

Mémoire de Stage - Package ManagHAL

Martin AMIENS

2024-06-16

Table of contents

1	{MANAGHAL}, DONNEES BIBLIOMETRIQUES, RESEAUX ET ANALYSES STATISTIQUES DE RESEAUX	3
1.1	Introduction	3
1.1.1	Sciences Ouvertes	3
1.1.2	{ManagHAL}, package R d'extraction de données bibliométrique HAL et d'analyse de réseaux	3
1.2	Méthodes	4
1.2.1	Modèle de graphe aléatoire	4
1.2.2	Méthodes informatiques	6
1.3	Résultats	7
1.3.1	Accessibilité et généricité de {managHAL}	7
1.3.2	Ajout de fonctionnalité à {managHAL}	10
1.4	Discussion	12
1.5	Annexes	14
1.5.1	load_team_table_url	14
1.5.2	load_team_table_csv	14
1.5.3	load_team_table	16
1.5.4	Exemple d'une documentation classique avec Roxygen2	16
1.5.5	HAL_query	17
1.5.6	Script_R_reseau_co-auteur	19
1.5.7	exemple_reseau_unipartite_bipartite	24
1.5.8	Script_R_SBM	24
1.5.9	Script_R_Sous-groupe	26
	BIBLIOGRAPHIE	29

1 {MANAGHAL}, DONNEES BIBLIOMETRIQUES, RESEAUX ET ANALYSES STATISTIQUES DE RESEAUX

1.1 Introduction

1.1.1 Sciences Ouvertes

En 2017, AgroParisTech signe l'Appel de Jussieu pour promouvoir la biodiversité, visant à maintenir la diversité éditoriale et éviter la domination des grands groupes éditeurs. AgroParisTech, en partenariat avec INRAE, adopte en 2020 une politique de science ouverte ([AgroParisTech \(2024\)](#), voir: La science ouverte à AgroParisTech – Politique d'établissement janvier 2020), facilitant la libre diffusion et la transparence des recherches scientifiques. Un exemple concret des principes appliqués suite à l'adoption de cette politique sont les principes FAIR pour la gestion des données de recherche : Findable (facilement trouvable), Accessible (accès clairement défini), Interoperable (intégrable avec d'autres outils), et Reusable (réutilisable avec toutes les informations nécessaires).

Pour atteindre ces objectifs, AgroParisTech crée le 2 janvier 2023 une cellule HAL, dirigée par la Direction de la Recherche, de l'Innovation et du Transfert Technologique, et par la Direction de la Documentation et du Patrimoine Culturel. Cette cellule facilite l'intégration de l'archive ouverte HAL dans la communauté de recherche d'AgroParisTech et veille à la qualité des métadonnées. HAL, ou HyperArticles en Lignes, est "l'archive nationale pour la diffusion ouverte des résultats de recherche" ("[About HAL](#)" (n.d.)). Elle est une plateforme en ligne développée en 2001 par le Centre pour la communication scientifique directe du CNRS. Cette plateforme dispose d'un API permettant l'extraction et l'analyse des données via des outils comme RStudio. Une explication plus détaillée de cet API et d'une requête HAL sera faite dans la section méthode.

1.1.2 {ManagHAL}, package R d'extraction de données bibliométrique HAL et d'analyse de réseaux

Les outils pour réaliser des analyses bibliométriques se répartissent en trois catégories, "general bibliometric and performance analysis, science mapping analysis, and libraries" ([Moral-Muñoz et al. \(2020\)](#)) ou analyse générale bibliométrique et de performances, analyses par cartographie scientifique et librairies / packages. Le package {managHAL} se trouve dans les troisième et deuxième catégories. Il existe à ce jour d'autres outils d'analyses bibliométriques mais la majorité ne permettent pas de travailler via l'api HAL et se concentrent sur webofscience, scopus, google scholar,... Il existe des outils d'extraction de publications HAL, cependant aucun package R ne permet actuellement d'extraire ces données pour réaliser des analyses statistiques. Pour répondre à ce besoin, Théodore Vanrenghem a commencé à développer le package R {managHAL}. Le package permet l'extraction de données HAL et la création de bilans bibliographiques. En cours de développement, ce package ne permet pas actuellement la réalisation d'analyses bibliométriques.

Ainsi, durant mes 2 mois et 17 jours de stage au sein de l'unité mixte de recherche Mathématiques et Informatique Appliquées (MIA) - Paris Saclay, associée aux tutelles AgroParisTech, INRAE et Université Paris Saclay, j'ai travaillé sur rendre le package `{managHAL}` plus accessible et utilisable par d'autres unités de recherches que l'unité MIA - Paris Saclay. J'ai aussi travaillé sur la programmation et l'intégration d'outils d'analyse statistique de réseau au sein du package. Encadré par deux tuteurs, Julie Aubert et Pierre Barbillon, j'ai contribué à la mise en forme du package et à la création de fonctions spécifiques.

Pour comprendre les choix réalisés lors du développement du package, il est nécessaire de réaliser une introduction aux réseaux. Un réseau d'interaction est constitué de nœuds (les entités) et d'arêtes (les liens entre ces entités). Dans le cadre du package `{managHAL}`, j'utiliserai les réseaux pour modéliser un réseau de co-publications (ou co-auteurs). Les nœuds représenteront des auteurs, et les arêtes entre ces nœuds (auteurs) seront l'existence de publications scientifiques co-écrite par les deux nœuds. La taille d'un nœud représentera le nombre de publications écrites par cet auteur (nœud) et la largeur d'une arête pourra correspondre au nombre de publications co-écrites par les deux nœuds (co-auteurs). Les réseaux ne sont cependant pas exclusifs à l'analyse de données bibliométriques. En effet, les réseaux sont utilisés comme représentation et comme objet d'analyse statistique dans de nombreux domaines. Par exemple, des réseaux de régulations de gènes sont réalisés à partir de données de co-expression. Il est possible de donner d'autres exemples comme les analyses de réseaux d'interactions entre des espèces (réseau trophique) réalisées en écologie ([lien vers une analyse réalisée dans une vignette du package {sbm}](#)). Les utilisations des réseaux sont nombreuses. L'introduction aux réseaux et leurs analyses sera approfondie dans la section méthodes.

Ce package s'adresse donc à un public souhaitant réaliser des analyses bibliométriques approfondies à partir de HAL. Nous verrons cependant dans la section discussion qu'il est possible d'élargir les usages du package.

1.2 Méthodes

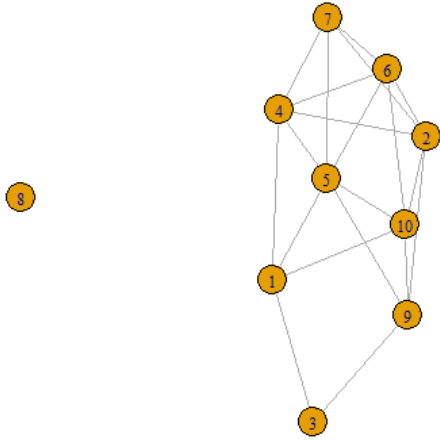
1.2.1 Modèle de graphe aléatoire

Cette partie est principalement basée sur la publication "Using Latent Block Models to Detect Structure in Ecological Networks" [Aubert et al. \(2022\)](#).

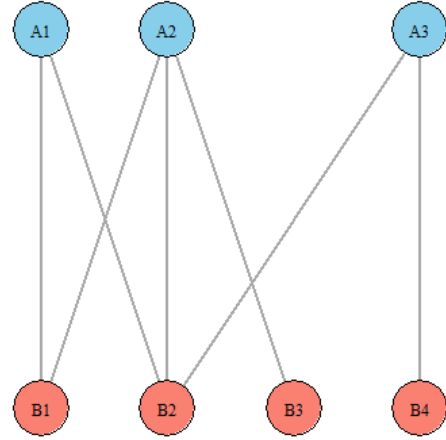
Les réseaux se divisent en deux catégories principales (voir figures suivantes): les réseaux Unipartites où tous les nœuds sont du même type et les réseaux Bipartites où il y a deux types distincts de nœuds, et les liens ne se forment qu'entre des nœuds de types différents. (voir [exemple_reseau_unipartite_bipartite](#) pour le code de réalisation des réseaux d'exemple)

Exemple des deux types de réseaux

Réseau Unipartite



Réseau Bipartite



Les interactions dans un réseau peuvent être binaires ou pondérées. Un réseau peut être représenté par une matrice d'adjacence Y où :

$$Y_{ij} \neq 0 \quad \text{si un lien existe entre les nœuds } i \text{ et } j,$$

$$Y_{ij} = 0 \quad \text{si aucun lien n'existe entre les nœuds } i \text{ et } j.$$

Nous décrivons les nœuds d'un réseau comme : $V = 1, \dots, N$, (N correspond à la taille du réseau). Les arrêtes peuvent être décrites comme $E = \{(i, j) | i, j \in V\}$ représentant les paires de nœuds entre lesquels il existe une interaction. Chaque arête (i, j) indique une connexion entre les nœuds i et j .

