04		5	4	9	0.248731	0.202216	0.033527
member_10	member_10		3	4	7	0.14383	0.115208	0.016234
member_14	member_14		3	1	4	0.144111	0.056502	0.024901
member_09	member_09		3	6	9	0.190151	0.238706	0.026165
member_20	member_20		0	0	0	0.0	0.0	0.007983
member_05	member_05		13	9	22	0.413746	0.305735	0.102945
member_19	member_19		2	2	4	0.14383	0.119187	0.016234
member_08	member_08		3	4	7	0.128593	0.159725	0.033519
member_18	member_18		2	1	3	0.122607	0.062685	0.013104
member_21	member_21		1	1	2	0.063495	0.052041	0.012745
member_07	member_07		8	3	11	0.343496	0.128324	0.067714
member_16	member_16		1	1	2	0.046321	0.046157	0.017915
member_06	member_06		8	6	14	0.304654	0.282309	0.093322
member_11	member_11		4	3	7	0.144374	0.119187	0.046033
member_17	member_17		0	3	3	0.0	0.131058	0.007983
membre_x2	membre_x2		1	0	1	0.0242	0.0	0.015178
member_25	member_25		1	0	1	0.080335	0.0	0.011472
member_24	member_24		1	0	1	0.080335	0.0	0.011472
membre_x1	membre_x1		1	0	1	0.036165	0.0	0.011739

Analyse d'une communauté en ligne (Viseur *et al.*, 2023) #2



G	H	I	K	M
indegree	outdegree	degree	closnesscentrality	betweenesscentrality
0	0	0	0	0
0	3	3	0,5	0
1	4	5	0,52381	0
1	6	7	0,536585	0
1	0	1	0	0
1	0	1	0	0
1	0	1	0	0
1	1	2	0,314286	0
2	1	3	0,385965	0
1	0	1	0	0
2	2	4	0,407407	0
3	4	7	0,5	0,47619
1	1	2	0,360656	0
3	1	4	0,314286	0,333333
3	6	9	0,5	19,75
3	4	7	0,536585	27,1
5	4	9	0,52381	2,7
4	3	7	0,488889	7,892857

