Projet Networks

ATALLAH Joanne & COUSTILLAC Célestine

Introduction

Ce projet a pour but d'étudier l'évolution d'un réseau de chercheurs après un changement important, appelé « traitement ». Le groupe étudié est composé de 215 individus travaillant sur des sujets proches et partageant un même lieu de travail. L'objectif est de **comprendre comment ce changement a modifié les interactions scientifiques au sein du groupe, et d'identifier ses effets** sur la structure globale du réseau, ainsi que sur des sous-groupes définis par certaines caractéristiques individuelles, notamment le genre et le statut de « core » (chercheur identifié comme central par son expertise).

Les données utilisées pour cette étude sont organisées en deux fichiers :

- Une table des **nœuds**, indiquant pour chaque chercheur son genre, son statut « core », et son éventuel rôle de coordinateur.
- Une table des **liens**, précisant l'existence de collaborations avant et après traitement.

À partir de ces données, deux réseaux ont été construits : un premier représentant les collaborations avant traitement, et un second après traitement. Chaque nœud correspond à un chercheur, et chaque lien indique une collaboration scientifique active entre deux individus. Le réseau est modélisé sous forme **non orientée**, ce qui signifie que les collaborations sont considérées comme réciproques : si un chercheur A collabore avec un chercheur B, alors l'inverse est également vrai. Un nettoyage des doublons a donc été effectué dans les données de liens, puis les attributs individuels ont été associés aux nœuds.

L'ensemble de l'analyse a été réalisé en Python, principalement à l'aide des bibliothèques NetworkX, Pandas et Matplotlib. (Code annexe A)

L'analyse s'articule autour de plusieurs questions clés, présentées dans l'ordre de leur traitement dans le rapport, selon une logique avant/après traitement :

(1) Comment la structure globale du réseau évolue-t-elle après traitement ?
(2) La croissance du réseau suit-elle un mécanisme d'attachement préférentiel, comme dans les réseaux scale-free ?
(3) En quoi cette évolution dépend-elle des caractéristiques individuelles, notamment le genre et le statut « core » ?
(4) Les femmes et les chercheurs non-core sont-ils mieux intégrés dans le réseau après transformation ?
(5) Observe-t-on des logiques d'homophilie ou de formation de sous-groupes ?
(6) L'appartenance au groupe « core » modifie-t-elle ces dynamiques relationnelles ?
(7) Peut-on identifier des communautés structurées au sein du réseau, et comment évoluent-elles après traitement ?

En répondant à ces questions, ce travail propose une exploration approfondie de l'impact du traitement sur les structures collaboratives, les dynamiques d'inclusion, et les logiques relationnelles dans un réseau scientifique réel.

I - Analyse générale du réseau

I.I - Évolution des indicateurs structurels globaux

Avant d'étudier les effets du traitement, il est essentiel de décrire la structure du réseau de chercheurs dans les deux périodes observées (avant et après traitement). Cette analyse descriptive repose sur plusieurs indicateurs classiques en théorie des réseaux. Nous présentons ainsi, pour chaque période, les principales mesures structurelles du réseau (Code annexe B) :

	Réseaux avant traitement	Réseaux avant traitement
Nombre de chercheurs	215	215
Nombre de collaborations	363	581
Nombre de chercheurs isolés	74	26
Nombre de composantes connexes	85	41
Taille de la composante géante	111	138
Pourcentage dans la composante géante	51.628 %	64.186 %
Longueur moyenne des plus courts chemins dans la composante géante	3.708	3.720
Densité du réseau	0.016	0.025
Degré moyen des chercheurs	3.377	5.405
Clustering moyen	0.268	0.422
Clustering global	0.484	0.559

L'analyse du réseau de collaborations entre chercheurs, avant et après traitement, met en évidence une transformation claire vers un réseau plus connecté, plus cohérent et mieux structuré. Bien que le **nombre total de chercheurs reste identique (215)**, ce qui assure une base de comparaison stable, plusieurs indicateurs montrent une nette amélioration de la connectivité.

Tout d'abord, le **nombre de collaborations augmente fortement**, passant de 363 à 581, ce qui suggère que les chercheurs ont globalement multiplié leurs interactions. Cette évolution s'accompagne d'une **forte diminution du nombre de chercheurs isolés**, qui passe de 74 à 26, preuve que de nombreux individus auparavant exclus des collaborations ont été intégrés dans des relations de travail. En parallèle, le **nombre de composantes connexes**, c'est-à-dire de groupes de chercheurs déconnectés les uns des autres, **diminue de moitié**, passant de 85 à 41. Cela traduit une meilleure interconnexion entre les différents segments du réseau. Ce renforcement global se reflète aussi dans la taille de la composante géante, qui passe de 111 à 138 chercheurs. En proportion, cela représente une progression de 51,6 % à 64,2 % de l'ensemble du réseau. Autrement dit, près des deux tiers des chercheurs font désormais partie d'un même ensemble connecté, ce qui marque une amélioration importante de la cohésion du réseau. C'est à l'intérieur de cette composante principale que se calcule la **longueur moyenne des plus courts chemins**, un indicateur essentiel pour mesurer l'accessibilité entre les individus. Cette mesure **reste stable** entre les deux versions du réseau, passant très légèrement de 3,708 à 3,720. Cela montre que, même si de nombreux liens ont été ajoutés, la distance moyenne entre deux chercheurs connectés n'a presque pas changé. Ce résultat est cohérent avec l'idée que le réseau était déjà bien relié en son centre, et que le traitement a surtout permis d'y intégrer des chercheurs auparavant isolés ou plus éloignés, sans modifier les distances entre les nœuds déjà proches. (Code annexe C et D)

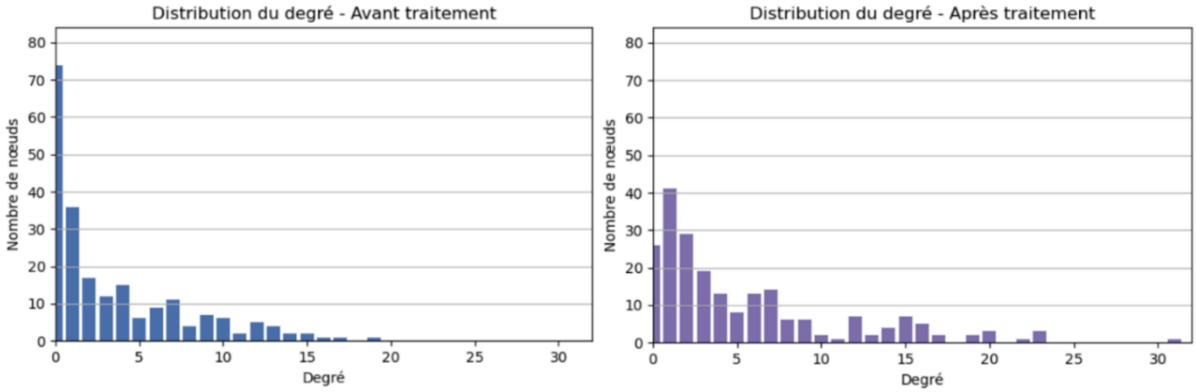
De manière cohérente avec cette dynamique d'intégration, le **degré moyen**, c'est-à-dire le nombre moyen de collaborations par chercheur, **passe de 3,4 à 5,4**. Ainsi, non seulement les chercheurs sont mieux connectés globalement, mais ils interagissent aussi individuellement avec un plus grand nombre de collègues.

Ce gain en connectivité se traduit aussi par une densification du réseau : la **densité augmente** de 0,016 à 0,025, ce qui indique que davantage de liens sont effectivement créés parmi les connexions possibles. Cette dynamique de renforcement est également visible à un niveau plus local. Le **clustering moyen passe de 0,268 à 0,422**, ce qui montre qu'en moyenne, les chercheurs sont plus souvent insérés dans des petits groupes où leurs collègues collaborent aussi entre eux. Toutefois, cette mesure accorde le même poids à tous les nœuds, même les moins connectés, ce qui peut fausser la perception de la structure réelle du réseau. C'est pourquoi on complète cette analyse avec le clustering global, qui prend en compte l'ensemble des triangles fermés par rapport à tous les triplets possibles. Cette mesure met davantage en valeur les zones denses du réseau, en particulier autour des nœuds les plus connectés. Dans notre cas, le **clustering global augmente** également, de 0,484 à 0,559. Le fait que le clustering global reste nettement supérieur au clustering moyen, avant comme après traitement, montre que la densité du réseau reste principalement concentrée autour des chercheurs les plus connectés. Cela suggère que même si davantage d'individus participent à des structures locales plus cohésives, la **cohésion globale repose encore sur un noyau central d'individus fortement interconnectés**.

Dans l'ensemble, le réseau passe d'une structure fragmentée, avec de nombreux chercheurs isolés ou en petits groupes séparés, à un ensemble plus intégré, où les collaborations sont plus nombreuses, mieux réparties, et localement plus denses. Cette évolution montre que le réseau est devenu plus solide et plus favorable aux échanges entre les chercheurs.

I.II - Répartition des collaborations entre chercheurs

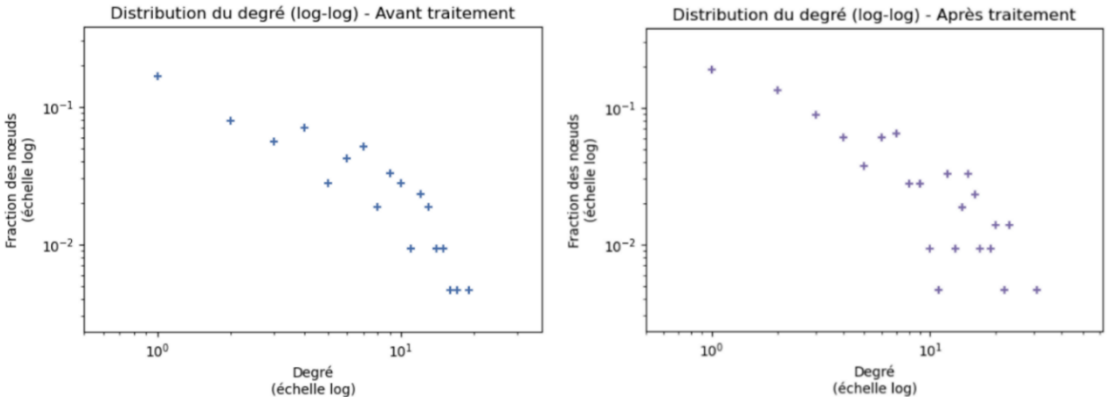
Afin de compléter ces premiers constats globaux, il est pertinent d'examiner plus en détail la manière dont les collaborations sont réparties entre les chercheurs. En effet, si le degré moyen augmente nettement après traitement, cela ne renseigne pas à lui seul sur la distribution réelle des liens dans le réseau. C'est pourquoi l'analyse des distributions du degré, avant et après transformation, apporte un éclairage complémentaire. (Code annexe E.1)



Les graphiques précédents illustrent ainsi plus concrètement cette évolution de la structure du réseau en montrant la distribution du degré des chercheurs avant et après traitement. On y observe des **différences nettes dans la répartition des collaborations individuelles** (i.e. le nombre de liens par chercheur). **Avant traitement**, la majorité des chercheurs a un degré très faible, voire nul : plus de 30 % des individus n'ont aucune connexion, ce qui indique une forte proportion de chercheurs totalement isolés. La courbe décroît rapidement, ce qui traduit une structure très inégalitaire, avec très peu de chercheurs ayant un nombre élevé de collaborations. **Après traitement**, la distribution est beaucoup plus étalée. D'une part, la proportion de chercheurs sans lien chute nettement, passant d'environ 34 % à environ 13 %, ce qui confirme une forte réduction de l'isolement dans le réseau. D'autre part, la présence de degrés plus élevés devient plus fréquente, avec un étalement vers la droite : les chercheurs sont non seulement plus nombreux à être connectés, mais certains d'entre eux établissent beaucoup plus de collaborations qu'auparavant. Cette transformation est cohérente avec les évolutions observées dans les indicateurs globaux (degré moyen, densité, taille de la composante géante, etc.), et illustre bien la montée en puissance des échanges au sein du collectif scientifique analysé.