Il existe plusieurs approches différentes pour analyser la structure d'un réseau. Dans ce mémoire, une seule approche sera considérée. Elle suppose l'existence de groupe fonctionnels partageant un même patron de connexion. Nous nous baserons par la suite sur une méthode possible pour cette approche: les modèles probabilistes génératifs.

Les Modèles en Bloc Stochastiques (aussi appelé SBM) sont des modèles probabilistes. Ils supposent que les nœuds d'un même réseau sont divisés en blocs (groupe, clusters) latents regroupant les entités ayant des modèles de connectivité similaires. Latents signifie qu'il n'y a pas d'a priori sur le type de structure recherchée. Ces modèles sont utiles pour détecter les communautés, les clusters aux sein d'un réseau complexe. Pour réaliser un modèle en bloc stochastique, il est nécessaire de supposer que le réseau obtenu à partir de mes données est une réalisation de mon modèle.

Les nœuds sont partitionnés en K groupes latents. On définit un vecteur Z où $Z_i = k$, k est le bloc auquel appartient le nœud i . Dans le cas d'un réseau unipartite, on définit :

- $P(Z_i = k) = \pi_k$, comme la probabilité d'appartenance au groupe k .
- $P(Y_{ij} = 1 | Z_i = k, Z_j = k') = \gamma_{kk'}$, comme la probabilité pour une paire de nœuds appartenant à un bloc k et à un bloc k' d'être en interaction (dans le cas binaire).

Nous avons donc comme paramètres du modèle $\theta = (\gamma, \pi)$. A partir du réseau connu, il est possible d’inférer les paramètres du modèle. Par nécessité d’écourter le rapport, une explication détaillée ne sera pas réalisée. Je vous conseille cependant d’aller voir la publication “Using Latent Block Models to Detect Structure in Ecological Networks” Aubert et al. (2022) où les explications à ce propos sont claires et précises.

1.2.2 Méthodes informatiques

Comme précisé dans l’introduction, l’entièreté des méthodes informatiques développées ont été réalisées sur RStudio en utilisant le langage R version 4.4.1 sous windows 11 x64. Les packages utilisés sont les suivants :

Le package `{dplyr}` (Wickham et al. 2023) est utilisé pour la manipulation et la transformation des données grâce à une syntaxe claire et cohérente. Ce package facilite les opérations courantes telles que la sélection, le filtrage, le regroupement et le résumé des données. Le package `{mailR}` (Premraj 2021) est utilisé pour l’envoi d’emails avec des pièces jointes directement depuis R, ce qui est utile pour automatiser l’envoi de rapports. Le package `{askpass}` (Ooms 2023) est utilisé pour gérer les mots de passe de manière sécurisée. Le package `{furrr}` (Vaughan and Dancho 2022) est utilisé pour paralléliser les tâches en combinaison avec le package `future` (Bengtsson 2021). Le package `{future}` (Bengtsson 2021) permet l’exécution de code en parallèle, ce qui réduit le temps de traitement des tâches lourdes. Le package `{ggplot2}` (Wickham 2016) est utilisé pour la création de visualisations graphiques sophistiquées. Le package `{matgritr}` (Bache and Wickham 2022) fournit des opérateurs pour améliorer la lisibilité du code, en particulier l’opérateur `%>%` (pipe). Le package `{purrr}` (Wickham and Henry 2023) est utilisé pour les opérations fonctionnelles sur les listes et autres structures de données. Le package `{quarto}` (Allaire and Dervieux 2024) est utilisé pour la création de documents dynamiques et de rapports. Le package `{readxl}` (Wickham and Bryan 2023) permet la lecture de fichiers Excel. Le package `{rvest}` (Wickham 2024) est utilisé pour le web scraping, c’est-à-dire l’extraction de données à partir de pages web. Le package `{SnowballC}` (Bouchet-Valat 2023) est utilisé pour le traitement de la langue naturelle, en particulier pour la lemmatisation. Le package `{stringr}` (Wickham 2023) fournit des fonctions pour la manipulation des chaînes de caractères. Le package `{tictoc}` (Izrailev 2024) est utilisé pour mesurer le temps d’exécution des morceaux de code. Le package `{tm}` (Feinerer and Hornik 2024) est utilisé pour le text mining (extraction de connaissances à partir de textes). Le package `{wordcloud}` (Fellows 2018) est utilisé pour la création de nuages de mots. Le package `{Visnetwork}` (Chiquet, Donnet, and Barbillon 2020) est utilisé pour la visualisation de réseaux dynamiques. Le package `{Sbm}` (Chiquet, Donnet, and Barbillon 2024) est utilisé pour réaliser les analyses statistiques de réseaux.

Afin de pouvoir construire une requête HAL pour récupérer des données bibliographiques de publications d’auteurs, il est nécessaire de comprendre comment construire une requête. Pour construire une requête HAL, il faut au moins un paramètre `q` qui contient la requête de recherche. Ce paramètre doit spécifier le champ de recherche suivi de la valeur. Par exemple, pour rechercher le terme “test” : `http://api.archives-ouvertes.fr/search/?q=test&wt=xml` . Si le champ n’est pas spécifié, la recherche s’effectue par défaut dans l’index texte. Pour rechercher dans un champ particulier, il faut utiliser `champ:terme` : `http://api.archives-ouvertes.fr/search/?q=title_t:japon&wt=xml` . Pour échapper certains caractères spéciaux utilisés par Apache Solr, utilisez `\`. Par exemple, `(1+1):2` devient `\(1\+1\)\:2`. (voir “Documentation API-HAL” (n.d.)). Il est possible d’indiquer le format de la réponse (sous-entendant le format des données récupérées) via le paramètre `wt` . Pour finir il est possible de spécifier les champs à retourner dans la réponse via le paramètre `fl` et de faire un filtre des données via le paramètre `fq` (“Documentation API-HAL” (n.d.)). Tous les possibilités de l’API HAL n’ont pas été abordé dans ce court résumé. Il est pourtant compréhensible que l’API HAL est très diverse et permet

une grande liberté à l'utilisateur pour l'extraction des données. De plus, il existe un nombre de champs considérables. Certains seront abordés par la suite dans la section résultats.

Pour finir, l'entièreté du stage s'est inscrit dans une dynamique de développement et d'intégration continue via la forge gitlab d'INRAE. La forge gitLab, est un logiciel DevOps passant par le langage Git. Git est un système de controle, de versionning du code des développeurs gratuit et open source. Il permet à plusieurs développeurs de travailler sur un code sur un même dépôt et d'avancer ensemble par le biais de "merge". "Merge" signifie que la personne modifiant un fichier contenant du code pousse en ligne sur le depot partagé, les modifications réalisées.

1.3 Résultats

Cette section présente les résultats obtenus au cours de mon stage. Elle est divisée en deux parties principales: Les modifications au package {managHAL} d'origine dans l'objectif de le rendre plus générique et plus accessible, puis l'ajout de fonctionnalités au package {managHAL}. Par choix, seulement certaines fonctions et scripts R du package sont abordés. L'exécution de ces derniers est réalisée par le biais d'un exemple. Cet exemple est basé sur des données de l'UMR MIA-Paris-Saclay. Chaque fonction est expliquée en détail, suivie des raisons justifiant son développement. Il est important de noter que mon stage se terminant le 26 juillet, Le package est encore en développement. Certaines fonctionnalités sont encore en travail. Les fonctions présentées fonctionnent dans le cadre de l'exemple. Si vous souhaitez executer les scripts fournis dans le mémoire, il est nécessaire d'avoir accès au package {managHAL}. Dans cette section, les noms de *fonctions* seront en italique et les noms des **scripts R** contenant les fonctions seront en gras.

1.3.1 Accessibilité et genericité de {managHAL}

Lors de mon stage, j'ai d'abord décidé de rendre le package plus accessible et générique. Pour ce faire, il m'était nécessaire de réorganiser et modifier certaines fonctions déjà présentes codées par Théodore Vanrengterghem.