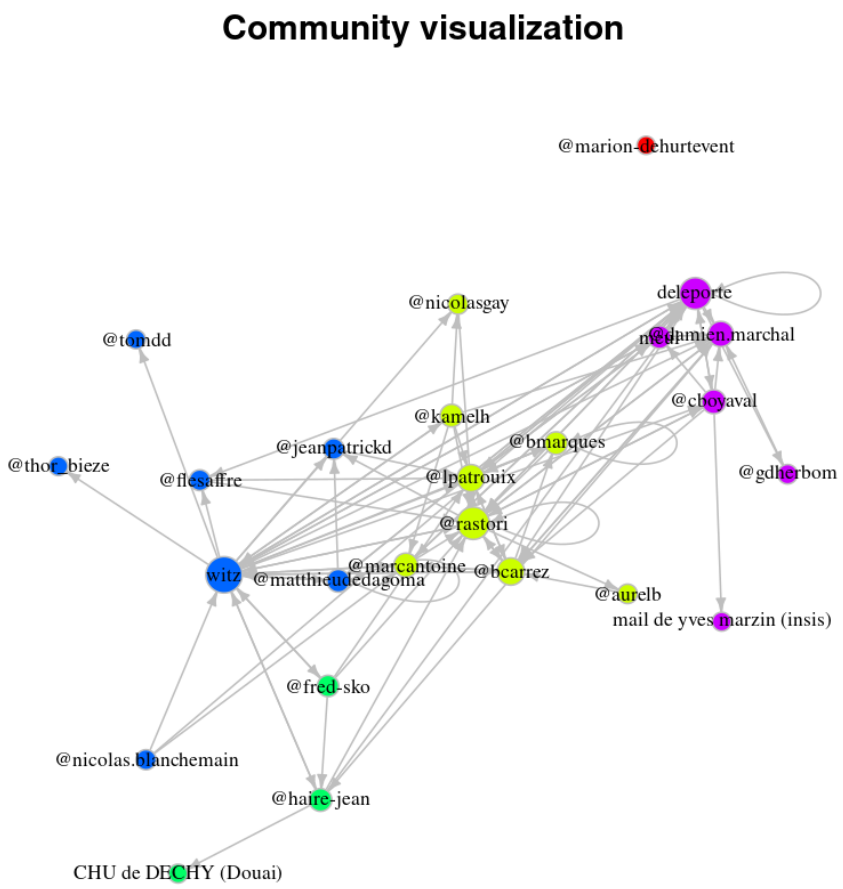
Reproduction sous R avec igraph #1

R avec igraph permet le calcul de métriques :

	node	degree	degree_in	degree_out	eigenvector	betweenness
1	witz	28	10	18	0.6856781	168.511905
12	@rastori	22	13	9	1.0000000	99.678571
3	deleporte	21	9	12	0.7558246	62.745238
2	@lpatrouix	14	10	4	0.8143815	38.166667
19	@bcarrez	14	8	6	0.8127615	46.635714
17	@damien.marchal	11	8	3	0.7145773	35.533333
7	@marcantoine	9	5	4	0.3904061	2.700000
10	@cboyaval	9	3	6	0.4026362	19.750000
4	meul	7	4	3	0.4384254	7.892857
6	@kamelh	7	1	6	0.1130774	0.000000

R avec igraph permet aussi la visualisation du graphe avec, si souhaité, la coloration des nœuds (unique ou en fonction d'une métrique) et la détection de communautés.

Reproduction sous R avec igraph #2



Utilisation de Spygraph

Spygraph est un logiciel de crawl ciblé (« *focused crawler* ») permettant d'explorer un ensemble de pages web, en se limitant aux sites pré-sélectionnés (mode « *inside* ») ou pour en découvrir de nouveaux (mode « *outside* ») (Viseur, 2022).

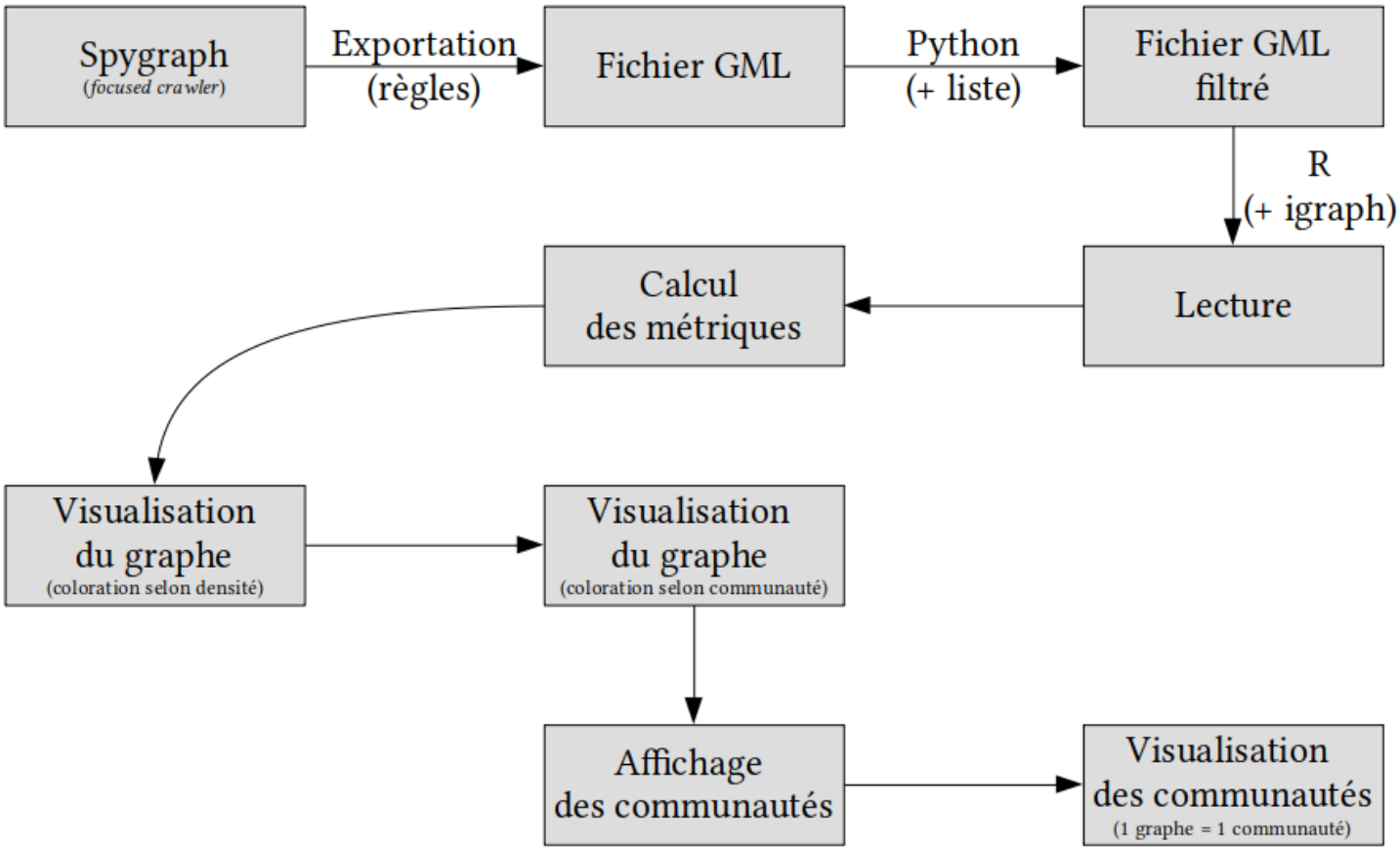
Le résultat d'un crawl peut ensuite être exporté au format DOT (analyse dans Gephi) ou au format GML (analyse dans igraph).