I.III - Analyse Log-Log

On observe également que la forme de ces distributions évoque celle observée dans de nombreux réseaux de collaboration scientifique, souvent qualifiés de scale-free. Dans ce type de structure, la majorité des nœuds ont un faible degré, tandis qu'une minorité de hubs concentre une grande partie des liens. Cela semble être le cas ici, comme le suggèrent les histogrammes : avant traitement, plus de 70 chercheurs ont un degré inférieur à 3, contre très peu avec un degré supérieur à 10, après traitement, on observe une extension de cette queue de distribution jusqu'à des degrés supérieurs à 30. Pour tester rigoureusement l'hypothèse selon laquelle la distribution des degrés dans notre réseau suit une loi de puissance (*power law*), nous avons représenté cette distribution en échelle log-log. Ce type de graphique est particulièrement adapté pour identifier des comportements de type loi de puissance, car dans ce repère, ces lois apparaissent sous la forme de droites. L'exposant α correspond alors à la pente de la droite, et l'ordonnée à l'origine représente le coefficient multiplicatif.



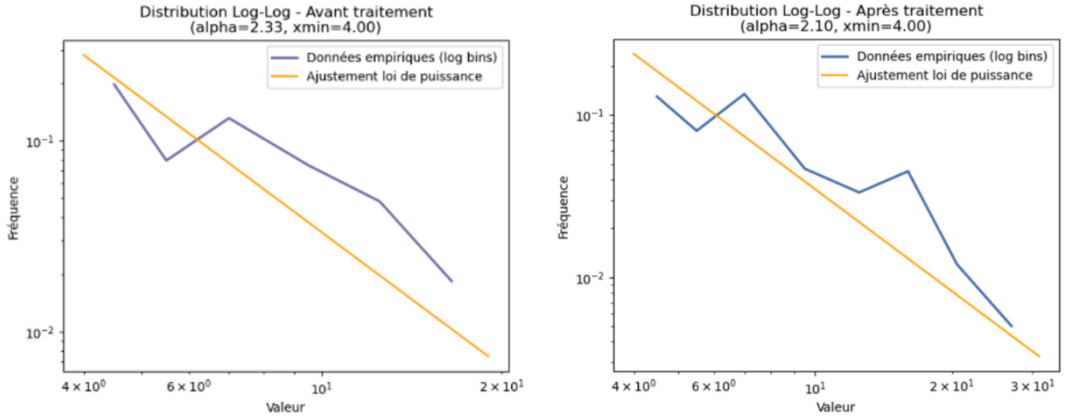
Et c'est exactement ce que l'on observe dans notre réseau, à partir d'un certain seuil minimal x_{min}, au-delà duquel la queue de la distribution semble bien suivre une loi de puissance.

Mathématiquement, une variable x suit une loi de puissance si elle est tirée d'une distribution de probabilité de la forme :

$$p(x) \propto x - \alpha$$

où α est un paramètre constant de la distribution, appelé **exposant** ou **paramètre d'échelle** (*scaling parameter*). Ce paramètre prend généralement une valeur comprise entre 2 et 3, bien qu'il existe quelques exceptions. En pratique, peu de phénomènes empiriques obéissent à une loi de puissance sur toute l'étendue des valeurs de x. Plus souvent, la loi de puissance ne s'applique qu'aux valeurs supérieures à un certain seuil x_{min}. Dans ce cas, on dit que **la queue de la distribution** suit une loi de puissance. (Article POWER-LAW DISTRIBUTIONS IN EMPIRICAL DATA, A. Clauset, C. R. Shalizi and M. E. J. Newman)

Afin de le confirmer, nous avons utilisé la bibliothèque `powerlaw`, qui permet d'appliquer un ensemble de techniques statistiques pour identifier la présence d'une loi de puissance et en estimer les paramètres. Concrètement, nous avons extrait les degrés de chaque nœud du graphe (avant et après traitement), puis ajusté une loi de puissance à ces données à l'aide de la méthode du maximum de vraisemblance. Nous avons fixé $x_{min} = 4$, ce qui signifie que nous concentrons l'ajustement uniquement sur la queue de la distribution, là où la loi de puissance est supposée s'appliquer. Les paramètres estimés sont ensuite utilisés pour tracer à la fois la distribution empirique (par regroupement log-bins) et l'ajustement théorique sur le même graphique.



Sur les deux figures obtenues, nous constatons que l'ajustement est relativement bon dans les deux cas. **Avant traitement**, l'exposant estimé est $\alpha = 2{,}33$, tandis qu'**après traitement**, il diminue à $\alpha = 2{,}10$ Cette diminution de l'exposant traduit un changement structurel du réseau : une plus grande hétérogénéité s'installe, avec une probabilité plus élevée d'observer des nœuds de très haut degré après traitement. En d'autres termes, la distribution devient plus lourde en queue, ce qui indique que certains nœuds gagnent en centralité de manière significative. Cela suggère une montée en influence de certains acteurs du réseau après l'événement perturbateur. (Code annexe E.2)

II.IV - Comparaison avec des modèles de réseaux théoriques

Remarque : les valeurs présentées correspondent à une génération aléatoire de différents réseaux à un instant donné (avec le même nombre de nœuds et la même densité que notre réseau). En relançant le code, de nouvelles valeurs peuvent apparaître en raison de la nature stochastique du processus, mais l'interprétation globale demeure inchangée.

Pour mieux comprendre la nature de notre réseaux on a décidé d'analyser G_apres comme représentatif de notre réseaux finale, il est pertinent de le comparer à plusieurs classes de réseaux théoriques bien connues : le graphe aléatoire d'Erdős-Rényi (ER), le modèle petit-monde de Watts-Strogatz (WS), le réseau scale-free de Barabási-Albert (BA), et le graphe régulier. Deux métriques essentielles sont ici mobilisées : la distance moyenne entre les nœuds (*average shortest path*) et le *clustering global*. Le réseau réel présente une distance moyenne de 3,72 et un clustering global de 0,56. Cette configuration se distingue nettement de celle des graphes aléatoires (distance plus faible à 3,4 mais clustering négligeable à 0,03) et des graphes réguliers (distance plus grande à 4,22 et clustering quasi nul à 0,01). Le graphe WS offre un bon niveau de clustering (0,37) mais souffre d'une distance moyenne bien plus élevée (6,45), tandis que le modèle BA reproduit bien la faible distance (2,91) mais échoue à capturer la structure locale, avec un clustering très faible (0,058). Ainsi, notre réseau cumule les propriétés essentielles d'un réseau de collaboration scientifique observé dans la réalité : une forte cohésion locale (clustering élevé), l'émergence de communautés denses, et une distance moyenne modérée reflétant une bonne accessibilité de l'information. Il ne correspond parfaitement à aucun modèle théorique classique, mais semble émerger comme un compromis entre un réseau *scale-free* (pour sa structure hub et sa distribution en loi de puissance) et un réseau petit-monde (pour sa densité de triangles et sa structure communautaire). En d'autres termes, notre réseau reflète la nature hybride typique des réseaux sociaux réels de chercheurs, combinant hétérogénéité du degré, forte localité des liens et connectivité globale efficace, ce qui en fait un graphe structurellement réaliste et empiriquement cohérent avec les dynamiques de collaboration scientifique. (Code annexe G)

I.V - Mécanisme d'attachement préférentiel

Le modèle de réseau aléatoire, tel que proposé par Erdős et Rényi, repose sur une probabilité uniforme d'apparition de liens entre les nœuds. Or, ce cadre théorique s'avère inadapté à la description de notre réseau empirique. En effet, il échoue à reproduire deux propriétés fondamentales observées dans notre structure réelle : (1) un clustering global élevé, indicateur de cohésion locale, et (2) une distribution des degrés qui ne suit pas une loi de Poisson.

Dans le but de mieux modéliser l'évolution de la structure après traitement, nous avons formulé l'hypothèse que le mécanisme d'**attachement préférentiel** pourrait être à l'origine de l'apparition de nouveaux liens. Ce principe, théorisé par Barabási et Albert (1999), stipule que les nouveaux nœuds ont tendance à se connecter préférentiellement aux nœuds déjà bien connectés, selon une probabilité proportionnelle à leur degré k. Ce comportement, souvent résumé par l'expression « rich-get-richer », est à l'origine de la formation des réseaux **scale free**, dont la distribution des degrés suit une loi de puissance.

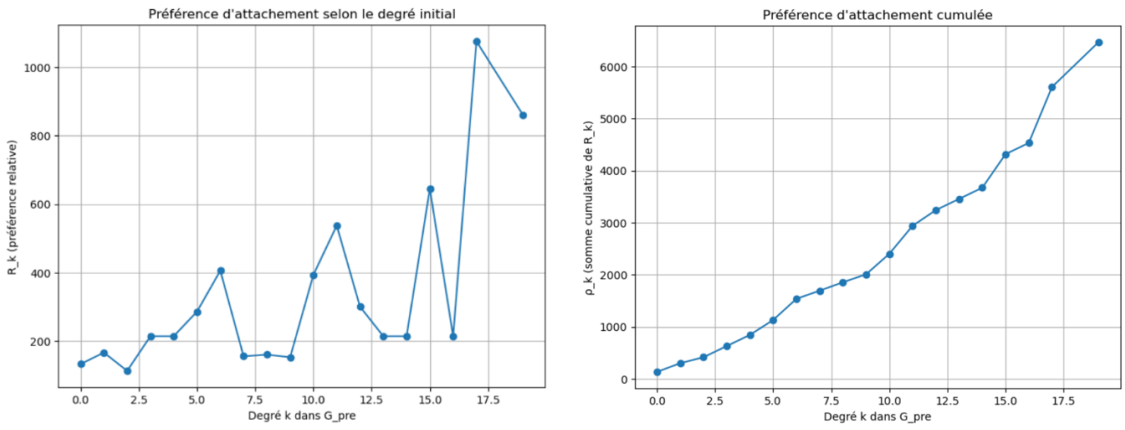
Pour explorer cette piste, nous avons d'abord identifié, dans le graphe initial G_{avant}, les nœuds totalement isolés (degré nul). Parmi eux, nous avons isolé ceux qui, dans $G_{\text{après}}$, ont établi au moins une connexion. Ces **nœuds activés** sont interprétés comme des **nouveaux entrants fonctionnels**, car bien qu'ils soient présents dans la structure initiale, ils n'y exercent aucune interaction. Leur comportement de connexion reflète donc une dynamique d'intégration comparable à celle d'un véritable nouveau nœud.

Pour chacun de ces nœuds activés, nous avons extrait la liste des nœuds cibles avec lesquels ils ont formé des liens, en nous assurant de ne pas compter plusieurs fois les mêmes paires (filtrage des doublons via tri). Pour chaque degré cible k, nous avons comptabilisé le nombre de connexions établies, que nous avons noté P_k. Afin de corriger cette distribution brute pour les tailles de classes de degré, nous avons introduit la **préférence relative** R_k, définie par : $R_k = P_k \cdot \frac{N}{n_k}$

où P_k est le nombre de liens vers des nœuds de degré k, n_k est le nombre total de nœuds de degré k $dans$ G_{avant} et N le nombre total de nœuds dans G_{avant}.