1.3.1.1 Modifications de la fonction *load_mia_table*

J'ai commencé par modifier dans le fichier **mia_table.R** la fonction *load_mia_table*. A l'origine, cette fonction permettait le chargement d'un csv en ligne fourni par les Ressources Humaines depuis un serveur. Le fichier contient des informations tel que le nom, le prenom, l'idhal, l'equipe, etc, des personnes appartenant au labo. Cette fonction n'était pas générique. Ainsi, il était nécessaire de la modifier. J'ai fait le choix de renommer le fichier **load_table.R**. J'ai remplacé la fonction *load_mia_table* par trois fonctions *load_team_table_csv*, *load_team_table_url*, et *load_team_table*. J'ai aussi modifier la majorité des fonctions déjà présentes dans le package faisant appel à *load_mia_table*. Les paramètres en entrée des trois fonctions sont les même que la fonction *load_mia_table* d'origine avec l'ajout d'un paramètre permettant à l'utilisateur de rentrer l'adresse du fichier. J'ai ensuite crée une documentation via le package "Roxygène2" conforme aux critères requis pour un package mis en ligne (veuillez vous referez à l'annexe : [Exemple d'une documentation classique avec Roxygen2](#)) pour chacune des trois fonctions.

load_team_table_url permet de charger un csv dans Rstudio depuis un fichier en ligne. J'ai décidé pour *load_team_table_url* de garder en majeure partie le code d'origine et de simplement le réadapter par soucis de temps. J'y ai ajouté des conditions de présence de colonnes obligatoires afin d'obliger

l'utilisateur à fournir des colonnes sous un format spécifique. (Pour voir le corps de la fonction, veuillez vous référer à l'annexe : [load_team_table_url](#))

`load_team_table_csv` est basé sur la construction de `load_team_table_url`. Je l'ai adapté pour pouvoir charger depuis un fichier local de la machine de l'utilisateur en utilisant `read.csv`. De même que dans `load_team_table_url`, j'y ai ajouté des conditions de présence de colonnes obligatoires afin d'obliger l'utilisateur à fournir des colonnes sous un format spécifique. (Pour voir le corps de la fonction, veuillez vous référer à l'annexe : [load_team_table_csv](#))

Enfin, `load_team_table` appelle l'une des deux fonctions en fonction du type d'adresse fourni : URL ou chemin de fichier local. Le type de l'adresse est déterminé en utilisant `grepl`, qui utilise une expression régulière. (Pour voir le corps de la fonction, veuillez vous référer à l'annexe : [load_team_table](#))

1.3.1.2 Résultat de l'Application de `load_team_table`

```
data_RH_csv_Example <- load_team_table(filter_id = TRUE,
  date_cols = c(6, 7), filepath_or_url = "./data/Data_RH_Example.csv")
# afin de ne garder que les lignes avec un idhal
# correct
data_RH_csv_Example <- data_RH_csv_Example[which(!is.na(data_RH_csv_Example$idhal)),
]
head(data_RH_csv_Example)
```

	civilite	nom	prenom	statut	rattachement	debut_contrat	fin_contrat
1	M.	ADJAKOSSA	Éric	CEC	AgroParisTech		
6	Mme	AUBERT	Julie	IR	INRAE		
8	M.	BARBILLON	Pierre	PR	AgroParisTech		
	financement	equipe	unite		orcid	idhal	
1		SOLsTIS	MIA	PS	0000-0002-5280-0347	749339	
6		SOLsTIS	MIA	PS	0000-0001-5203-5748	15356	
8		SOLsTIS	MIA	PS	0000-0002-7766-7693	16750	
	adresse_mail						
1	eric-houngla.adjakossa@inrae.fr						
6	julie.aubert@inrae.fr						
8	Pierre.Barbillon@Agroparistech.fr						

Lors de l'application de la fonction `load_team_table` ci-dessus en utilisant le jeu de données test `Data_RH_Example`, on observe le chargement d'un csv contenant les colonnes d'origines du csv fournie en entrée. Les valeurs manquantes ou mal renseignées des idhals numériques ont été remplacées par des NAs. Les dates présentes deans les colonnes de dates spécifiés par l'utilisateur ont été nettoyées et ont le bon format.

1.3.1.3 Modifications des fonctions régissant la requête HAL

Dans le package `{managHAL}` originel, il y avait deux fichiers contenant les fonctions nécessaires à la construction d'une requête HAL. Ces deux fichiers se nomment **HAL_reports.R** et **HAL_queries.R**. La fonction principale de construction d'une requête HAL est `HAL_query`. Elle prend en entrée différents

paramètres et construit une url correspondant à une requête HAL. Ces différents paramètres définissent les différents champs, filtres, sorties demandées de la requête. Les champs de sortie ne convenant pas et manquait d'information pour la construction d'un réseau. Ainsi, après réflexion et discussion avec mes tuteurs Julie Aubert et Pierre Barbillon, j'ai modifié les différents champs de sorties afin d'obtenir les données nécessaires et essentielles à la construction d'un réseau. Par exemple, le champ 'authFullNamePersonIDIDHal_fs' permet de spécifier à HAL_query de récupérer les noms et les identifiants HAL numériques des auteurs d'une même publication. Le paramètre add_output ajouté par Théodore Vangrangerghem dans la fonction d'origine permet à l'utilisateur de rentrer des champs de sortie spécifiques en plus. Ce paramètre a été beaucoup utilisé au cours de ce stage dans la création de fonction permettant la création de réseau.

La fonction d'origine ne crée pas d'url d'une page permettant de récupérer les publications autres qu'à partir d'identifiants numériques d'auteurs, j'ai modifié les paramètres d'entrée et le corps de la fonction afin que l'utilisateur puissent spécifier si les identifiants fournis sont des identifiants de structures ou des identifiants de personnes. La nouvelle fonction HAL_query crée donc un url qui peut donc récupérer au choix de l'utilisateur une liste de publications associés à une structure via son identifiant ou une liste de publications associés à un identifiant numérique. (Pour voir le corps de la fonction, veuillez vous référer à l'annexe : [HAL_query](#))

Plusieurs fonctions présentes avant mes modifications faisait appel à HAL_query. Ces dernières comme HAL_extract_csv ont donc été modifiées afin de correspondre à la nouvelle version de HAL_query. De même Les fonctions auxquelles fait appel HAL_query ont elles aussi été modifiées pour correspondre à HAL_query. HAL_extract_csv est une fonction récupérant l'url via HAL_query. Pour montrer les résultats obtenus avec HAL_query, des exemples ci-dessous seront réalisés avec HAL_extract_csv pour une liste d'identifiants numérique d'auteurs et pour un identifiant d'une unité/laboratoire. Il sera ajouté des sorties via le paramètres add_outputs :

```
date_min = "01/01/2022"
date_max = "01/01/2023"

# Publications à partir d'un id de laboratoire.
HAL_publis_Labo <- managHAL::HAL_extract_csv(id = 1002311,
  date_min, date_max, type_id = "struct_id", add_output = c("structAcronym_s",
    "structId_i", "structHasAlphaAuthIdHalPersonid_fs"))
HAL_publis_Labo[1, ]
```

	docid	halId_s	version_i	docType_s
1	4440523	hal-04440523	1	COMM

1	Isabelle Lebert, Maxime Rates, Julien Pradel, Laure Mathews-Martin, Aurore Latour, et al.. Evaluation de la citationRéf
---	---

1	Colloque final de restitution du projet Biodiversa-Bioroddi, Nov 2023, Lyon (Sciences Po), France
	publicationDate_tdate
1	2023-01-01T00:00:00Z

1	Isabelle Lebert_FacetSep_184610_FacetSep_isabelle-lebert,Maxime Rates_FacetSep_749136_FacetSep_184610_FacetSep_isabelle-lebert
---	--

1	UMR EPIA,VAS,INRAE,UMR CBGP,Cirad,IRD [France-Sud],INRAE,Institut Agro,UM,LRFSN,ANSES,MIA Paris
---	---

1	1002382,301767,577435,1100832,11574,451860,577435,1096330,1042499,1100589,561191,301715,1002311
---	---

```
1 L_AlphaSep_1002382_FacetSep_Unité Mixte de Recherche d'Épidémiologie des maladies Animales et z
```

```
# Publications à partir d'ids auteurs
HAL_publis_auteurs <- managHAL::HAL_extract_csv(id = data_RH_csv_Example[,
  12], date_min, date_max, type_id = "person_id")
HAL_publis_auteurs[1, ]
```

```
      docid      halId_s version_i docType_s
1 4311126 hal-04157375      1      ART
```

```
1 Joanna Moro, Gaëtan Roisné-Hamelin, Nadezda Khodorova, Douglas Rutledge, Jean-Charles Martin, e
      citationRef_s
```

```
1 Journal of Nutrition, 2023, 153 (9), pp 2571-2584. 10.1016/j.tjnut.2023.06.039
  publicationDate_tdate
1 2023-01-01T00:00:00Z
```

```
1 Joanna Moro_FacetSep_0_FacetSep_,Gaëtan Roisné-Hamelin_FacetSep_0_FacetSep_,Nadezda Khodorova_F
```

1.3.2 Ajout de fonctionnalité à {managHAL}

1.3.2.1 Création d'un réseau de co-auteurs et visualisation avec visNetwork

Afin d'obtenir les informations sous-jacentes des informations bibliographiques, j'ai commencé à travailler à la création d'un réseau de co-auteur. Pour l'instant, la fonctionnalité n'existe qu'au format de script R sur un document quarto. Au cours des semaines restantes, l'objectif est de continuer à travailler dessus et créer des fonctions simples et claires permettant à l'utilisateur de créer son propre réseau à partir de ses données. Le script de création de fonction est inspiré d'un script fournie au préalable par ma tutrice Julie Aubert. Des modifications et des ajouts ont été apportés pour améliorer l'extraction, le nettoyage, la construction, et la visualisation des données. Suite à mes modifications et mes ajouts, le script se divise en plusieurs parties ((Pour voir le script R, veuillez vous référer à l'annexe : [Script_R_reseau_co-auteur](#)) :

La première partie correspond à l'extraction et au nettoyage des données. Les données brutes ont été organisées et nettoyées pour être adaptées à la construction du réseau. Les années de publication ont été extraites et les identifiants des auteurs ont été nettoyés. Les auteurs ont été associés aux structures pertinentes, et les incohérences de noms ou d'identifiants HAL ont été corrigées. Des identifiants fictifs ont été créés pour les auteurs sans identifiant.