L'exportation peut se baser sur des hyperliens, sur des noms de domaines (avec sous-domaine) ou sur des noms de domaines (sans sous-domaine), ce qui permet de réduire la volumétrie de nœuds.

En fonction des résultats, l'exportation peut ensuite être réitérée de manière à inclure ou exclure certains sites web (p. ex. réseaux sociaux).

Spygraph entraîne la nécessité de pouvoir traiter de grands graphes.

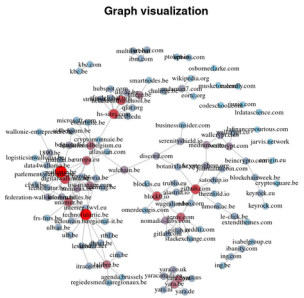
Analyse d'un graphe d'hyperliens #1

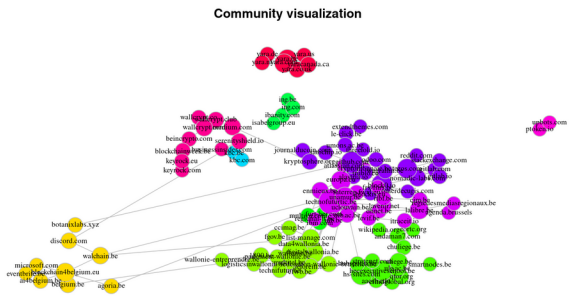


Analyse d'un graphe d'hyperliens #2

R avec igraph permet toujours le calcul de métriques, de manière à caractériser les nœuds et faciliter l'analyse des relations entre acteurs.