Cette quantité R_k permet de détecter si les nœuds de degré k attirent **plus de liens que prévu** par une distribution uniforme. Si l'attachement était purement aléatoire, R_k serait constant pour tout k. Une croissance de R_k avec k est donc un signal clair de préférence proportionnelle au degré. Les résultats sont sans ambiguïté : R_k **croît avec** $\boldsymbol{k}$, atteignant des valeurs exceptionnelles pour les degrés les plus élevés (ex. $R_{17} > 1000$). Pour mieux visualiser cette dynamique, nous avons également calculé une version **cumulée** de cette préférence, notée ρ_k , définie comme : $\rho_k = \int_1^k R_{k'}\, dk'$

La fonction ρ_k , plus lisse, confirme la tendance croissante : les nœuds de haut degré concentrent une part disproportionnée des connexions des nouveaux activés. Cette observation constitue une **preuve empirique forte** de l'existence d'un **mécanisme d'attachement préférentiel** dans notre réseau. Notre démarche, de la définition de R_k à l'analyse de ρ_k, constitue ainsi une **modélisation indirecte mais robuste** du phénomène d'attachement préférentiel. Les résultats obtenus viennent conforter l'idée que **l'évolution du graphe $G_{\text{après}}$ n'est pas aléatoire**, mais bien régie par une dynamique endogène, où les **nœuds les plus centraux continuent à se renforcer**. Ce comportement est emblématique des **réseaux scale free**, et valide la pertinence du **modèle de Barabási-Albert** comme base de modélisation de la croissance observée dans notre système. (Code annexe F)



I.IV - Analyse des nœuds les plus influents

Dans la continuité de notre analyse du réseau, nous avons cherché à **savoir s'il existait**, comme dans le célèbre cas de Paul Erdos, un **chercheur particulièrement central dans notre réseau**. L'objectif était de voir si un individu jouait un rôle de point de passage important dans les collaborations, en étant à la fois très connecté (avec un grand nombre de liens) et bien placé pour rejoindre rapidement les autres membres du réseau. Pour cela, nous avons croisé trois indicateurs classiques de centralité : le degré (nombre de collaborateurs directs), la closeness (proximité moyenne avec les autres) et la betweenness (capacité à relier différents nœuds en étant sur les plus courts chemins).

	Degré		Closeness		Betweenness	
	Avant	Après	Avant	Après	Avant	Après
1	Chercheur 7463 19	Chercheur 43551 31	Chercheur 43544 0.2064	Chercheur 9925 0.2603	Chercheur 43544 0.056403	Chercheur 43551 0.071856
2	Chercheur 9938 17	Chercheur 7463 23	Chercheur 9938 0.2027	Chercheur 43544 0.2603	Chercheur 7463 0.044771	Chercheur 80891 0.055260
3	Chercheur 36929 16	Chercheur 9925 23	Chercheur 7463 0.1950	Chercheur 43551 0.2580	Chercheur 43552 0.042635	Chercheur 7463 0.049378
4	Chercheur 9881 15	Chercheur 43544 23	Chercheur 43552 0.1923	Chercheur 9928 0.2492	Chercheur 9938 0.041066	Chercheur 9895 0.047636
5	Chercheur 43544 15	Chercheur 80891 22	Chercheur 7475 0.1891	Chercheur 9938 0.2485	Chercheur 9877 0.032536	Chercheur 36924 0.047192
6	Chercheur 7475 14	Chercheur 9882 20	Chercheur 72071 0.1866	Chercheur 77958 0.2403	Chercheur 43119 0.032194	Chercheur 9928 0.044930
7	Chercheur 80891 14	Chercheur 992820	Chercheur 9945 0.1860	Chercheur 43547 0.2377	Chercheur 9899 0.028650	Chercheur 9925 0.044893
8	Chercheur 9886 13	Chercheur 9938 20	Chercheur 9881 0.1854	Chercheur 85232 0.2377	Chercheur 9881 0.025010	Chercheur 43544 0.043933
9	Chercheur 36851 13	Chercheur 43539 19	Chercheur 9886 0.1836	Chercheur 91627 0.2364	Chercheur 72071 0.022555	Chercheur 43547 0.042144
10	Chercheur 69859 13	Chercheur 85232 19	Chercheur 43551 0.1836	Chercheur 7463 0.2351	Chercheur 7475 0.022047	Chercheur 43538 0.035338

Avant traitement, plusieurs chercheurs se distinguent, notamment 9938, 43544 ou 7463, qui apparaissent régulièrement dans les classements par degré, closeness et betweenness. Toutefois, aucun ne domine clairement toutes les mesures. Après traitement, c'est le chercheur 43551 qui arrive en tête du classement par degré (31 liens) et par betweenness (0,0719), tandis que 43544, 7463 ou encore 9925 occupent des places élevées selon les différents critères.

Ces résultats montrent que les chercheurs les plus centraux ne sont pas toujours les mêmes selon l'indicateur considéré. Autrement dit, un individu peut être très connecté sans forcément être le plus proche des autres, ou jouer un rôle de liaison sans avoir le plus de liens. Cela confirme que **le réseau ne tourne pas autour d'une seule figure centrale, mais repose plutôt sur plusieurs pôles d'influence complémentaires**. Cette configuration illustre un réseau distribué, dans lequel la centralité est partagée entre différents profils, chacun jouant un rôle spécifique dans l'organisation des collaborations. (Code annexe I)

II – Analyse selon les attributs des chercheurs

Après avoir examiné la structure générale du réseau et son évolution globale, il est important d'approfondir l'analyse en tenant compte de certaines caractéristiques individuelles des chercheurs. En particulier, deux dimensions sont explorées ici : le **sexe** des individus, et leur **statut core ou non-core**, qui reflète leur niveau de reconnaissance ou d'expertise dans le réseau.

L'objectif est de voir si ces attributs influencent la manière dont les chercheurs sont connectés, leur position dans le réseau, ou leur intégration après traitement.

II.I - Analyse selon le sexe des chercheurs

On commence par étudier les différences de connectivité entre **femmes et hommes** dans le réseau. Il est important de noter que la composition du réseau reste identique avant et après traitement, avec 49 femmes et 166 hommes, soit environ **23 % de femmes et 77 % d'hommes**. On commence par observer quelques statistiques générales par sexe, comme le nombre moyen de collaborations, le taux d'isolement ou la position dans le réseau. Dans un second temps, on regarde s'il existe une homophilie, c'est-à-dire si les chercheurs ont tendance à collaborer davantage avec des personnes du même sexe.

II.I.1 - Statistiques globales par sexe

Le tableau ci-dessous résume plusieurs mesures clés du réseau, calculées séparément pour les femmes et les hommes, avant et après traitement. Il permet de comparer directement leur niveau d'intégration dans le réseau, ainsi que l'évolution de leur position.

	Hommes		Femmes	
	Avant traitement	Après traitement	Avant traitement	Après traitement
Nombre moyen de collaborations	3.500	5.120	2.959	6.367
Proportion de chercheurs isolés par genre	0.325	0.139	0.408	0.061
Coefficient moyen de clustering	0.286	0.420	0.206	0.428
Coefficient moyen de betweenness	0.003	0.005	0.003	0.005
Coefficient moyen de closeness	0.077	0.115	0.069	0.132

L'analyse de ces indicateurs met en évidence plusieurs écarts, mais aussi des évolutions notables qui montrent que le traitement du réseau a particulièrement bénéficié aux femmes.

Au départ, les hommes ont en moyenne plus de collaborations que les femmes (3,5 contre 3), ce qui montre que les femmes sont un peu moins connectées dans le réseau. Mais après traitement, la situation s'inverse, les femmes ont en moyenne 6,4 collaborations, contre 5,1 pour les hommes. Cela veut dire que **les femmes ont plus que doublé leur nombre de liens (+115 %), alors que les hommes n'en ont gagné qu'environ 46 %**. Le traitement du réseau a donc permis aux femmes de rattraper leur retard et même de passer devant. Cette évolution se retrouve aussi dans la proportion de chercheurs isolés. Avant traitement, 41 % des femmes n'étaient connectées à personne, contre 33 % des hommes. Après traitement, cette part diminue fortement pour les deux genres, mais plus nettement chez les femmes, qui ne sont plus que 6 % à être isolées, contre 14 % chez les hommes. Cela montre que **les femmes ont été mieux réintégrées dans le réseau, et que le traitement a réduit un déséquilibre initial important**. On observe également des différences dans les mesures de centralité, qui indiquent la place qu'occupent les individus dans le réseau. Avant traitement, les femmes ont un clustering moyen plus faible (0,206 contre 0,286), ce qui signifie qu'elles sont moins souvent insérées dans des petits groupes de collaborateurs qui sont eux-mêmes en lien. Leur closeness est également plus basse (0,069 contre 0,077), ce qui reflète une plus grande distance moyenne par rapport aux autres chercheurs. Mais après traitement, tous les indicateurs augmentent pour les deux genres, et les femmes dépassent les hommes en clustering (0,428 contre 0,420) et en closeness (0,132 contre 0,115). Cela signifie qu'elles sont **désormais plus proches du reste du réseau et mieux intégrées dans des groupes soudés**. En ce qui concerne la betweenness, les deux genres sont quasiment à égalité avant et après traitement, ce qui indique un rôle similaire en termes de liaison entre différentes parties du réseau. (Code annexe H)

En résumé, ces données montrent que les femmes étaient initialement moins intégrées, mais que le traitement a permis une progression plus forte de leur connectivité et de leur position dans le réseau, jusqu'à dépasser les hommes sur plusieurs indicateurs clés. Le **réseau apparaît donc plus équilibré après transformation, avec une meilleure inclusion des femmes dans les dynamiques collaboratives**.

II.I.2 - Homophilie de genre

Au-delà de ces différences de position dans le réseau, il est aussi intéressant de **se demander si les chercheurs ont tendance à collaborer avec des personnes du même sexe.** Cette question renvoie à la notion d'homophilie, que l'on peut mesurer à travers le coefficient d'assortativité par genre. Ce coefficient varie entre –1 et +1. Une valeur proche de +1 indique une forte préférence pour les liens avec des personnes du même genre, une valeur proche de –1 signifie une préférence pour les personnes de genre différent, et une valeur proche de 0 traduit l'absence de préférence particulière.

Dans notre cas, **avant traitement**, l'assortativité par genre est légèrement positive (environ 0,043), ce qui indique une faible tendance à l'homophilie : les chercheurs collaboraient un peu plus souvent avec des personnes du même sexe que ce qui serait attendu par hasard, mais la préférence restait très modérée. **Après traitement**, le coefficient devient légèrement négatif (environs -0,025), ce qui signifie que cette légère préférence a disparu. On observe même une petite tendance inverse : les chercheurs collaborent un peu plus souvent avec des personnes de genre différent, bien que cela reste très faible.

En résumé, le traitement du réseau semble avoir réduit la séparation par genre dans les collaborations. Il a non seulement permis une meilleure intégration des femmes, mais aussi **favorisé des interactions plus mixtes, ce qui va dans le sens d'un réseau plus équilibré et plus inclusif**. (Code annexe J.1)

II.II - Analyse selon le statut « core » des chercheurs

Après avoir étudié les différences liées au genre, on s'intéresse maintenant à une autre distinction importante dans le réseau: celle entre les chercheurs « core » et « non-core ». Cette catégorisation reflète un niveau d'expertise ou de reconnaissance dans le domaine. Dans notre cas, **le réseau comprend 25 chercheurs core soit environ 12 % de la population total et les non-core 190 soit 88 %**, ce qui montre que les profils les plus reconnus sont nettement minoritaires au sein du réseau.

II.II.1 - Statistiques globales par statut « core »

Pour cela, on commence par comparer un ensemble d'indicateurs globaux entre les chercheurs core et non-core, avant et après traitement. Le tableau ci-dessous permet de visualiser directement les écarts entre les deux groupes, que ce soit en termes de nombre de collaborations, de taux d'isolement ou de position dans le réseau.