La deuxième partie concerne la construction et la visualisation du graphe. À partir des données nettoyées, des matrices de contingence ont été créées. Ces matrices ont été converties en listes de nœuds (auteurs) et en listes d'arêtes (collaborations). Les nœuds ont été colorés et groupés en fonction des années de publication, et des légendes ont été ajoutées pour faciliter la compréhension du graphe. Des descriptions supplémentaires ont été intégrées pour enrichir les légendes, et la visualisation finale du réseau a été réalisée via visnetwork.

Il est possible de voir via l'exemple suivant (obtenu avec le code de l'annexe : [Script_R_reseau_co-auteur](#)) qu'un réseau de co-auteurs permet une visualisation claire des collaborations et des périodes d'activité des différents auteurs.

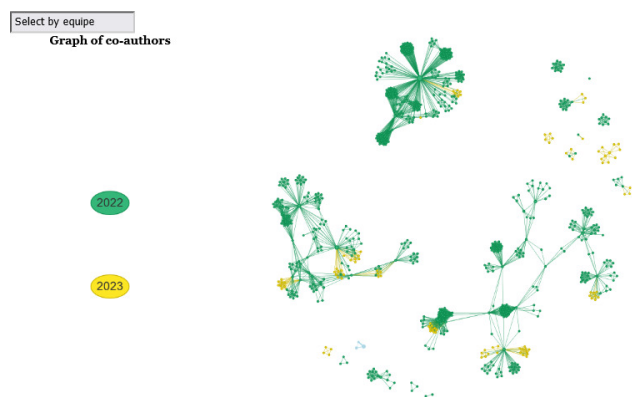


Figure 1.1: Réseau de co-auteurs

A travers cet exemple, les limitations de la création de réseau sur les données récupérées via un identifiant de laboratoire transparaissent. De fait, on voit l'apparition de cercle de co-auteurs très denses. Ces cercles correspondent généralement à une unique publication contenant de nombreux auteurs.

1.3.2.2 Modélisation en blocs stochastiques

J'ai tout de même réalisé un modèle en bloc stochastique sur ce réseau. (voir le code du script en annexe :[Script_R_SBM](#)). Pour cela j'ai créé une matrice d'adjacence vide et je l'ai remplie avec les connexions pondérées des nœuds, puis nettoyée pour la lisibilité et visualisée avec plotMyMatrix. Ensuite, un modèle SBM est estimé à partir de cette matrice, les nœuds sont classés en groupes colorés. Enfin, le réseau est visualisé avec visNetwork, incluant des options pour une meilleure compréhension et esthétique.

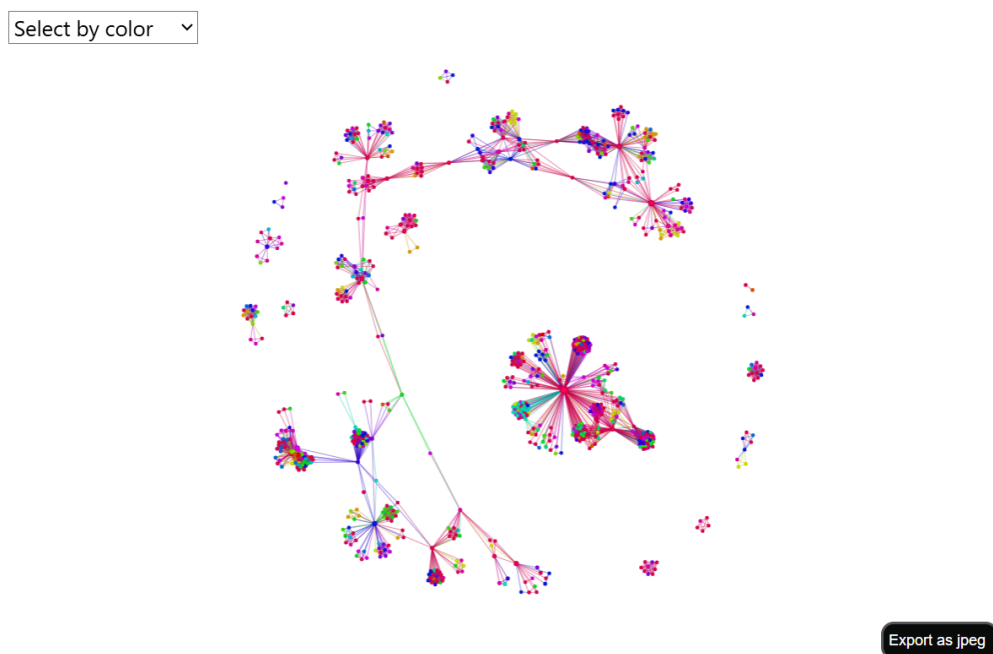


Figure 1.2: Application des couleurs d'appartenance des noeuds via le modèle

L'apparition de groupes de noeuds avec des interactions similaires colorés était attendue. Or dans l'exemple suivant, des groupes d'une même couleur n'apparaissent pas et les noeuds avec des profils de connexion similaire ne semble pas être placé dans un même groupe. C'est pourquoi, j'ai décidé sur conseil de mes tuteurs de réaliser les mêmes manipulations sur un sous-groupe de données ne contenant que celles des auteurs affiliés à une seule structure. Nous avons noté la présence d'un biais car certains auteurs n'étant plus affiliés à la structure ont toujours le tag de la structure dans leur compte HAL. Ils apparaissent affiliés à la structure sans l'être réellement. (voir le code du script en annexe :[Script_R_Sous-groupe](#))

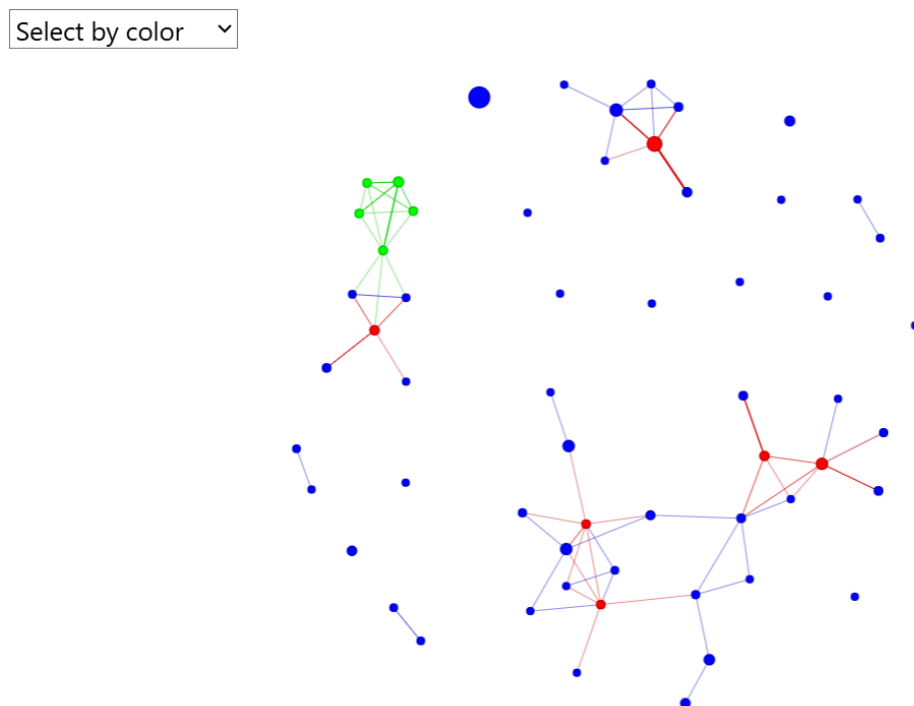


Figure 1.3: Application des couleurs d'appartenance des noeuds via le nouveau modèle sur le réseau de co-auteurs de l'UMR MIA - Paris Saclay

De plus, le sbm obtenu n'étant toujours pas satisfaisant, une autre proposition a été mise en avant, se contenter de réaliser un sbm sur les composantes connexes du réseau. Cela n'a pas été encore réalisé et fait partie des possibles améliorations future du package {managHAL}.

1.4 Discussion

Au cours de cette partie, plusieurs sujets seront abordés. Les limites actuelles du package {managHAL} et les possibles améliorations seront discutées. Les résultats obtenus seront replacés dans un contexte scientifique plus large. Enfin, les utilisations potentielles du package {managHAL} seront examinées.

Premièrement, il est important de parler des limites rencontrées au cours de ce projet. Un des défis a été l'intégration de l'API HAL. Malgré mes efforts, l'API HAL s'est révélée complexe et souvent peu intuitive. En effet, il existe des centaines de champs pour récupérer des données sur l'API HAL. Cela m'a freiné dans la compréhension de l'API et dans la modification des fonctions utilisant cette dernière. Les erreurs rencontrées étaient souvent dues à une mauvaise récupération des données ou à une confusion de ma part

par rapport au formatage de ces dernières. J'ai donc eu besoin de nombreuses heures de recherches, de débogage et de tests pour obtenir les résultats attendus.