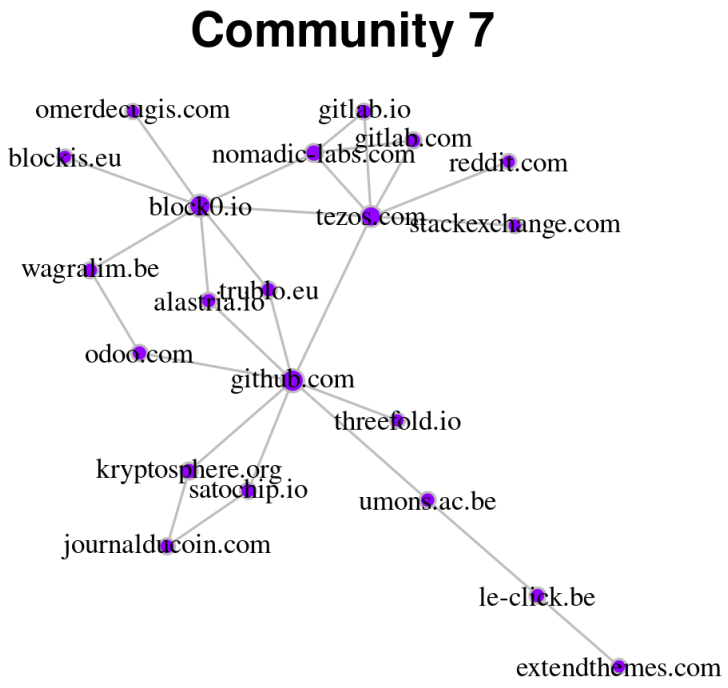
	node	degree	degree_in	degree_out	eigenvector	betweenness
2414	unamur.be	25	25	25	0.663486270	5407.114
2273	technofuturtic.be	22	22	22	0.760867880	13007.986
2562	wallonie.be	22	22	22	1.000000000	8285.543
1443	lalibre.be	21	21	21	0.115325357	5082.500
157	andaman7.com	17	17	17	0.003133171	4561.000
2640	yara.be	17	17	17	0.000000000	125.500
1540	lrdatascience.com	16	16	16	0.000000000	120.000
362	blockchain4belgium.eu	13	13	13	0.105812354	3320.329
732	digitalwallonia.be	12	12	12	0.759666811	6618.457
2290	tezos.com	12	12	12	0.007960624	4032.078





Analyse d'un graphe d'hyperliens #3

La taille rend visualisation du graphe plus difficile, d'où l'intérêt d'une visualisation partielle par communauté.



Script final

Ce script comporte six sections :