	Chercheur « non-core »		Chercheur « core »	
	Avant traitement	Après traitement	Avant traitement	Après traitement
Nombre moyen de collaborations	3.205	5.411	4.680	5.360
Proportion de chercheurs isolés par genre	0.363	0.200	0.126	0.080
Coefficient moyen de clustering	0.266	0.430	0.285	0.357
Coefficient moyen de betweenness	0.003	0.005	0.006	0.006
Coefficient moyen de closeness	0.071	0.118	0.113	0.127

En ce qui concerne le nombre moyen de collaborations, les chercheurs core sont plus connectés que les non-core avant traitement, 4,68 contre 3,21. Après traitement, la situation s'équilibre : les non-core atteignent 5,41 collaborations en moyenne, soit une augmentation de 68 %, tandis que les core montent à 5,36 soit seulement 14 % d'augmentation. Les **non-core dépassent donc légèrement les core en termes de collaborations, ce qui montre une forte amélioration de leur intégration**. Pour le taux d'isolement, on constate également une réduction plus marquée chez les non-core. Avant traitement, 36,3 % d'entre eux n'étaient connectés à personne, contre seulement 12,6 % pour les core. Après traitement, l'isolement chute à 20 % chez les non-core, et à 8 % chez les core. Bien que **les core restent globalement moins isolés, la baisse est plus importante chez les non-core**, ce qui indique un effort d'inclusion plus fort pour ce groupe. Le clustering moyen, qui mesure l'insertion dans des groupes de collègues interconnectés, est légèrement plus élevé chez les core au départ, 0,285 contre 0,266. Après traitement, c'est l'inverse : les non-core atteignent 0,430, contre 0,357 pour les core. Les **non-core enregistrent donc une hausse de leur clustering moyen de +61,6%, contre +25,3 % pour les core**, ce qui montre qu'ils ont été **nettement plus intégrés dans des groupes soudés**. Concernant la betweenness, les core ont une valeur deux fois plus élevée que les non-core avant traitement (0,006 contre 0,003), ce qui traduit un rôle plus important dans la circulation des informations. Après traitement, les deux groupes progressent légèrement, mais l'écart reste à peu près le même. Cela montre que **même si les non-core gagnent en importance, les core conservent un rôle de passage plus marqué**. Enfin, pour la closeness, qui mesure la proximité moyenne avec tous les autres chercheurs, les core sont clairement plus centraux au départ, 0,113 contre 0,071. Après traitement, l'écart se réduit : les non-core passent à 0,118, tandis que les core montent à 0,127. Là encore, les non-core progressent plus fortement avec une progression de 66 % contre 13 %, ce qui indique qu'ils deviennent plus accessibles et mieux intégrés dans la structure globale du réseau.

En résumé, **les chercheurs core conservent une position favorable**, **mais les non-core progressent plus fortement** sur presque tous les indicateurs. Cette dynamique montre que le traitement du réseau a permis une meilleure inclusion des profils initialement moins connectés, et a contribué à réduire les écarts structurels tout en renforçant la cohésion globale du réseau. (Code annexe L)

II.II.2 - Homophilie de statut « core »

On peut également se demander si les chercheurs ont tendance à collaborer avec d'autres personnes ayant le même statut, c'est-à-dire core avec core et non-core avec non-core. Le coefficient d'assortativité par core permet de répondre à cette question.

Avant traitement, la valeur est modérément positive (environ 0,154), ce qui indique qu'il existe une légère préférence pour les collaborations entre chercheurs de même statut, en particulier entre core. Après traitement, cette tendance s'accentue légèrement (environ 0,173), ce qui montre que le renforcement du réseau a aussi consolidé certaines logiques de regroupement internes.

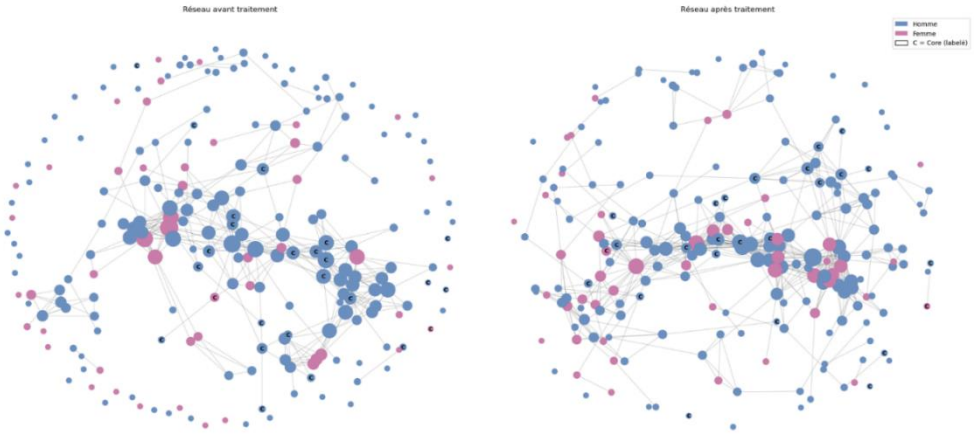
Autrement dit, bien que l'ensemble du réseau soit devenu plus connecté, **les collaborations ont aussi eu tendance à se structurer davantage à l'intérieur des groupes core et non-core**. Cela peut s'expliquer par des affinités de travail, des intérêts communs ou simplement par la dynamique naturelle du renforcement des liens existants. (Code annexe J.2)

II.III – Analyse et conclusion croisées

L'analyse des différences dans la structure du réseau en fonction des attributs individuels révèle que la transformation du réseau n'a pas bénéficié à tous les profils de la même manière. Si les indicateurs globaux montrent une progression générale en termes de connectivité, d'intégration et de densité, c'est en croisant les dimensions que l'on perçoit les effets différenciés du traitement.

Le cas des femmes non-core illustre clairement cette dynamique. Ce groupe, initialement moins bien intégré, ressort après traitement comme l'un des plus connectés, les moins isolés et les mieux insérés dans des groupes collaboratifs stables. À l'opposé, les femmes core, bien qu'ayant vu leur isolement disparaître, restent peu nombreuses et faiblement connectées, ce qui limite encore leur visibilité dans le réseau. Elles sont bien intégrées dans de petits groupes, mais restent en retrait dans l'ensemble du réseau. Les **hommes core, quant à eux, conservent une place centrale dans la structure du réseau**. Leur rôle de lien entre les différentes parties reste marqué, et ils continuent d'occuper des positions structurantes. Les hommes non-core, enfin, progressent également, mais de manière plus modérée.

Ces résultats montrent que la transformation du réseau n'a pas seulement augmenté le nombre de liens, mais a aussi permis **d'intégrer davantage de profils qui étaient moins connectés au départ**, tout en gardant les grands équilibres existants. Au final, **le réseau devient à la fois plus ouvert, plus équilibré et mieux organisé, aussi bien en termes de connexions que des types de chercheurs qui y sont reliés**. (Code annexe N)



(Code annexe M)

III – Détection de communautés

Après avoir examiné la structure globale du réseau ainsi que les différences selon les attributs des chercheurs, cette troisième partie s'intéresse à la manière dont les collaborations s'organisent en sous-groupes. L'objectif est d'**identifier des communautés**, c'est-à-dire des groupes de chercheurs plus fortement connectés entre eux qu'avec le reste du réseau. Pour cela, nous avons utilisé la **méthode de Louvain**, une approche classique pour détecter les communautés en se basant sur la structure des connexions du réseau. L'algorithme a été appliqué au réseau avant et après traitement, avec un paramètre de résolution fixé à 0,7 et en tenant compte du poids des liens entre les chercheurs. (Code annexe K)

Avant le traitement, 91 communautés ont été détectées. Toutefois, ce chiffre inclut 74 chercheurs totalement isolés (degré nul), chacun identifié comme une « communauté » à part entière en raison de leur absence de lien. En retirant ces cas particuliers, on identifie **en réalité 17 communautés fonctionnelles** dans le réseau actif. **Après traitement**, le nombre brut de communautés descend à 47, mais parmi celles-ci, 28 sont encore des chercheurs isolés, ramenant le nombre réel de **communautés actives à 19**.

Cette évolution montre une **réduction du morcellement du réseau** : le nombre de communautés actives diminue, mais leur taille moyenne augmente, signe que les anciens petits groupes se sont regroupés en entités plus larges et plus connectées. Cette tendance est cohérente avec les évolutions observées dans les métriques globales du graphe : le degré moyen passe de 3.38 à 5.40, le clustering augmente de 0.48 à 0.56, et le nombre total d'arêtes croît de plus de 60 %, ce qui indique un **resserrement de la structure** et une **fusion partielle des sous-groupes initiaux**. La baisse du nombre de communautés, combinée à l'augmentation de leur taille moyenne, montre que les petits groupes se sont regroupés pour former des ensembles plus grands et plus connectés.

Nous allons maintenant examiner plus en détail la structure et la composition des communautés détectées, afin de mieux comprendre les dynamiques de regroupement à l'œuvre dans le réseau collaboratif. Pour cela, nous nous concentrons sur les **cinq plus grandes communautés**, identifiées avant et après traitement, car ce sont elles qui structurent la majeure partie des interactions

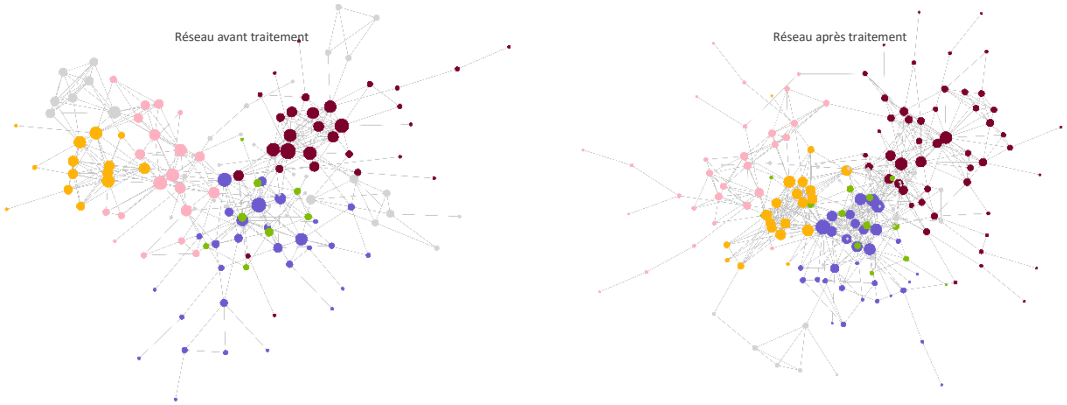
dans le réseau. Leur évolution en taille et en composition permet de mieux saisir les dynamiques de regroupement et d'intégration qui se sont mises en place.

Rang	1		2		3		4		5	
Traitement	Avant	Après	Avant	Après	Avant	Après	Avant	Après	Avant	Après
Nombre de membre	30	51	29	32	20	29	14	20	8	13
% de femme	26.7%	37.3%	17.2%	18.8%	10.0%	13.8%	14.3%	30.0%	25.0%	23.1%
% de core	10.0%	15.7%	31.0%	12.5%	25.0%	17.2%	7.1%	0.0%	0.0%	7.7%

Avant traitement, les plus grandes communautés comptaient entre 8 et 30 membres, tandis qu'après traitement, elles s'élargissent nettement, allant jusqu'à 51 membres. Cela confirme une fusion partielle des anciens sous-groupes, cohérente avec la **diminution du nombre total de communautés et l'augmentation des tailles moyennes**.

Concernant la **répartition des femmes, leur part augmente dans les grandes communautés**. Il est important de rappeler que les femmes ne représentent que 23 % de l'ensemble du réseau. Pourtant, après traitement, elles sont surreprésentées dans certaines communautés centrales. Par exemple, la plus grande communauté en compte 37,3 %, soit un taux nettement supérieur à la moyenne du réseau. La quatrième plus grande atteint 30 % de femmes, et même avant traitement, la plus grande communauté affichait déjà 26,7 % de femmes, ce qui restait au-dessus du taux global. Cela montre que les femmes ont non seulement gagné en connectivité globale, mais ont aussi été intégrées dans les principaux pôles du réseau, et pas uniquement dans des groupes périphériques. Leur présence accrue dans les grandes communautés reflète une forme d'inclusion structurelle réussie.