De plus, bien que le code développé ait montré une certaine efficacité en terme de temps de calcul, il reste des marges d'amélioration. Par exemple, certaines parties du code pourraient être optimisées pour améliorer les performances des fonctions et réduire le temps de traitement. Remplacer les boucles `for` par `apply`, qui est optimisée et plus rapide, pourrait être une solution. De même, pour la création de réseaux et l'utilisation de modélisation en bloc stochastique, il est encore possible d'améliorer le temps de traitement en recherchant et modifiant les options des fonctions utilisées ou en créant des sous-groupes comme présenté dans les résultats.

Une des ambitions de ce projet est de s'inscrire dans une dynamique de science ouverte. Le mémoire écrit et produit avec Quarto illustre cette démarche en rendant le mémoire et les résultats accessibles et reproductibles sur un répertoire GitHub ([repositoryMémoire](#)). S'il était possible de lier les résultats obtenus à des résultats provenant d'autres bases de données telles que WebOfScience, la valeur scientifique des recherches effectuées via le package `{managHAL}` pourrait être considérablement enrichie. Avant la fin du stage, je prévois de travailler sur la documentation et les tests. L'objectif est d'améliorer la documentation du code pour faciliter son utilisation par d'autres développeurs, de documenter les fonctions pour en faciliter l'utilisation par d'autres personnes, et d'ajouter des tests pour garantir la bonne fonctionnalité du package. Je prévois aussi d'ajouter de nouvelles fonctionnalités. Dans un premier temps, je souhaite coder les fonctions et sous-fonctions permettant de construire un réseau de co-auteurs plutôt que d'avoir un simple script R sur un document quarto. Puis, je souhaite coder les fonctions permettant d'obtenir un modèle en bloc stochastique via le package 'sbm'. Cela permettrait de simplifier l'utilisation du package et de le rendre plus générique. Dans un second temps, je prévois de nettoyer le package entièrement et de vérifier qu'il est bien aux normes pour qu'il puisse être publié à la fin de mon stage.

En ce qui concerne les utilisations potentielles du package développé, en premier lieu, `{managHAL}` est destiné à être utilisé par les personnes voulant réaliser des rapports sur les dynamiques sous-jacentes des publications au sein d'un laboratoire. Ensuite, il est possible qu'il soit utilisé par les personnes souhaitant faire un rapport bibliographique ou réaliser des vérifications quant aux erreurs d'ajout de publications des chercheurs sur HAL. Enfin, il y a des parallèles intéressants à explorer avec d'autres domaines scientifiques, comme la biologie. Les méthodes utilisées pour l'analyse de réseaux sont génériques et utilisables sur d'autres types de réseaux, et les fonctions de visualisation de réseaux par exemple peuvent être utilisées pour d'autres utilisations telle la visualisation d'un réseau de régulation de gène.

En conclusion, ce stage a permis de réaliser des avancées significatives dans le développement de l'outil qu'est le package `{managHAL}` malgré les défis rencontrés. Les prochaines étapes incluront des optimisations et des développements supplémentaires pour renforcer l'impact et la portée du travail réalisé, tout en restant alignées avec les principes de la science ouverte. Cette démarche, combinée à une réflexion continue sur les objectifs et les résultats, permettra non seulement d'améliorer la qualité du travail actuel, mais aussi de poser les bases pour l'entretien du package par les futurs développeurs.

1.5 Annexes

1.5.1 load_team_table_url

```
load_team_table_url <- function(filter_id = TRUE, date_cols,
  url) {
  idhal <- NULL
  tmp_file <- paste0(tempfile(), ".xlsx")
  download.file(url = url, destfile = tmp_file, mode = "wb")
  nbcol <- ncol(readxl::read_xlsx(tmp_file, sheet = "Membres"))
  col_types <- rep("text", nbcol)
  col_types[date_cols] <- "date"

  table <- readxl::read_xlsx(tmp_file, sheet = "Membres",
    col_types = col_types) %>%
    dplyr::rename_with(clean_names) %>%
    dplyr::mutate_if(function(x) {
      "POSIXt" %in% class(x)
    }, ~format(.x, "%d/%m/%Y"))

  if (filter_id) {
    table <- suppressWarnings(table %>%
      dplyr::filter(!is.na(as.numeric(idhal))) %>%
      dplyr::mutate(idhal = as.numeric(idhal)))
  }

  # Test if there is a column named 'id'
  # Possibility to add mandatory columns in the
  # future Just add 'OR'
  if (!(("idhal" %in% names(table)) | !("nom" %in%
    names(table)) | !("prenom" %in% names(table)))) {
    stop("one of the following columns are missing : 'idhal' ; 'nom' ; 'prenom' ")
  }

  return(table)
}
```

Cette fonction permet le chargement d'un fichier csv depuis un url. Elle prend en entrée une adresse url, un booléen déterminant le nettoyage ou non des valeurs idHALs, et un vecteur contenant le numéro des colonnes de dates. Elle donne en sortie le chargement d'un fichier csv.

1.5.2 load_team_table_csv

```
load_team_table_csv <- function(filter_id = TRUE,
  date_cols,
  filepath){
```

```

idhal <- NULL

# read the csv table  TODO : Add message error to guide user
# if sep is different than ";"
# and if header = false
nbcol <- ncol(read.csv(filepath, header = TRUE, sep = ";", encoding = "UTF-8"))

# define column types (dates or text if not dates)
col_types <- rep("character", nbcol)
col_types[date_cols] <- 'character'

table <- read.csv(filepath,
                  header = TRUE,
                  sep = ";",
                  encoding = "UTF-8",
                  colClasses = col_types
) %>% dplyr::rename_with( # clean the columns names
  clean_names
) %>%
  dplyr::mutate_if(function(x) { # Convert specified column to dates
    "POSIXt" %in% class(x)
  }, ~ format(.x, "%d/%m/%Y"))

# Filter out rows without IDHAL
if (filter_id) {
  idhal <- table$idhal
  table <- table[which(!is.na(idhal)),]
  table$idhal <- as.numeric(idhal)
  table$idhal[table$idhal == 0] <- NA
  table <- table[which(!is.na(idhal)),]
}

# Test if there is a column named "id"
# Possibility to add mandatory columns in the future
# Just add "OR"
if ( !("idhal" %in% names(table)) | !("nom" %in% names(table)) | !("prenom" %in% names(table))

return(table)
}

```

Cette fonction permet le chargement d'un fichier csv depuis une adresse locale de la machine de l'utilisateur. Elle prend en entrée une adresse locale d'un fichier csv, un booléen déterminant le nettoyage ou non des valeurs idHALs, et un vecteur contenant le numéro des colonnes de dates. Elle donne en sortie le chargement d'un fichier csv.

1.5.3 load_team_table

```
load_team_table <- function(filter_id = TRUE, date_cols = c(7,
  8), filepath_or_url) {

  # If the parameter is a URL
  if (grepl("^https?:/", filepath_or_url)) {

    # Call the load_team_table_url function
    # with the URL
    return(load_team_table_url(filter_id = filter_id,
      date_cols = date_cols, url = filepath_or_url))
  } else {

    # Otherwise, it's a local file path; call
    # the load_team_table_csv function with
    # the local file path
    return(load_team_table_csv(filter_id = filter_id,
      date_cols = date_cols, filepath = filepath_or_url))
  }
}
```

Cette fonction permet le chargement d'un fichier csv depuis une adresse locale de la machine de l'utilisateur ou depuis une adresse url. Elle prend en entrée une adresse locale d'un fichier csv ou une URL, un booléen déterminant le nettoyage ou non des valeurs idHALs, et un vecteur contenant le numéro des colonnes de dates. Elle donne en sortie le chargement d'un fichier csv.

1.5.4 Exemple d'une documentation classique avec Roxygen2

```
#' load_team_table_csv
#'
#' This function loads a team table from a CSV file, allowing optional filtering
#' based on the presence of an IDHAL and specifying which columns contain dates.
#'
#' @param filter_id erase person without IDHAL (default = TRUE)
#' @param date_cols positions of dates columns, others will be read as text
#' @param filepath the file path of the CSV table
#'
#' @export
#'
#' @importFrom magrittr %>%
#' @importFrom utils read.csv
#'
#' @return a data.frame containing information from team
#'
#' @examples
#'
```



```
#' \dontrun{
#' load_team_table_csv(filter_id = TRUE,
#'                      date_cols = c(7,8),
#'                      'C:/users/.../.../Classeur.csv')
#' }
```

Ce script permet la création d'une documentation de fonction classique via la commande : `devtools::document()`. Elle est au format utilisé par {roxygen2}.