(1) la configuration,
(2) la lecture du fichier GML,
(3) le calcul des métriques,
(4) la création des graphes (sauvegarde en PNG),
(5)le calcul puis l'affichage des communautés,
(6) la création des graphes partiels (sauvegarde en PNG).

```
# ------------------------------------------------
# Copyright: Robert Viseur
<robert.viseur@umons.ac.be>
# License: MIT License (MIT).
# Release: May 20, 2025.
# ------------------------------------------------

library(igraph)

# ------------------------------------------------

graphmono    <- FALSE
graphbig     <- TRUE
graphfile    <- "myfilename.gml"
graphfnplot1 <- "myfilename-plot-1.png"
graphfnplot2 <- "myfilename-plot-2.png"
graphprplot3 <- "myfilename-community-"
degrinmin    <- 2

files_to_delete <- list.files("^carruana-
2f.*\\.png$", full.names = TRUE)
file.remove(files_to_delete)

if (!file.exists(graphfile)) {
  stop(paste("Error : the file '", graphfile, "'
doesn't exist in the current directory."))
}
```

```
# ------------------------------------------------
# Load GML file then display head.
# ------------------------------------------------

graph <- read_graph(graphfile, format = "gml")
print(graph)
print(head(V(graph)))

# ------------------------------------------------
# Compute the centrality metrics :
# - degree centrality,
# - pagerank,
# - betweeness centrality.
# Display the values for top nodes
# ------------------------------------------------

node_names <- V(graph)$label
deg <- degree(graph, mode = "all")
deg_in <- degree(graph, mode = "in")
deg_out <- degree(graph, mode = "out")
eigenvector_vals <- eigen_centrality(graph,
directed = TRUE)$vector
betweenness_vals <- betweenness(graph)

centrality_df <- data.frame(
  node = node_names,
  degree = deg,
  degree_in = deg_in,
  degree_out = deg_out,
  eigenvector = eigenvector_vals,
  betweenness = betweenness_vals
)

top_nodes <- centrality_df[order(-
centrality_df$degree), ][1:10, ]
print(top_nodes)

# ------------------------------------------------
# Create the graph then save it in PNG.
# ------------------------------------------------

graph_raw <- graph

nodes_to_keep <- V(graph)[deg_in >= degrinmin]
graph <- induced_subgraph(graph, vids =
nodes_to_keep)
deg <- degree(graph, mode = "all")
nodes_to_keep <- V(graph)[deg > 0]
graph <- induced_subgraph(graph, vids =
nodes_to_keep)
deg <- degree(graph, mode = "all")

set.seed(123)
if (graphbig) {
  layout <- layout_with_fr(graph)
} else {
  layout <- layout_with_kk(graph)
}

deg_range <- max(deg) - min(deg)
if (deg_range == 0) {
  V(graph)$size <- 5
} else {
  V(graph)$size <- 5 + 5 * ((deg - min(deg)) /
deg_range)
}

if (graphmono) {
  V(graph)$color <- "skyblue"
} else {
```

```
  deg_scaled <- (deg - min(deg)) / (max(deg) -
min(deg))
  palette_fn <- colorRampPalette(c("skyblue",
"red"))
  V(graph)$color <- palette_fn(100)
[as.numeric(cut(deg_scaled, breaks = 100))]
}

V(graph)$label.color <- "black"
V(graph)$label.cex <- 0.7
V(graph)$frame.color <- "gray"

E(graph)$color <- "gray"
E(graph)$arrow.size <- 0.4

png(graphfnplot1, width = 1600, height = 900, res =
150)
plot(
  graph,
  layout = layout,
  main = "Graph visualization"
)
dev.off()

# ------------------------------------------------
# Compute the communities then display Top 5
# ------------------------------------------------

graph_undirected <- as.undirected(graph, mode =
"collapse")
communities <- cluster_louvain(graph_undirected)
cat("Nombre de communautés :", length(communities),
"\n")
V(graph)$community <- membership(communities)
num_comms <- length(unique(V(graph)$community))
V(graph)$color <- rainbow(num_comms)[V(graph)
$community]


comm_sizes <- sizes(communities)
num_top <- min(5, length(comm_sizes))
top_5_ids <- names(sort(comm_sizes, decreasing =
TRUE)[1:num_top])
for (i in seq_along(top_5_ids)) {
  comm_id <- as.numeric(top_5_ids[i])
  nodes_in_comm <- V(graph)[community == comm_id]
$label
  cat(paste0("Community ", comm_id, " (",
length(nodes_in_comm), " nodes) :\n"))
  print(nodes_in_comm)
}

# ------------------------------------------------
# Create the graph then save it in PNG
# (with no size 1 commununity).
# ------------------------------------------------

comm_sizes <- sizes(communities)
valid_comms <-
as.numeric(names(comm_sizes[comm_sizes > 1]))
nodes_to_keep <- V(graph)[community %in%
valid_comms]
graph <- induced_subgraph(graph, vids =
nodes_to_keep)

graph_undirected <- as.undirected(graph, mode =
"collapse")
communities <- cluster_louvain(graph_undirected)
V(graph)$community <- membership(communities)
```

```
layout <- layout_with_fr(graph, niter = 1000)
layout <- layout * 10
centroids <- t(sapply(unique(V(graph)$community),
function(comm) {
  nodes <- which(V(graph)$community == comm)
  colMeans(layout[nodes, , drop = FALSE])
}))
for (v in seq_along(V(graph))) {
  comm_id <- V(graph)$community[v]
  layout[v, ] <- layout[v, ] + 0.5 *
(centroids[comm_id, ] - layout[v, ])
}

png(graphfnplot2, width = 1600, height = 900, res =
150)
plot(
  graph,
  layout = layout,
  asp = 0,
  main = "Community visualization"
)
dev.off()

# ------------------------------------------------
# Create a graph by community then save it in PNG
# (top 5, only for big graphes).
# ------------------------------------------------

if (graphbig) {

  graph_undirected <- as.undirected(graph, mode =
"collapse")
  communities <- cluster_louvain(graph_undirected)
  V(graph)$community <- membership(communities)

  comm_sizes <- sizes(communities)
  num_top <- min(5, length(comm_sizes))
  top_5_ids <- names(sort(comm_sizes, decreasing =
TRUE)[1:num_top])

  for (i in seq_along(top_5_ids)) {
    comm_id <- as.numeric(top_5_ids[i])
    nodes <- V(graph)[community == comm_id]
    subgraph <- induced_subgraph(graph, vids =
nodes)

    layout_sub <- layout_with_fr(subgraph, niter =
1000) * 6

    filename <- paste0(graphprplot3, comm_id, "-n",
length(nodes), ".png")

    png(filename, width = 1600, height = 900, res =
150)
    par(mar = c(0, 0, 2, 0))
    plot(
      subgraph,
      layout = layout_sub,
      main = paste("Community", comm_id),
      vertex.label.cex = 0.7,
      edge.arrow.size = 0.3
    )
    dev.off()
  }
}

# ------------------------------------------------
```