Du côté des **chercheurs core**, qui représentent 12 % du réseau, on observe une **présence initialement concentrée dans certaines communautés**. Avant traitement, la deuxième plus grande communauté comptait 31 % de core, et la troisième 25 %, tandis que d'autres en étaient presque absentes. Après traitement, la **répartition devient plus équilibrée**, avec des taux allant de 0 % à 17,2 %. Par exemple, la plus grande communauté après traitement compte 15,7 % de core, ce qui montre que ces profils ne monopolisent pas les grands pôles, et que les non-core y sont bien représentés. Ci-dessous, graphiques des réseaux avant et après traitement, avec les cinq plus grandes communautés colorées distinctement. (Code annexe O)



Conclusion

Cette étude a permis de mettre en évidence les principales transformations du réseau de collaborations après le traitement. Globalement, le réseau devient plus connecté, moins fragmenté, et davantage structuré autour de communautés denses. Les chercheurs auparavant en marge, notamment les femmes et les non-core, bénéficient d'une meilleure intégration. On observe également une réduction de l'homophilie de genre et une ouverture des sous-groupes, qui témoignent d'un réseau plus inclusif et plus équilibré dans sa structuration. Le traitement semble donc avoir joué un rôle positif en facilitant les échanges et en réduisant certaines inégalités de départ. Il a permis à des profils initialement moins bien connectés de se rapprocher du centre du réseau, tout en conservant une forme d'organisation distribuée, cohérente avec les logiques observées dans les réseaux scientifiques réels.

Cependant, cette étude repose sur un seul groupe de 215 chercheurs et deux périodes d'observation. Pour mieux évaluer la portée des résultats, il serait utile de disposer de données sur plusieurs points dans le temps, afin de voir si les évolutions observées se maintiennent ou s'estompent. Une amélioration juste après traitement ne garantit pas un changement durable des pratiques. Par ailleurs, croiser les données de réseau avec des indicateurs de production scientifique (nombres de publications, qualités des publications) permettrait d'évaluer si les chercheurs mieux connectés sont aussi plus productifs ou plus visibles. Cela offrirait une lecture complémentaire, reliant structure relationnelle et performance. Enfin, l'analyse gagnerait à intégrer d'autres attributs, comme l'ancienneté, le domaine de recherche ou le grade académique, afin d'identifier plus finement quels profils profitent le plus des changements. Cela permettrait d'aller au-delà du genre et du statut core, et d'explorer d'autres facteurs d'inégalités ou de différenciation dans les dynamiques collaboratives.

Code Analyse de Réseaux

A Préparation des données

```python
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
from networkx.algorithms import community
import numpy as np

# 1. Chargement des donnees
nodes = pd.read_csv("nodes_cluster_C.csv",sep=";")
edges = pd.read_csv("edges_cluster_C.csv",sep=";")
print(nodes.head())
print(edges.head())

# 2. Nettoyage : ne garder qu'une apparition de chaque dyade
edges_unique = edges[edges['source'] < edges['target']]

# 3. Creation des graphes pour chaque periode
pre_edges = edges_unique[edges_unique['pre_link'] == 1]   # Selection des aretes actives avant l'intervention
post_edges = edges_unique[edges_unique['post_link'] == 1] # Selection des aretes actives apres l'intervention

G_avant = nx.Graph()   # Creation du graphe non oriente pour la periode "pre"
G_apres = nx.Graph()   # Creation du graphe non oriente pour la periode "post"

# Ajouter les noeuds avec attributs
for _, row in nodes.iterrows():  # on neglige l'index avec _
    G_avant.add_node(row['id'], gender=row['gender'], core=row['core'], coord=row['coord'])   # Ajout d'un noeud dans G_avant avec ses attributs
    G_apres.add_node(row['id'], gender=row['gender'], core=row['core'], coord=row['coord'])   # Ajout du meme noeud dans G_apres avec les memes attributs

# Ajouter les aretes (liens)
```

```
G_avant.add_edges_from(pre_edges[['source', 'target']].values)   # Ajout des aretes dans G_avant a partir des colonnes source et target
G_apres.add_edges_from(post_edges[['source', 'target']].values)  # Ajout des aretes dans G_apres a partir des colonnes source et target
```

Listing 1 – Preparation des donnees

B Analyse des réseaux avant et après traitement

```
## Analyse du reseau avant traitement

print("Nombre de noeuds :", G_avant.number_of_nodes())
print("Nombre de liens :", G_avant.number_of_edges())

# Densite = nb de liens existants / nb de liens possibles
print("Densite :", nx.density(G_avant))

# Degre moyen = somme des degres / nombre de noeuds
degres_avant = dict(G_avant.degree())
print("Degre moyen :", sum(degres_avant.values()) / G_avant.number_of_nodes())

# Nombre de composantes connexes (sous-graphes separes)
print("Nombre de composantes connexes :", nx.number_connected_components(G_avant))

# Taille de la plus grande composante connexe (composante geante)
print("Taille composante geante :", len(max(nx.connected_components(G_avant), key=len)))

# Nombre de chercheurs isoles (degre 0 dans G_avant)
print("Chercheurs isoles :", len([n for n in G_avant.nodes if G_avant.degree(n) == 0]))

# Clustering moyen local (moyenne du taux de formation de triangles pour chaque noeud)
print("Clustering moyen :", nx.average_clustering(G_avant))

# Clustering global (transitivite = rapport triangles / triplets)
print("Clustering global :", nx.transitivity(G_avant))

# Longueur moyenne des plus courts chemins (sur la composante geante uniquement)
```

```
giant_component = G_avant.subgraph(max(nx.
    connected_components(G_avant), key=len))
print("Longueur moyenne des plus courts chemins :", nx.
    average_shortest_path_length(giant_component))


## Analyse du reseau apres traitement

print("Nombre de noeuds :", G_apres.number_of_nodes())
print("Nombre de liens :", G_apres.number_of_edges())

# Densite = nb de liens existants / nb de liens possibles
print("Densite :", nx.density(G_apres))

# Degre moyen = somme des degres / nombre de noeuds
degres_apres = dict(G_apres.degree())
print("Degre moyen :", sum(degres_apres.values()) / G_apres.
    number_of_nodes())

# Nombre de composantes connexes (sous-graphes separes)
print("Nombre de composantes connexes :", nx.
    number_connected_components(G_apres))

# Taille de la plus grande composante connexe (composante
    geante)
print("Taille composante geante :", len(max(nx.
    connected_components(G_apres), key=len)))

# Nombre de chercheurs isoles (degre 0 dans G_apres)
print("Chercheurs isoles :", len([n for n in G_avant.nodes
    if G_apres.degree(n) == 0]))

# Clustering moyen local
print("Clustering moyen :", nx.average_clustering(G_apres))

# Clustering global
print("Clustering global :", nx.transitivity(G_apres))

# Longueur moyenne des plus courts chemins
giant_component = G_apres.subgraph(max(nx.
    connected_components(G_apres), key=len))
print("Longueur moyenne des plus courts chemins :", nx.
    average_shortest_path_length(giant_component))
```

Listing 2 – Analyse du graphe avant et apres traitement

C Analyse des composantes géantes

```
## Analyse avant traitement
components_avant = list(nx.connected_components(G_avant))  # Liste des composantes
giant_avant = max(components_avant, key=len)               # Trouver la composante geante

print('Taille composante geante avant traitement :', len(giant_avant))

# Pourcentage de la population dans la composante
pourcentage_giant_avant = len(giant_avant) / G_avant.number_of_nodes() * 100
print('Pourcentage dans composante geante avant traitement :', pourcentage_giant_avant, '%')

# Extraction de la composante geante
Giant_avant = G_avant.subgraph(giant_avant)
distance_moyenne_avant = nx.average_shortest_path_length(Giant_avant)
print('Distance moyenne dans la composante geante avant traitement :', distance_moyenne_avant)


## Analyse apres traitement
components_apres = list(nx.connected_components(G_apres))
giant_apres = max(components_apres, key=len)

print('\nTaille composante geante apres traitement :', len(giant_apres))
pourcentage_giant_apres = len(giant_apres) / G_apres.number_of_nodes() * 100
print('Pourcentage dans composante geante apres traitement :', pourcentage_giant_apres, '%')

Giant_apres = G_apres.subgraph(giant_apres)
distance_moyenne_apres = nx.average_shortest_path_length(Giant_apres)
print('Distance moyenne dans la composante geante apres traitement :', distance_moyenne_apres)
```

Listing 3 – Analyse des composantes geantes avant et apres traitement

D Pourcentage de paires accessibles en $\leq h$ sauts

```python
# 1. Calcul des distances AVANT traitement
distances_avant = dict(nx.all_pairs_shortest_path_length(Giant_avant))  # Dictionnaire des plus courts chemins entre toutes les paires de noeuds du graphe avant traitement
all_distances_avant = []  # Liste pour stocker toutes les distances entre les paires de noeuds

for source in distances_avant:  # Parcours de chaque noeud source
    for target, d in distances_avant[source].items():  # Pour chaque noeud cible, on recupere la distance
        if source != target:  # On exclut les distances a soi-meme (0)
            all_distances_avant.append(d)  # On stocke la distance dans la liste

h_max_avant = max(all_distances_avant)  # Profondeur maximale (plus grande distance observee)
percentages_avant = []  # Liste pour stocker les pourcentages cumules

for h in range(1, h_max_avant+1):  # Pour chaque valeur de h (nombre de sauts)
    proportion = np.sum(np.array(all_distances_avant) <= h) / len(all_distances_avant)  # Proportion de paires connectees en <= h sauts
    percentages_avant.append(proportion * 100)  # Conversion en pourcentage et ajout a la liste

# 2. Calcul des distances APRES traitement (meme logique)
distances_apres = dict(nx.all_pairs_shortest_path_length(Giant_apres))  # Distances dans le graphe apres traitement
all_distances_apres = []

for source in distances_apres:
    for target, d in distances_apres[source].items():
        if source != target:
            all_distances_apres.append(d)

h_max_apres = max(all_distances_apres)  # Profondeur maximale apres traitement
percentages_apres = []

for h in range(1, h_max_apres+1):
```

```
    proportion = np.sum(np.array(all_distances_apres) <= h)
        / len(all_distances_apres)
    percentages_apres.append(proportion * 100)

# 3. Trace des deux courbes sur le meme graphique
h_max_total = max(h_max_avant, h_max_apres)  # Profondeur
    maximale globale pour le graphique

plt.figure(figsize=(8,6))  # Taille de la figure
plt.plot(range(1, h_max_avant+1), percentages_avant, marker=
    'o', color='#486EAA', label='Avant traitement')  # Courbe
     "avant"
plt.plot(range(1, h_max_apres+1), percentages_apres, marker=
    'o', color='#7C6CA9', label='Apres traitement')  # Courbe
     "apres"

plt.title('Pourcentage de paires accessibles en <= h sauts')
      # Titre du graphique
plt.xlabel('h (nombre de sauts)')  # Legende axe x
plt.ylabel('Pourcentage de paires (%)')  # Legende axe y
plt.legend()  # Affichage de la legende
plt.grid(True)  # Grille
plt.show()  # Affichage du graphique
```