1.5.5 HAL_query

```
HAL_query <- function(id,
                      date_min = NULL,
                      date_max = NULL,
                      format = c("csv", "bibtex", "json", "xml", "online"),
                      grouped = FALSE,
                      type_id = c("person_id", "struct_id"),
                      maxrows = 1000,
                      add_filters = list(),
                      add_exclusions = list(
                        status_i = 111,
                        instance_s = c("sfo", "dumas", "memsic", "hceres")
                      ),
                      add_outputs = character(0),
                      sorted_by = c("producedDate_tdate", "desc"),
                      thesis_strict = T) {

  # ----- DEFAULT PARAM -----
  online <- format[[1]] == "online"
  outputs <- c(
    "docid", "halId_s", "version_i", "docType_s", "citationFull_s",
    "citationRef_s", "publicationDate_tdate", "authFullNamePersonIDIDHal_fs", add_outputs
  )
  filters <- list(
    docType_s = c("COMM", "ART", "OUV", "COUV", "DOUV", "POSTER", "SOFTWARE", "THESE", "HDR")
  )
  filters[names(add_filters)] <- add_filters

  exclusions <- list()
  exclusions[names(add_exclusions)] <- add_exclusions
  # -----

  # Url start
  if (online) {
    start_query <- "https://hal.archives-ouvertes.fr/search/index/?"
  } else {
    start_query <- paste0(
```

```

    "https://api.archives-ouvertes.fr/search/hal/", # API HAL
    "?omitHeader=true", # header omitted
    "&wt=", format[[1]], "&" # csv, bibtex, ...
  )
}
if (grouped) {
  date_query <- query_date_grouped_parsing(
    id = id,
    type_id = type_id,
    date_min = date_min,
    date_max = date_max,
    online = online
  )
} else {
  date_query <- query_date_ungrouped_parsing(
    id = id,
    type_id = type_id,
    date_min = date_min,
    date_max = date_max,
    online = online
  )
}

filter_query <- query_filter_parsing(
  filters = filters, exclusions = exclusions,
  online = online
)
if (thesis_strict & !online) {
  add <- "&fq=NOT+(docType_s:(THESE+OR+HDR)+AND+submitType_s:(notice+OR+annex))"
} else {
  add <- ""
}

sorting_query <- query_sort_parsing(sorted_by = sorted_by)
output_query <- query_output_parsing(
  outputs = outputs, n_row = maxrows,
  online = online
)
obj <- list(
  urls = paste0(
    start_query,
    date_query,
    filter_query,
    add,
    sorting_query,
    output_query
  ) %>%
  utils::URLencode(),
  description = list(

```

```

    idhal = id,
    date_min = date_min,
    date_max = date_max,
    outputs = outputs,
    format = format[[1]],
    grouped = grouped,
    maxrows = maxrows,
    filters = filters,
    exclusions = exclusions,
    outputs = outputs,
    sorted_by = sorted_by,
    thesis_strict = thesis_strict
  )
)
class(obj) <- "halUrl"
return(obj)
}

```

Cette fonction permet de créer une adresse url d'une requête à l'API HAL. Elle prend en entrée différents paramètres de spécification du rendu. Elle donne en sortie une URL.

1.5.6 Script_R_reseau_co-auteur

```

#### ----- Creation of a co-authors
#### graph ----- ####

#### ----- First part : Data extraction and
#### cleaning

publication_id_struc <- publication_id_struc %>%
  mutate(Publication_Year = substr(publicationDate_tdate,
    1, 4))

for (i in seq_along(publication_id_struc$structHasAlphaAuthIdHalPersonid_fs)) {
  publication_id_struc$structHasAlphaAuthIdHalPersonid_fs[i] <- gsub("\\\\",
    "", publication_id_struc$structHasAlphaAuthIdHalPersonid_fs[i])
}

# extraction of info on the authors
data_graph_authors <- publication_id_struc %>%
  select(halId_s, Publication_Year, authFullNamePersonIDIDHal_fs) %>%
  separate_rows(authFullNamePersonIDIDHal_fs, sep = ",") %>%
  separate(authFullNamePersonIDIDHal_fs, into = c("Full_Name",
    "Person_ID", "idhal"), sep = "_FacetSep_",
    fill = "right") %>%
  mutate(across(c(Full_Name, Person_ID, idhal), as.character)) %>%
  mutate(Person_ID = ifelse(Person_ID == "0", NA,

```

```

    Person_ID))

# extraction of info on which authors belongs to
# which structure at the times of the
# publications
data_graph_author_struc <- publication_id_struc %>%
  select(structHasAlphaAuthIdHalPersonid_fs) %>%
  separate_rows(structHasAlphaAuthIdHalPersonid_fs,
    sep = "_AlphaSep_") %>%
  separate(structHasAlphaAuthIdHalPersonid_fs, into = c("id_name_struc",
    "id_full_name"), sep = "_JoinSep_") %>%
  separate(id_full_name, into = c("idhal_s", "Person_ID",
    "Full_Name"), sep = "_FacetSep_", fill = "right") %>%
  separate(id_name_struc, into = c("id_struc", "name_struc"),
    sep = "_FacetSep_", fill = "right") %>%
  separate(Full_Name, into = c("Full_Name", "First_Letter"),
    sep = ",", fill = "right") %>%
  mutate(Person_ID = ifelse(Person_ID == "0", NA,
    Person_ID))
## data_graph_author_struc <-
## find_inconsistent_names_and_ids(data_graph_author_struc)
data_graph_authors <- find_inconsistent_names_and_ids(data_graph_authors)
data_graph_authors <- merge(data_graph_authors, data_graph_author_struc[,
  c("Person_ID", "id_struc")], by = "Person_ID",
  all.x = TRUE)

# Creation of false ids only for authors without
# Person_ID adding a mark to know which one were
# created and which were not
data_graph_authors$False_id_mark <- !is.na(data_graph_authors$Person_ID)
unique_authors_without_id_vec <- unique(data_graph_authors$Full_Name[which(data_graph_authors$False_id_mark
  FALSE)]) ##%>%
## distinct(Full_Name, .keep_all = TRUE) ##%>%
## filter(is.na(Person_ID))

# to create my own ids
creation_id <- function(unique_authors_no_id_name_column) {
  ids <- character(length(unique_authors_no_id_name_column))
  count <- 0
  for (i in 1:length(unique_authors_no_id_name_column)) {
    count <- count + 1
    ids[i] <- count
  }
  return(ids)
}
unique_authors_without_id <- data.frame(Full_Name = unique_authors_without_id_vec)
unique_authors_without_id$False_Id = creation_id(unique_authors_without_id$Full_Name)

```

```

# delete lines where False_Id equals Person_ID i
# lose info here but one person out of a
# thousands seems ok (to be tested)
unique_authors_without_id <- unique_authors_without_id %>%
  filter(!False_Id %in% data_graph_authors$Person_ID)

# give a false id to all authors without person
# id
data_graph_authors <- data_graph_authors %>%
  left_join(unique_authors_without_id %>%
    select(Full_Name, False_Id), by = "Full_Name") %>%
  mutate(Person_ID = ifelse(is.na(Person_ID), False_Id,
    Person_ID)) %>%
  select(-False_Id, -idhal)

#### ----- Second part : Constructing and
#### visualizing networks

# Creating the 'edge list' and the 'node list' To
# do so we first create the contingency matrix,
# and then we applied the melt function
contingence_table = data_graph_authors %>%
  count(as.numeric(Person_ID), halId_s) %>%
  dplyr::select(-n) %>%
  table %>%
  as.matrix
contingence_matrix = contingence_table %*% t(contingence_table)
contingence_matrix[lower.tri(contingence_matrix)] = 1000 # We set 1000 in order to applied a fil
melt_matrix = melt(contingence_matrix)
colnames(melt_matrix) = c("from", "to", "value")
edge_list = melt_matrix %>%
  filter(from != to & !value %in% c(0, 1000)) %>%
  rename(width = value)
node_list = melt_matrix %>%
  filter(from == to) %>%
  dplyr::select(-to) %>%
  rename(id = from) %>%
  mutate(label = id)

# Creating a group variable in the node_list in
# order to set differents colors corresponding
# with the different years
data_graph_authors$Publication_Year <- as.numeric(as.character(data_graph_authors$Publication_Yea
df_group = data_graph_authors %>%
  summarise(group = as.character(floor(mean(Publication_Year))),
    .by = Person_ID) %>%
  rename(id = Person_ID)