Comparaison

	Gephi	igraph
+	Interface graphique conviviale et interactive. Richesse fonctionnelle (métriques diversifiées, détection de communautés, visualisation des graphes).	Projet de logiciel libre dont le développement est actif (voir dépôts GitHub). **Python :** Automatisation partielle des analyses par l'intégration possible des outils au sein d'un programme écrit en Python. **R :** Outil de référence abondamment documenté au sein de l'écosystème R. Excellent support de R (avec la bibliothèque igraph) par ChatGPT (assistance à la programmation)
-	Analyse exclusivement manuelle. Pérennité incertaine du logiciel après une absence de mises à jour entre 2018 et 2022.	Moindre convivialité du fait d'une utilisation par programmation (R, Python).

Références

Arif, T. (2015). The mathematics of social network analysis: metrics for academic social networks. International Journal of Computer Applications Technology and Research, 4(12), 889-893. https://ijcatr.com/archives/volume4/issue12/ijcatr04121003.pdf.

Bastian, M., Heymann, S., & Jacomy, M. (2009, March). Gephi: an open source software for exploring and manipulating networks. In Proceedings of the international AAAI conference on web and social media (Vol. 3, No. 1, pp. 361-362). https://doi.org/10.1609/icwsm.v3i1.13937.

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. Complex syst, 1695, 1-9. [url]

Viseur, R., Charleux, A., & Fally, B. (2023). How makers responded to the Personal Protective Equipment shortage during the COVID-19 pandemic: An analysis focused on the Hauts-de-France region. European Management Journal, 41(4), 634-647. https://doi.org/10.1016/j.emj.2023.04.014.

Viseur, R. (2022). Spygraph : un robot d'exploration léger dédié à l'analyse de graphes d'hyperliens. Actes du congrès INFORSID. , Dijon (France), 31 mai-03 juin 2025. http://inforsid.fr/actes/2022/INFORSID_2022_p81-84.pdf.

Viseur, R. (2014). Initial Results from the Study of the Open Source Sector in Belgium. In Proceedings of The International Symposium on Open Collaboration (pp. 1-5). https://doi.org/10.1145/2641580.2641591.

Cette recherche est supportée par le projet Inside3D (Interreg « France – Wallonie – Vlaanderen).
Les outils testés ou développés sur des jeux de données pré-existants serviront ultérieurement
à l'analyse de l'écosystème formé par les acteurs de l'impression 3D en contexte hospitalier.

UMONS
Université de Mons

Université de Mons
Faculté Warocqué d'économie
et de gestion - Service TIC
Place Warocqué, 17
B-7000 Mons

Tél. : +32.65.373.201

www.umons.ac.be
info.warocque@umons.ac.be

Plus d'information...

Dr Ir Robert VISEUR
Chargé de cours

Tél. : +32.65.374.054
robert.viseur@umons.ac.be