Listing 4 – Pourcentage de paires accessibles en h sauts ou moins

E Distribution du degré

E.1 Histogrammes et distribution log-log du degre

```
# === Histogramme des degres - AVANT ===
degree_count_avant = nx.degree_histogram(G_avant) # nombre
    de noeuds par degre (index = degre)
degree_avant = list(range(len(degree_count_avant))) # liste
    des degres possibles
hist_data_avant = (degree_avant, degree_count_avant) # tuple
     pour l'affichage

# === Histogramme des degres - APRES ===
degree_count_apres = nx.degree_histogram(G_apres)
degree_apres = list(range(len(degree_count_apres)))
hist_data_apres = (degree_apres, degree_count_apres)

# === Bornes communes ===
ymax = max(max(degree_count_avant), max(degree_count_apres))
     + 10 # meme echelle Y avec une marge
xmax = max(len(degree_count_avant), len(degree_count_apres))
     # meme echelle X (degre max)
```

```
14
15 # === Affichage - AVANT ===
16 plt.figure(figsize=(6, 4))
17 plt.bar(*hist_data_avant, color='#486EAA') # histogramme
18 plt.title('Distribution du degre - Avant traitement')
19 plt.xlabel('Degre')
20 plt.ylabel('Nombre de noeuds')
21 plt.ylim(0, ymax) # echelle Y uniforme
22 plt.xlim(0, xmax) # echelle X uniforme
23 plt.grid(axis='y')
24 plt.tight_layout()
25 plt.show()
26
27 # === Affichage - APRES ===
28 plt.figure(figsize=(6, 4))
29 plt.bar(*hist_data_apres, color='#7C6CA9')
30 plt.title('Distribution du degre - Apres traitement')
31 plt.xlabel('Degre')
32 plt.ylabel('Nombre de noeuds')
33 plt.ylim(0, ymax)
34 plt.xlim(0, xmax)
35 plt.grid(axis='y')
36 plt.tight_layout()
37 plt.show()
38
39 # === Distribution log-log - AVANT ===
40 total_nodes_avant = G_avant.number_of_nodes() # total de
       noeuds dans le graphe
41 degree_distribution_avant = {} # dictionnaire : degre ->
       frequence (%)
42
43 for k, v in enumerate(degree_count_avant):
44     if v > 0 and k > 0: # on elimine les zeros (log(0)
           interdit)
45         degree_distribution_avant[k] = v / total_nodes_avant
46
47 x_axis_avant = list(degree_distribution_avant.keys())
48 y_axis_avant = list(degree_distribution_avant.values())
49
50 # === Distribution log-log - APRES ===
51 total_nodes_apres = G_apres.number_of_nodes()
52 degree_distribution_apres = {}
53
54 for k, v in enumerate(degree_count_apres):
55     if v > 0 and k > 0:
56         degree_distribution_apres[k] = v / total_nodes_apres
57
58 x_axis_apres = list(degree_distribution_apres.keys())
59 y_axis_apres = list(degree_distribution_apres.values())
60
```

```
# === Echelle log-log commune ===
ymin = min(min(y_axis_avant), min(y_axis_apres))
ymax = max(max(y_axis_avant), max(y_axis_apres))

# === Affichage log-log - AVANT ===
plt.figure(figsize=(6, 4))
plt.title('Distribution du degre (log-log) - Avant traitement')
plt.xlabel('Degre\n(echelle log)')
plt.ylabel('Fraction des noeuds\n(echelle log)')
plt.xscale("log", base=10)
plt.yscale("log", base=10)
plt.xlim(min(x_axis_avant)/2, max(x_axis_avant)*2) #on fait /2 et *2 pour agrandir l'echelle
plt.ylim(ymin/2, ymax * 2)
plt.scatter(x_axis_avant, y_axis_avant, color='#486EAA', s=30, marker='+')
plt.show()

# === Affichage log-log - APRES ===
plt.figure(figsize=(6, 4))
plt.title('Distribution du degre (log-log) - Apres traitement')
plt.xlabel('Degre\n(echelle log)')
plt.ylabel('Fraction des noeuds\n(echelle log)')
plt.xscale("log", base=10)
plt.yscale("log", base=10)
plt.xlim(min(x_axis_apres)/2, max(x_axis_apres)*2)
plt.ylim(ymin/2, ymax * 2)
plt.scatter(x_axis_apres, y_axis_apres, color='#7C6CA9', s=30, marker='+')
plt.show()
```

Listing 5 – Histogrammes et distribution log-log du degre

E.2 Ajustement d une loi de puissance

```
import powerlaw



# === Ajustement - AVANT traitement ===
data = [d for n, d in G_avant.degree()]  # Extraire les degres du graphe apres traitement
# Ajustement d une loi de puissance par maximum de vraisemblance (xmin fixe a 4)
fit = powerlaw.Fit(data, discrete=True, xmin=4, fit_method='Likelihood')
```

```
# Parametres estimes : alpha (exposant), xmin (seuil), sigma (erreur standard)
alpha, xmin, sigma = fit.alpha, fit.xmin, fit.sigma
# Trace log-log : donnees empiriques + ajustement
fig, ax = plt.subplots()
fit.plot_pdf(ax=ax, color="#7C6CA9", linewidth=2, label='Donnees empiriques (log bins)')
fit.power_law.plot_pdf(ax=ax, color='orange', linestyle='-', label='Ajustement loi de puissance')
ax.set_title(f'Distribution Log-Log - Avant traitement\n(alpha={alpha:.2f}, xmin={xmin:.2f})')
ax.set_xlabel('Valeur')
ax.set_ylabel('Frequence')
ax.legend()
plt.show()

# === Ajustement - APRES traitement ===
data = [d for n, d in G_apres.degree()] # Extraire les degres du graphe apres traitement
# Ajustement d une loi de puissance par maximum de vraisemblance (xmin fixe a 4)
fit = powerlaw.Fit(data, discrete=True, xmin=4, fit_method='Likelihood')
# Parametres estimes : alpha (exposant), xmin (seuil), sigma (erreur standard)
alpha, xmin, sigma = fit.alpha, fit.xmin, fit.sigma

# Trace log-log : donnees empiriques + ajustement
fig, ax = plt.subplots()
fit.plot_pdf(ax=ax, color="#486EAA", linewidth=2, label='Donnees empiriques (log bins)')
fit.power_law.plot_pdf(ax=ax, color='orange', linestyle='-', label='Ajustement loi de puissance')
ax.set_title(f'Distribution Log-Log - Apres traitement\n(alpha={alpha:.2f}, xmin={xmin:.2f})')
ax.set_xlabel('Valeur')
ax.set_ylabel('Frequence')
ax.legend()
plt.show()
```

Listing 6 – Ajustement d une loi de puissance

F Attachement Préférenciel

```python
from collections import Counter

# 1. Noeuds isoles dans G_avant
isolated_nodes = [n for n in G_avant.nodes() if G_avant.degree(n) == 0]

# 2. Noeuds actifs (avant isoles, apres connectes)
activated_nodes = [n for n in isolated_nodes if G_apres.degree(n) > 0]

# 3. Compter les connexions selon le degre k des cibles (dans G_avant)
k_targets = []
seen_pairs = set()  # pour eviter les doublons

for node in activated_nodes:
    for neighbor in G_apres.neighbors(node):
        if not G_avant.has_edge(node, neighbor):
          # Creer une paire unique pour eviter les doublons
            pair = tuple(sorted((node, neighbor)))
            if pair not in seen_pairs:
                seen_pairs.add(pair)
                  # Maintenant, choisir intelligemment la "cible"
                if neighbor in isolated_nodes:
                   # Si le neighbor est aussi un isole, on decide arbitrairement : on prend node
                    degree_target = G_avant.degree(node)
                else:
                    degree_target = G_avant.degree(neighbor)

                k_targets.append(degree_target)

# Ensuite : k_targets contient vraiment les degres des vraies cibles (sans compter deux fois les liens entre isoles)
Pk_counts = Counter(k_targets)

#  4. n_k : nombre de noeuds avec degre k dans G_avant
deg_pre_all = dict(G_avant.degree())
nk_counts = Counter(deg_pre_all.values())

# 5. N : nombre total de noeuds dans G_avant
N_total = G_avant.number_of_nodes()

# 6. Calcul de R_k
Rk = {}
```

```
for k in Pk_counts:
    if nk_counts[k] > 0:
        Rk[k] = Pk_counts[k] * N_total / nk_counts[k]

# 7. Visualisation de R_k
plt.figure(figsize=(8,6))
plt.plot(sorted(Rk.keys()), [Rk[k] for k in sorted(Rk)],
    marker='o')
plt.xlabel("Degre k dans G_pre")
plt.ylabel("R_k (preference relative)")
plt.title("Preference d'attachement selon le degre initial")
plt.grid(True)
plt.show()

# 8. Calcul de rho_k (somme cumulative)
#Rk doit etre ordonne par k
sorted_ks = sorted(Rk.keys())
sorted_Rk = [Rk[k] for k in sorted_ks]
rho_k = np.cumsum(sorted_Rk)

# Trace de rho_k
plt.figure(figsize=(8,6))
plt.plot(sorted_ks, rho_k, marker='o')
plt.xlabel("Degre k dans G_pre")
plt.ylabel("rho_k (somme cumulative de R_k)")
plt.title("Preference d'attachement cumulee")
plt.grid(True)
plt.show()
```

Listing 7 – Preference d attachement entre G_avant et G_apres

G Comparaison avec des réseaux théoriques

```
# === 1. Choix du reseau reel  ===
G_real = G_apres

# === 2. Caracteristiques du reseau reel ===
n = G_real.number_of_nodes()                    # nombre de noeuds
m = G_real.number_of_edges()                    # nombre d'aretes
density = nx.density(G_real)                    # densite du graphe

print(f"Notre reseau : {n} noeuds, {m} aretes, densite {density:.4f}")

# === 3. Generation des graphes de comparaison ===

# a) Graphe aleatoire Erdos-Renyi avec meme n et meme densite
p_er = density
G_er = nx.erdos_renyi_graph(n, p_er)

# b) Graphe small-world Watts-Strogatz avec meme n et degre moyen k ~ 2m/n
k = int(round(2 * m / n))
G_sw = nx.watts_strogatz_graph(n, k, 0.1)

# c) Graphe scale-free Barabasi-Albert avec m' ~ m/n (liens ajoutes a chaque nouveau noeud)
m_ba = max(1, int(round(m / n)))
G_ba = nx.barabasi_albert_graph(n, m_ba)

# d) Graphe regulier : chaque noeud a degre k (on ajuste k si necessaire)
if (n * k) % 2 != 0:
    k -= 1  # on rend n*k pair
G_reg = nx.random_regular_graph(k, n)

# === 4. Mesure des proprietes des graphes ===

def get_props(G):
    try:
        comp = max(nx.connected_components(G), key=len)
                    # plus grande composante connexe
        G_cc = G.subgraph(comp)
        avg_path = nx.average_shortest_path_length(G_cc)
                   # longueur moyenne du plus court chemin
    except:
```

```
        avg_path = None                                                 # sinon None
    clustering = nx.transitivity(G)                                     # clustering global
    return avg_path, clustering

# Reel
real_avg, real_clust = get_props(G_real)
# Aleatoire
er_avg, er_clust = get_props(G_er)
# Small-world
sw_avg, sw_clust = get_props(G_sw)
# Scale-free
ba_avg, ba_clust = get_props(G_ba)
# Regulier
reg_avg, reg_clust = get_props(G_reg)

# === 5. Tableau comparatif ===

df_compare = pd.DataFrame({
    'Type': ['Reel', 'Aleatoire (ER)', 'Small-World (WS)', 'Scale-Free (BA)', 'Regulier (k-regular)'],
    'Avg Shortest Path': [real_avg, er_avg, sw_avg, ba_avg, reg_avg],
    'Global Clustering': [real_clust, er_clust, sw_clust, ba_clust, reg_clust]
})

print("\nComparaison des proprietes :")
print(df_compare)
```