```

```

node_list$id <- as.character(node_list$id)
node_list = node_list %>%
  left_join(df_group, by = "id")

# Defining nodes and edges that will serve to
# build a legend for edges and nodes sizes
# (number of publications), and colors (years)
range_year = range(data_graph_authors$Publication_Year)
node_size = data.frame(id = c("A", "B", "C"), value = c(1,
  6, 15), label = c("A", "B", "C"), group = "Number of publications")
edge_size = data.frame(from = c("A", "B", "C"), to = c("B",
  "C", "A"), width = c(1, 6, 15), label = c("1",
  "6", "15"))
node_colors = data.frame(id = as.character(range_year),
  value = 1, label = as.character(range_year), group = as.character(range_year))
edge_colors = data.frame(from = as.character(range_year),
  to = as.character(range_year[c(2, 1)]), width = 1,
  label = "")
node_list = rbind(node_list, node_size, node_colors)
edge_list = rbind(edge_list %>%
  mutate(label = ""), edge_size, edge_colors)
edge_list = edge_list %>%
  mutate(size = 20)

# Defining colors for Publication year
df_col = data.frame(group = as.character(c(range_year[1]:range_year[2])),
  color = colorRampPalette(colors = c("#35B779FF",
    "#FDE725FF"))(range_year[2] - range_year[1] +
    1))
node_list = node_list %>%
  left_join(df_col, by = "group")

# Legend nodes Defining colors for Publication
# year
df_col_Legend = data.frame(label = as.character(c(range_year[1]:range_year[2])),
  group = as.character(c(range_year[1]:range_year[2])),
  color = colorRampPalette(colors = c("#35B779FF",
    "#FDE725FF"))(range_year[2] - range_year[1] +
    1))

# legend additional information table
node_info <- data.frame(id = c("A", "B", "C"), info = c("Additional info for node A",
  "Additional info for node B", "Additional info for node C"))

# Join node_info with node_list to add the title
# column to the legend nodes
node_list <- node_list %>%
  left_join(node_info, by = "id") %>%
  mutate(title = ifelse(!is.na(info), info, "")) %>%

```

```

    select(-info) # remove the info column after using it to create the title

# Convert idhal to character type
cleaned_table <- cleaned_table %>%
  mutate(idhal = as.character(idhal))

tmp2 <- data_graph_authors[data_graph_authors$Person_ID %in%
  node_list$id, ]
node_info2 <- tmp2 %>%
  select(Full_Name, Person_ID) %>%
  rename(id = Person_ID)

# Constructing node_info3
tmp3 <- cleaned_table[cleaned_table$idhal %in% node_list$id,
  ]
node_info3 <- tmp3 %>%
  select(idhal, nom, prenom, unite, equipe) %>%
  rename(id = idhal)

# Joining node_info2 with node_list to add the
# title column
node_list <- node_list %>%
  left_join(node_info2, by = "id") %>%
  mutate(title_info = ifelse(!is.na(Full_Name), paste("Name:",
    Full_Name, ",", "numeric idHal:", id), NA),
    title = ifelse(title == "", title_info, title))

# Joining node_info3 with node_list to add the
# title column
node_list <- node_list %>%
  left_join(node_info3, by = "id") %>%
  mutate(title_info = ifelse(!is.na(nom), paste("Name:",
    paste(nom, prenom), ",", "Team:", equipe, ",",
    "Unit:", unite, ",", "numeric idHal:", id),
    title_info)) %>%
  mutate(title = if_else(!is.na(nom), title_info,
    title)) %>%
  select(-title_info)

node_list <- node_list %>%
  distinct(node_list$id, .keep_all = TRUE)

# Network visualization
visNetwork(node_list, edge_list, width = "100%") %>%
  visIgraphLayout() %>%
  visGroups(groupname = "Number of publications",
    color = "lightblue", font = list(size = 22)) %>%
  visLegend(addNodes = df_col_Legend, main = "Graph of co-authors",
    useGroups = FALSE) %>%

```

```
visEdges(font = list(size = 22)) %>%
visOptions(selectedBy = "equipe") %>%
visExport(type = "jpeg", name = "Full_Network_Co-authors_Example_managHAL",
          label = "Export as jpeg")
```

Ce script permet la création d'un réseau de co-auteur à partir de données bibliographique HAL.

1.5.7 exemple_reseau_unipartite_bipartite

```
# Création d'un réseau unipartite
set.seed(42) # Pour la reproductibilité
unipartite <- erdos.renyi.game(10, 0.3)
# unipartite_graph <- plot(unipartite, main =
# 'Réseau Unipartite', width = 800, height = 600)
# print(unipartite_graph)
matrice_adjacence_unipartite <- as.matrix(unipartite,
      "adjacency")

# Création d'un réseau bipartite Noms des nœuds
nodes_A <- c("A1", "A2", "A3")
nodes_B <- c("B1", "B2", "B3", "B4")
# Liste des arêtes
edges <- c("A1", "B1", "A1", "B2", "A2", "B1", "A2",
      "B2", "A2", "B3", "A3", "B2", "A3", "B4")
# Création du graphe
bipartite <- graph(edges, directed = FALSE)
V(bipartite)$type <- bipartite_mapping(bipartite)$type
plot(bipartite, layout = layout_as_bipartite, vertex.color = c("skyblue",
      "salmon")[V(bipartite)$type + 1], vertex.label.color = "black",
      vertex.shape = "circle", vertex.size = 30, edge.width = 2,
      main = "Réseau Bipartite")
matrice_adjacence_bipartite <- as.matrix(bipartite,
      "adjacency")
```

Ce script permet de créer les réseaux unipartites et bipartites présent dans les méthodes.

1.5.8 Script_R_SBM

```
# Create an empty adjacency matrix
matrice_adjacence <- matrix(0, nrow = length(node_list$id),
      ncol = length(node_list$id))
colnames(matrice_adjacence) <- node_list$id
rownames(matrice_adjacence) <- node_list$id

# fill in the adjacency matrix
```



```

for (k in 1:nrow(edge_list)) {
  i <- match(edge_list$from[k], node_list$id)
  j <- match(edge_list$to[k], node_list$id)
  matrice_adjacence[i, j] <- edge_list$width[k]
}
# deleting the legend lines
matrice_adjacence <- matrice_adjacence[1:(nrow(matrice_adjacence) -
  5), ]
matrice_adjacence <- matrice_adjacence[, 1:(ncol(matrice_adjacence) -
  5)]

plotMyMatrix(matrice_adjacence, dimLabels = list(row = "id",
  col = "id"))
# Estimation of a simple SBM using the adjacency
# matrix
my_model <- estimateSimpleSBM(matrice_adjacence, model = "poisson",
  dimLabels = c("id"))
# ne fonctionne pas sur quarto (l'accès au ancien
# plot)

# creation graph avec attribution des groupes
load("C:/users/marti/OneDrive/Bureau/Stage/donnees_sbm.Rdata")
group_membership <- apply(my_model$indMemberships,
  1, which.max)
group_colors <- rainbow(max(group_membership))
df_col_group = data.frame(group = 1:17, group_colors)
node_list <- node_list[1:(nrow(node_list) - 5), ]
df_col = data.frame(id <- node_list$id, group_membership <- group_membership,
  color = df_col_group$group_colors[apply(my_model$indMemberships,
  1, which.max)])
node_list$group = df_col$group_membership....apply.my_model.indMemberships...1..which.max.
node_list$color = df_col$color

# Network visualization
visNetwork(node_list, edge_list, width = "100%") %>%
  visIgraphLayout() %>%
  visGroups(groupname = "Number of publications",
    color = "lightblue", font = list(size = 22)) %>%
  visLegend() %>%
  visEdges(font = list(size = 22)) %>%
  visOptions(selectedBy = "color")

```

Ce script permet de réaliser un modèle en bloc stochastiques depuis le réseau créé dans l'annexe [Script_R_reseau_co-auteur](#).