Listing 8 – Comparaison du reseau reel avec des modeles generatifs

H Analyse par genre

```
# Proportions hommes/femmes
prop_gender = nodes['gender'].value_counts(normalize=True)
print(f"Proportion d'hommes (0) : {prop_gender.get(0, 0):.3f} ({prop_gender.get(0, 0)*100:.1f}%)")
print(f"Proportion de femmes (1) : {prop_gender.get(1, 0):.3f} ({prop_gender.get(1, 0)*100:.1f}%)")

# 1. Identifier les individus isoles dans G_avant ou G_apres
isolated_nodes_pre = list(nx.isolates(G_avant))
isolated_nodes_post = list(nx.isolates(G_apres))

# 2. Joindre avec les attributs pour connaitre leur genre
isolated_pre_df = nodes[nodes['id'].isin(isolated_nodes_pre)]
isolated_post_df = nodes[nodes['id'].isin(
    isolated_nodes_post)]

# 3. Calculer les totaux par genre
total_femmes = (nodes['gender'] == 1).sum()
total_hommes = (nodes['gender'] == 0).sum()

isolated_femmes_pre = (isolated_pre_df['gender'] == 1).sum()
isolated_hommes_pre = (isolated_pre_df['gender'] == 0).sum()

isolated_femmes_post = (isolated_post_df['gender'] == 1).sum()
isolated_hommes_post = (isolated_post_df['gender'] == 0).sum()

# 4. Calculer les pourcentages
pct_femmes_isolees_pre = isolated_femmes_pre / total_femmes * 100
pct_hommes_isoles_pre = isolated_hommes_pre / total_hommes * 100

pct_femmes_isolees_post = isolated_femmes_post / total_femmes * 100
pct_hommes_isoles_post = isolated_hommes_post / total_hommes * 100

# 5. Affichage
print("--- Pourcentage d'isoles par genre ---")
print(f"Avant traitement (G_avant) :")
print(f"Femmes isolees : {pct_femmes_isolees_pre:.2f}%")
print(f"Hommes isoles : {pct_hommes_isoles_pre:.2f}%")

print("\nApres traitement (G_apres) :")
```

```
print(f"Femmes isolees : {pct_femmes_isolees_post:.2f}%")
print(f"Hommes isoles : {pct_hommes_isoles_post:.2f}%")

#  Degres
deg_pre = dict(G_avant.degree())
deg_post = dict(G_apres.degree())

# Creation du DataFrame
df_degree = pd.DataFrame({
    'id': list(deg_pre.keys()),
    'deg_pre': [deg_pre[n] for n in deg_pre],
    'deg_post': [deg_post[n] for n in deg_post]
})

# Ajout de l'information de genre
df_degree = df_degree.merge(nodes[['id', 'gender']], on='id'
    )

# Calcul de l'evolution du degre
df_degree['degree_change'] = df_degree['deg_post'] -
    df_degree['deg_pre']

# Moyenne des degres avant traitement par genre
mean_pre = df_degree.groupby('gender')['deg_pre'].mean()

# Moyenne des degres apres traitement par genre
mean_post = df_degree.groupby('gender')['deg_post'].mean()

# Evolution moyenne du degre par genre
mean_change = df_degree.groupby('gender')['degree_change'].
    mean()

# Affichage
print("Degre moyen AVANT traitement :")
print(mean_pre)

print("\nDegre moyen APRES traitement :")
print(mean_post)

print("\nEvolution moyenne du degre (apres - avant) :")
print(mean_change)


# Clustering
clust_pre = nx.clustering(G_avant)
clust_post = nx.clustering(G_apres)

# Creation du DataFrame
df_clust = pd.DataFrame({
    'id': list(clust_pre.keys()),
```

```
    'clust_pre': [clust_pre[n] for n in clust_pre],
    'clust_post': [clust_post[n] for n in clust_post]
})

# Ajout de l'information de genre
df_clust = df_clust.merge(nodes[['id', 'gender']], on='id')

# Comparaison femmes vs hommes
result_pre = df_clust.groupby('gender')['clust_pre'].mean()
result_post = df_clust.groupby('gender')['clust_post'].mean
    ()

print("Clustering moyen avant traitement :")
print(result_pre)

print("\nClustering moyen apres traitement :")
print(result_post)


# Centralite
closeness_pre = nx.closeness_centrality(G_avant)
betweenness_pre = nx.betweenness_centrality(G_avant)
closeness_post = nx.closeness_centrality(G_apres)
betweenness_post = nx.betweenness_centrality(G_apres)

df_centrality = pd.DataFrame({
    'id': list(closeness_pre.keys()),
    'closeness_pre': [closeness_pre[n] for n in
        closeness_pre],
    'betweenness_pre': [betweenness_pre[n] for n in
        betweenness_pre],
    'closeness_post': [closeness_post[n] for n in
        closeness_post],
    'betweenness_post': [betweenness_post[n] for n in
        betweenness_post]
})
df_centrality = df_centrality.merge(nodes[['id', 'gender']],
     on='id')

mean_closeness_pre = df_centrality.groupby('gender')['
    closeness_pre'].mean()
mean_closeness_post = df_centrality.groupby('gender')['
    closeness_post'].mean()
mean_betweenness_pre = df_centrality.groupby('gender')['
    betweenness_pre'].mean()
mean_betweenness_post = df_centrality.groupby('gender')['
    betweenness_post'].mean()

print("Closeness moyen avant :", mean_closeness_pre)
print("Closeness moyen apres :", mean_closeness_post)
```

```
print("Betweenness moyen avant :", mean_betweenness_pre)
print("Betweenness moyen apres :", mean_betweenness_post)
```

Listing 9 – Analyse comparative par genre : isolation, degre, clustering, centralite

I Recherche d'un nœud central à la manière d'Erdős

```
# Top 10 degres dans G_avant
top_degres_avant = sorted(G_avant.degree, key=lambda x: x
    [1], reverse=True)[:10]
print("Top 10 degre avant traitement :")
for node, deg in top_degres_avant:
    print(f"Chercheur {node} - Degre : {deg}")

# Top 10 degres dans G_apres
top_degres_apres = sorted(G_apres.degree, key=lambda x: x
    [1], reverse=True)[:10]
print("\nTop 10 degre apres traitement :")
for node, deg in top_degres_apres:
    print(f"Chercheur {node} - Degre : {deg}")

# Closeness dans G_avant
closeness_avant = nx.closeness_centrality(G_avant)
top_closeness_avant = sorted(closeness_avant.items(), key=
    lambda x: x[1], reverse=True)[:10]
print("\nTop 10 closeness avant traitement :")
for node, closeness in top_closeness_avant:
    print(f"Chercheur {node} - Closeness : {closeness}")

# Closeness dans G_apres
closeness_apres = nx.closeness_centrality(G_apres)
top_closeness_apres = sorted(closeness_apres.items(), key=
    lambda x: x[1], reverse=True)[:10]
print("\nTop 10 closeness apres traitement :")
for node, closeness in top_closeness_apres:
    print(f"Chercheur {node} - Closeness : {closeness}")

# Betweenness dans G_avant
betweenness_avant = nx.betweenness_centrality(G_avant)
top_betweeneess_avant = sorted(betweenness_avant.items(),
    key=lambda x: x[1], reverse=True)[:10]
print("\nTop 10 betweenness avant traitement :")
for node, betweeness in top_betweeneess_avant:
    print(f"Chercheur {node} - Betweenness : {betweeness}")

# Betweenness dans G_apres
betweenness_apres = nx.betweenness_centrality(G_apres)
```

```
top_betweeneess_apres = sorted(betweenness_apres.items(),
    key=lambda x: x[1], reverse=True)[:10]
print("\nTop 10 betweenness apres traitement :")
for node, betweeness in top_betweeneess_apres:
    print(f"Chercheur {node} - Betweenness : {betweeness}")
```

Listing 10 – Top 10 des chercheurs par degre, closeness et betweenness

J Homophilie (assortativité)

J.1 Homophilie par genre

```
# Assortativite par genre
assort_genre_avant = nx.attribute_assortativity_coefficient(
    G_avant, 'gender')
print("Assortativite par genre avant traitement :",
    assort_genre_avant)

assort_genre_apres = nx.attribute_assortativity_coefficient(
    G_apres, 'gender')
print("Assortativite par genre apres traitement :",
    assort_genre_apres)

# Degres selon le genre
deg_avant, deg_apres = dict(G_avant.degree()), dict(G_apres.
    degree())
gen_avant, gen_apres = nx.get_node_attributes(G_avant, '
    gender'), nx.get_node_attributes(G_apres, 'gender')

f_avant = [deg_avant[n] for n in G_avant if gen_avant[n] ==
    1]
m_avant = [deg_avant[n] for n in G_avant if gen_avant[n] ==
    0]
f_apres = [deg_apres[n] for n in G_apres if gen_apres[n] ==
    1]
m_apres = [deg_apres[n] for n in G_apres if gen_apres[n] ==
    0]

# Moyennes et taux
moy_f_avant, moy_f_apres = np.mean(f_avant), np.mean(f_apres
    )
moy_m_avant, moy_m_apres = np.mean(m_avant), np.mean(m_apres
    )

taux_f = (moy_f_apres - moy_f_avant) / moy_f_avant * 100
taux_m = (moy_m_apres - moy_m_avant) / moy_m_avant * 100

```

```
print(f"Taux d'augmentation du degre moyen des femmes : {taux_f}%")
print(f"Taux d'augmentation du degre moyen des hommes : {taux_m}%")
```

Listing 11 – Homophilie par genre et evolution du degre

J.2 Homophilie par appartenance au noyau (core)

```
# Assortativite par core
assort_core_avant = nx.attribute_assortativity_coefficient(G_avant, 'core')
print("Assortativite par core avant traitement :", assort_core_avant)

assort_core_apres = nx.attribute_assortativity_coefficient(G_apres, 'core')
print("Assortativite par core apres traitement :", assort_core_apres)

# Degres selon core
deg_avant, deg_apres = dict(G_avant.degree()), dict(G_apres.degree())
core_avant, core_apres = nx.get_node_attributes(G_avant, 'core'), nx.get_node_attributes(G_apres, 'core')

core_avant_vals = [deg_avant[n] for n in G_avant if core_avant[n] == 1]
noncore_avant_vals = [deg_avant[n] for n in G_avant if core_avant[n] == 0]
core_apres_vals = [deg_apres[n] for n in G_apres if core_apres[n] == 1]
noncore_apres_vals = [deg_apres[n] for n in G_apres if core_apres[n] == 0]

# Moyennes et taux
moy_core_av, moy_core_ap = np.mean(core_avant_vals), np.mean(core_apres_vals)
moy_nc_av, moy_nc_ap = np.mean(noncore_avant_vals), np.mean(noncore_apres_vals)

taux_core = (moy_core_ap - moy_core_av) / moy_core_av * 100
taux_nc = (moy_nc_ap - moy_nc_av) / moy_nc_av * 100

print(f"Taux d'augmentation du degre moyen - Core     : {taux_core}%")
print(f"Taux d'augmentation du degre moyen - Non-core : {taux_nc}%")
```

Listing 12 – Homophilie par core et evolution du degre

K Détection de communautés (Louvain)

```
# Nombre de communautes de taille >= 2
nb_communautes_size_2plus = sum(1 for c in partition_pre if len(c) >= 2)
print(f"Nombre de communautes de taille >= 2 avant le choc : {nb_communautes_size_2plus}")

nb_communautes_post_2plus = sum(1 for c in partition_post if len(c) >= 2)
print(f"Nombre de communautes >= 2 apres le choc : {nb_communautes_post_2plus}")

# Total de femmes dans le reseau (sert a calculer le % de femmes captees)
total_f = nodes[nodes['gender'] == 1].shape[0]

# Fonction pour afficher les stats d'une liste de communautes
def afficher_stats_communautes(partition, top_k, titre):
    print(f"\n{titre}")
    for i, comm in enumerate(sorted(partition, key=len, reverse=True)[:top_k], 1):
        df = nodes[nodes['id'].isin(comm)]  # Sous-ensemble des individus dans la communaute
        n = len(df)  # Taille de la communaute
        if n == 0:
            print(f" - Com {i} : vide")  # Eviter division par zero
            continue
        f = (df['gender'] == 1).sum()  # Nombre de femmes
        c = (df['core'] == 1).sum()    # Nombre de membres core
        # Affichage du nombre de noeuds, % de femmes, % de core, % de femmes captees
        print(f" - Com {i} : {n} noeuds | % femmes : {f/n*100:.1f}% | % core : {c/n*100:.1f}% | femmes captees : {f/total_f*100:.1f}%")

# Appel pour les communautes avant et apres traitement
afficher_stats_communautes(partition_pre, 17, "Top 17 communautes AVANT le choc")
afficher_stats_communautes(partition_post, 19, "Top 19 communautes APRES le choc")
```