1.5.9 Script_R_Sous-groupe

```
## test pour obtenir un sous groupe
find_info_author_associated_with_id_struc <- function(data_frame_info_author_struc) {
  data_info_author_associated_with_id_struc <- data_frame_info_author_struc[data_frame_info_aut
    1002311, ]
  return(data_info_author_associated_with_id_struc)
}
data_test_info_author_id_struc <- find_info_author_associated_with_id_struc(data_graph_author_str
data_graph_author_struc$Full_Name[data_graph_author_struc$id_struc ==
  1002311]
data_test_info_author_id_struc <- unique(data_test_info_author_id_struc)

#### -----
data_test_info_author_id_struc <- data_test_info_author_id_struc %>%
  separate(Full_Name, into = c("Nom", "Prenom"),
    extra = "merge", sep = " ", fill = "right")
data_test_info_author_id_struc <- data_test_info_author_id_struc %>%
  unite("Full_Name", Prenom, Nom, sep = " ")
temp_cleaned_data_author <- NULL
temp_cleaned_data_author$Full_Name <- managHAL:::clean_names(data_test_info_author_id_struc$Full_
# print inconsistent ids
inconsistent_ids <- find_inconsistent_ids(data_test_info_author_id_struc)
if (length(inconsistent_ids$Person_ID > 1)) {
  message(paste0("Il y a ", length(inconsistent_ids$Person_ID),
    " auteurs ayant leurs identifiant(s) HAL mal associé(s) à leurs noms :"))
  print(inconsistent_ids$Person_ID)
}
# print inconsistent ids and names
inconsistent_names_ids <- find_inconsistent_names_ids(data_test_info_author_id_struc)
if (length(inconsistent_names_ids$Full_Name > 1)) {
  message(paste0("Il y a ", length(inconsistent_names_ids$Full_Name),
    " Auteur ayant plusieurs formes auteurs ayant leurs identifiant(s) HAL mal associé(s) :"))
  print(inconsistent_names_ids$Full_Name)
}
# Replace Person_ID values in data_authors with
# corresponding values from inconsistent_ids
data_test_info_author_id_struc$Person_ID <- ifelse(data_test_info_author_id_struc$Full_Name %in%
  inconsistent_ids$Full_Name, inconsistent_ids$Person_ID[match(data_test_info_author_id_struc$F
  inconsistent_ids$Full_Name)], data_test_info_author_id_struc$Person_ID)
data_test_info_author_id_struc <- unique(data_test_info_author_id_struc)
# Update the temporary data.frame
temp_cleaned_data_author$Person_ID <- data_test_info_author_id_struc$Person_ID
### 2nd version : via chatGPT
for (i in seq_len(nrow(inconsistent_names_ids))) {
  idx <- which(temp_cleaned_data_author$Full_Name ==
    inconsistent_names_ids$Full_Name[i])

  if (length(idx) == i) {
```

```

        temp_cleaned_data_author$Person_ID[idx] <- inconsistent_names_ids$Person_ID[i]
    }
}
# My version : Replace Person_ID values in the
# temporary dataframe for (i in
# seq_len(nrow(inconsistent_names_ids))) {
# temp_cleaned_data_author[temp_cleaned_data_author$Full_Name
# == inconsistent_names_ids$Full_Name[i],
# 'Person_ID'] <-
# inconsistent_names_ids$Person_ID[i] } Replace
# Person_ID values in the original dataframe
data_test_info_author_id_struc$Person_ID <- temp_cleaned_data_author$Person_ID

#### -----
#### Create a logical vector for matching
#### Full_Name
match_vector <- data_test_info_author_id_struc$Full_Name %in%
    node_list$Full_Name
# Print the match_vector to see which rows match
print(match_vector)
# Assign Person_ID where Full_Name matches
data_test_info_author_id_struc <- data_test_info_author_id_struc %>%
    mutate(Person_ID = ifelse(is.na(Person_ID) & Full_Name %in%
        node_list$Full_Name, node_list$id[match(Full_Name,
        node_list$Full_Name)], Person_ID))
# Print the updated dataframe
print(data_test_info_author_id_struc)
node_list <- node_list %>%
    filter(id %in% data_test_info_author_id_struc$Person_ID)
# Filter edge_list to keep only rows where both
# 'from' and 'to' are in
# data_test_info_author_id_struc$Person_ID
edge_list <- edge_list %>%
    filter(from %in% data_test_info_author_id_struc$Person_ID &
        to %in% data_test_info_author_id_struc$Person_ID)
# Network visualization
visNetwork(node_list, edge_list, width = "100%") %>%
    visIgraphLayout() %>%
    visGroups(groupname = "Number of publications",
        color = "lightblue", font = list(size = 22)) %>%
    visLegend(addNodes = df_col_Legend, main = "Graph of Co-authors from MIA",
        useGroups = FALSE) %>%
    visEdges(font = list(size = 22)) %>%
    visOptions(selectedBy = "equipe") %>%
    visExport(type = "jpeg", name = "Full_Network_Co-authors_Example_managHAL",
        label = "Export as jpeg")
## visSave(selfcontained = TRUE, file =
## 'Partial_Network_Co-authors_Example_managHAL.html',
## background = 'white' )

```

```

## -----
## sbm Create an empty adjacency matrix
matrice_adjacence <- matrix(0, nrow = length(node_list$id),
  ncol = length(node_list$id))
colnames(matrice_adjacence) <- node_list$id
rownames(matrice_adjacence) <- node_list$id
# fill in the adjacency matrix
for (k in 1:nrow(edge_list)) {
  i <- match(edge_list$from[k], node_list$id)
  j <- match(edge_list$to[k], node_list$id)
  matrice_adjacence[i, j] <- edge_list$width[k]
}
plotMyMatrix(matrice_adjacence, dimLabels = list(row = "id",
  col = "id"))
# Estimation of a simple SBM using the adjacency
# matrix
my_model <- estimateSimpleSBM(matrice_adjacence, model = "poisson",
  dimLabels = c("id"))
# ne fonctionne pas sur quarto (l'accès au ancien
# plot)
my_model$nbBlocks
group_membership <- apply(my_model$indMemberships,
  1, which.max)
group_colors <- rainbow(max(group_membership))
df_col_group = data.frame(group = 1:3, group_colors)
## node_list <- node_list[1:(nrow(node_list) -
## 5), ] la légende non présente car selection
## labo
df_col = data.frame(id <- node_list$id, group_membership <- group_membership,
  color = df_col_group$group_colors[apply(my_model$indMemberships,
  1, which.max)])
node_list$group = df_col$group_membership...apply.my_model.indMemberships...1..which.max.
node_list$color = df_col$color
# Network visualization
visNetwork(node_list, edge_list, width = "100%") %>%
  visIgraphLayout() %>%
  visGroups(groupname = "Number of publications",
    color = "lightblue", font = list(size = 22)) %>%
  visLegend() %>%
  visEdges(font = list(size = 22)) %>%
  visOptions(selectedBy = "color")
plot(my_model, type = "expected")

```

Ce script permet de créer un sous-groupe du réseau, de construire un modèle en bloc stochastique à partir de ce nouveau réseau et de visualiser le tout.

BIBLIOGRAPHIE

- “About HAL.” n.d. <https://about.hal.science/>.
- AgroParisTech. 2024. “Science Ouverte.” <https://www.agroparistech.fr/recherche/science-ouverte>.
- Allaire, JJ, and Christophe Dervieux. 2024. “Quarto: R Interface to ‘Quarto’ Markdown Publishing System.” <https://CRAN.R-project.org/package=quarto>.
- Aubert, J., P. Barbillon, S. Donnet, and V. Miele. 2022. *Using Latent Block Models to Detect Structure in Ecological Networks*. Wiley. <https://doi.org/10.1002/9781119902799.ch6>.
- Bache, Stefan Milton, and Hadley Wickham. 2022. “Magrittr: A Forward-Pipe Operator for r.” <https://CRAN.R-project.org/package=magrittr>.
- Bengtsson, Henrik. 2021. “A Unifying Framework for Parallel and Distributed Processing in r Using Futures” 13. <https://doi.org/10.32614/RJ-2021-048>.
- Bouchet-Valat, Milan. 2023. “SnowballC: Snowball Stemmers Based on the c ‘Libstemmer’ UTF-8 Library.” <https://CRAN.R-project.org/package=SnowballC>.
- Chiquet, Julien, Sophie Donnet, and Pierre Barbillon. 2020. “Sbm: Stochastic Blockmodels.” The R Foundation. <https://doi.org/10.32614/cran.package.sbm>.
- . 2024. “Sbm: Stochastic Blockmodels.” <https://CRAN.R-project.org/package=sbm>.
- “Documentation API-HAL.” n.d. <https://api.archives-ouvertes.fr/docs/search/?#q>.
- Feinerer, Ingo, and Kurt Hornik. 2024. “Tm: Text Mining Package.” <https://CRAN.R-project.org/package=tm>.
- Fellows, Ian. 2018. “Wordcloud: Word Clouds.” <https://CRAN.R-project.org/package=wordcloud>.
- Izrailev, Sergei. 2024. “Tictoc: Functions for Timing r Scripts, as Well as Implementations of “Stack” and “StackList” Structures.” <https://CRAN.R-project.org/package=tictoc>.
- Moral-Muñoz, José A., Enrique Herrera-Viedma, Antonio Santisteban-Espejo, and Manuel J. Cobo. 2020. “Software Tools for Conducting Bibliometric Analysis in Science: An up-to-Date Review.” *El Profesional de La Información*. <https://doi.org/10.3145/epi.2020.ene.03>.
- Ooms, Jeroen. 2023. “Askpass: Password Entry Utilities for r, Git, and SSH.” <https://CRAN.R-project.org/package=askpass>.
- Premraj, Rahul. 2021. “mailR: A Utility to Send Emails from r.” <https://CRAN.R-project.org/package=mailR>.
- Vaughan, Davis, and Matt Dancho. 2022. “Furrr: Apply Mapping Functions in Parallel Using Futures.” <https://CRAN.R-project.org/package=furrr>.
- Wickham, Hadley. 2016. “Ggplot2: Elegant Graphics for Data Analysis.” <https://ggplot2.tidyverse.org>.
- . 2023. “Stringr: Simple, Consistent Wrappers for Common String Operations.” <https://CRAN.R-project.org/package=stringr>.
- . 2024. “Rvest: Easily Harvest (Scrape) Web Pages.” <https://CRAN.R-project.org/package=rvest>.
- Wickham, Hadley, and Jennifer Bryan. 2023. “Readxl: Read Excel Files.” <https://CRAN.R-project.org/package=readxl>.
- Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. “Dplyr: A Grammar of Data Manipulation.” <https://CRAN.R-project.org/package=dplyr>.
- Wickham, Hadley, and Lionel Henry. 2023. “Purrr: Functional Programming Tools.” <https://CRAN.R-project.org/package=purrr>.