Listing 13 – Analyse des communautes detectees avant et apres le choc

L Analyse du noyau (core) : degre, isolement et centralite

```
# ====== AVANT ======

# Construction du DataFrame : chaque noeud de G_avant devient une ligne avec son degre
data_avant = pd.DataFrame.from_dict(dict(G_avant.degree()), orient='index', columns=['degree'])

# Ajout de l'attribut 'core' (1 si core, 0 sinon) pour chaque noeud
data_avant['core'] = pd.Series(nx.get_node_attributes(G_avant, 'core'))

# Affichage du degre moyen des membres core et non-core
print("Degre moyen par core - AVANT")
print(data_avant.groupby('core')['degree'].mean())


# ====== APRES ======

# Meme construction pour G_apres : degre et attribut core
data_apres = pd.DataFrame.from_dict(dict(G_apres.degree()), orient='index', columns=['degree'])
data_apres['core'] = pd.Series(nx.get_node_attributes(G_apres, 'core'))

# Affichage du degre moyen par core
print("\nDegre moyen par core - APRES")
print(data_apres.groupby('core')['degree'].mean())


# ====== Chercheurs isoles ======

# Ajout d'une colonne 'isole' : True si le degre est nul (degre == 0)
data_avant['isole'] = data_avant['degree'] == 0
data_apres['isole'] = data_apres['degree'] == 0

# Proportion d'isoles par core (moyenne de la colonne booleenne)
print("Proportion de chercheurs isoles par core - AVANT")
print(data_avant.groupby('core')['isole'].mean())
```

```
34
35 print("\nProportion de chercheurs isoles par core - APRES")
36 print(data_apres.groupby('core')['isole'].mean())
37
38
39 # ====== Centralites AVANT ======
40
41 # Calcul du clustering local pour chaque noeud
42 data_avant['clustering'] = pd.Series(nx.clustering(G_avant))
43
44 # Calcul de la betweenness centrality (centralite d'
       intermediarite)
45 data_avant['betweenness'] = pd.Series(nx.
       betweenness_centrality(G_avant))
46
47 # Calcul de la closeness centrality (centralite de proximite
       )
48 data_avant['closeness'] = pd.Series(nx.closeness_centrality(
       G_avant))
49
50
51 # ====== Centralites APRES ======
52
53 data_apres['clustering'] = pd.Series(nx.clustering(G_apres))
54 data_apres['betweenness'] = pd.Series(nx.
       betweenness_centrality(G_apres))
55 data_apres['closeness'] = pd.Series(nx.closeness_centrality(
       G_apres))
56
57
58 # ====== Moyennes par core ======
59
60 # Moyennes des 3 centralites groupees par core (0 ou 1)
61 print("Mesures centrales - AVANT")
62 print(data_avant.groupby('core')[['clustering', 'betweenness
       ', 'closeness']].mean())
63
64 print("\nMesures centrales - APRES")
65 print(data_apres.groupby('core')[['clustering', 'betweenness
       ', 'closeness']].mean())
```

Listing 14 – Degre, isolement et mesures centrales par core

M Représentation graphique du réseau avant et après traitement

```
from matplotlib.patches import Patch

# Graphe avant traitement
G = G_avant

# Position des noeuds avec le layout de type ressort (k controle l'espacement)
pos = nx.spring_layout(G, k=0.3)

# Couleur des noeuds selon le genre : bleu pour homme (0), rose pour femme (1)
node_colors = ['#6A8FBF' if G.nodes[n]['gender'] == 0 else '#C97BA9' for n in G.nodes]

# Taille des noeuds proportionnelle au degre (normalisee entre 70 et 770)
raw_degrees = dict(G.degree())
min_deg = min(raw_degrees.values())
max_deg = max(raw_degrees.values())
node_sizes = [70 + ((raw_degrees[n] - min_deg) / (max_deg - min_deg)) * 700 for n in G.nodes]

# Etiquette "C" en gras uniquement pour les noeuds core
labels = {n: r"$\bf{C}$" if G.nodes[n]['core'] == 1 else "" for n in G.nodes}

# Affichage du graphe avant
plt.figure(figsize=(12, 10))
nx.draw_networkx_edges(G, pos, alpha=0.3, width=0.5)
nx.draw_networkx_nodes(G, pos, node_color=node_colors, node_size=node_sizes)
nx.draw_networkx_labels(G, pos, labels=labels, font_size=9)

# Legende des couleurs et du label core
legend_elements = [
    Patch(facecolor='#6A8FBF', label='Homme'),
    Patch(facecolor='#C97BA9', label='Femme'),
    Patch(facecolor='none', edgecolor='black', label='C = Core (etiquete)')
]

plt.legend(handles=legend_elements, loc='upper right')
plt.axis('off')
plt.tight_layout()
plt.show()

```

```python
# Graphe apres traitement
G = G_apres
pos = nx.spring_layout(G, k=0.3)
node_colors = ['#6A8FBF' if G.nodes[n]['gender'] == 0 else '#C97BA9' for n in G.nodes]
raw_degrees = dict(G.degree())
min_deg = min(raw_degrees.values())
max_deg = max(raw_degrees.values())
node_sizes = [70 + ((raw_degrees[n] - min_deg) / (max_deg - min_deg)) * 700 for n in G.nodes]
labels = {n: r"$\bf{C}$" if G.nodes[n]['core'] == 1 else "" for n in G.nodes}

# Affichage du graphe apres
plt.figure(figsize=(12, 10))
nx.draw_networkx_edges(G, pos, alpha=0.3, width=0.5)
nx.draw_networkx_nodes(G, pos, node_color=node_colors, node_size=node_sizes)
nx.draw_networkx_labels(G, pos, labels=labels, font_size=9)

legend_elements = [
    Patch(facecolor='#6A8FBF', label='Homme'),
    Patch(facecolor='#C97BA9', label='Femme'),
    Patch(facecolor='none', edgecolor='black', label='C = Core (etiquete)')
]

plt.legend(handles=legend_elements, loc='upper right')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Listing 15 – Visualisation du graphe avant et apres traitement

N Calcul des mesures par genre et statut *core*

```python
# === AVANT traitement ===

# Creation d'un DataFrame contenant le degre de chaque noeud
data_avant = pd.DataFrame.from_dict(dict(G_avant.degree()), orient='index', columns=['degree'])

# Ajout des attributs de genre et de core (centre) a chaque noeud
data_avant['gender'] = pd.Series(nx.get_node_attributes(G_avant, 'gender'))
data_avant['core'] = pd.Series(nx.get_node_attributes(G_avant, 'core'))
```

```
9
10 # Ajout d'une colonne booleenne indiquant si le noeud est
       isole (degre = 0)
11 data_avant['isole'] = data_avant['degree'] == 0
12
13 # Calcul des mesures de centralite pour chaque noeud
14 data_avant['clustering'] = pd.Series(nx.clustering(G_avant))
15 data_avant['betweenness'] = pd.Series(nx.
       betweenness_centrality(G_avant))
16 data_avant['closeness'] = pd.Series(nx.closeness_centrality(
       G_avant))
17
18 # Affichage de la moyenne des mesures par genre et statut
       core
19 print("Mesures AVANT traitement par genre + core :")
20 print(data_avant.groupby(['gender', 'core'])[['degree', '
       isole', 'clustering', 'betweenness', 'closeness']].mean()
       )
21
22 # === APRES traitement ===
23
24 # Meme procedure appliquee au graphe apres traitement
25 data_apres = pd.DataFrame.from_dict(dict(G_apres.degree()),
       orient='index', columns=['degree'])
26 data_apres['gender'] = pd.Series(nx.get_node_attributes(
       G_apres, 'gender'))
27 data_apres['core'] = pd.Series(nx.get_node_attributes(
       G_apres, 'core'))
28 data_apres['isole'] = data_apres['degree'] == 0
29 data_apres['clustering'] = pd.Series(nx.clustering(G_apres))
30 data_apres['betweenness'] = pd.Series(nx.
       betweenness_centrality(G_apres))
31 data_apres['closeness'] = pd.Series(nx.closeness_centrality(
       G_apres))
32
33 # Affichage des moyennes par genre et core pour les mesures
       calculees
34 print("\nMesures APRES traitement par genre + core :")
35 print(data_apres.groupby(['gender', 'core'])[['degree', '
       isole', 'clustering', 'betweenness', 'closeness']].mean()
       )
```

Listing 16 – Calcul des mesures structurelles avant et apres traitement

O Visualisation des communautés avant et après traitement

```
# === COMMUNAUTES AVANT TRAITEMENT ===

# Classer les communautes par taille decroissante
partition_sorted = sorted(communities_avant, key=len,
    reverse=True)

# Attribuer l'indice de communaute a chaque noeud
for node in G_avant.nodes:
    for i, comm in enumerate(partition_sorted):
        if node in comm:
            G_avant.nodes[node]["community"] = i + 1

# Couleurs pour les 5 plus grosses communautes, sinon gris
    clair
node_colors = []
for node in G_avant.nodes(data=True):
    if node[1]["community"] == 1:
        node_colors.append("xkcd:bordeaux")
    elif node[1]["community"] == 2:
        node_colors.append("xkcd:light indigo")
    elif node[1]["community"] == 3:
        node_colors.append("xkcd:soft pink")
    elif node[1]["community"] == 4:
        node_colors.append("xkcd:amber")
    elif node[1]["community"] == 5:
        node_colors.append("xkcd:dark lime green")
    else:
        node_colors.append("lightgrey")

# Positionnement des noeuds
pos = nx.kamada_kawai_layout(G_avant)

# Taille des noeuds selon le degre (normalisee)
degrees = dict(G_avant.degree())
degrees = {n: (v - min(degrees.values())) / (max(degrees.
    values()) - min(degrees.values())) * 10000 for n, v in
    degrees.items()}

# Affichage du graphe
plt.rcParams['figure.figsize'] = [70, 55]
nx.draw_networkx(G_avant, pos=pos, node_color=node_colors,
                 node_size=[v for v in degrees.values()],
                 edge_color='silver', width=3.0, with_labels
                     =False)

plt.axis('off')
```

```
42 plt.show()
43
44 # === COMMUNAUTES APRES TRAITEMENT ===
45
46 # Detection des communautes par Louvain
47 communities_apres = community.louvain_communities(G_apres,
       resolution=0.7, seed=123)
48
49 # Classer les communautes par taille decroissante
50 partition_sorted = sorted(communities_apres, key=len,
       reverse=True)
51
52 # Attribuer l'indice de communaute a chaque noeud
53 for node in G_apres.nodes:
54     for i, comm in enumerate(partition_sorted):
55         if node in comm:
56             G_apres.nodes[node]["community"] = i + 1
57
58 # Couleurs pour les 5 plus grosses communautes, sinon gris
59 node_colors = []
60 for node in G_apres.nodes(data=True):
61     if node[1]["community"] == 1:
62         node_colors.append("xkcd:bordeaux")
63     elif node[1]["community"] == 2:
64         node_colors.append("xkcd:light indigo")
65     elif node[1]["community"] == 3:
66         node_colors.append("xkcd:soft pink")
67     elif node[1]["community"] == 4:
68         node_colors.append("xkcd:amber")
69     elif node[1]["community"] == 5:
70         node_colors.append("xkcd:dark lime green")
71     else:
72         node_colors.append("lightgrey")
73
74 # Position des noeuds
75 pos = nx.kamada_kawai_layout(G_apres)
76
77 # Taille des noeuds selon le degre (normalisee)
78 degrees = dict(G_apres.degree())
79 degrees = {n: (v - min(degrees.values())) / (max(degrees.
       values()) - min(degrees.values())) * 10000 for n, v in
       degrees.items()}
80
81 # Affichage du graphe
82 plt.figure(figsize=(70, 55))
83 nx.draw_networkx(G_apres, pos=pos, node_color=node_colors,
84                  node_size=[v for v in degrees.values()],
85                  edge_color='silver', width=3.0, with_labels
                       =False)
86
```

```
87 plt.axis('off')
88 plt.show()
```

Listing 17 – Coloration des communautes principales dans G_avant et G_apres