Université de Rouen Normandie - UFR Sciences et Techniques
Master 2 mention Bioinformatique – Parcours BIMS
2023 - 2024

Rapport de stage

Analyse textuelle des études scientifiques evaluant l'impact des vers de terre sur l'environnement

Présenté et soutenu par

Antoine Malet

Campus Agro Paris Saclay, Unité MIA Paris-Saclay
Equipe SOLsTIS

Encadrants :

David Makowski
Sophie Donnet

Université de Rouen Normandie - UFR Sciences et Techniques
Master 2 mention Bioinformatique – Parcours BIMS
2023 - 2024

Rapport de stage

Analyse textuelle des études scientifiques evaluant l'impact des vers de terre sur l'environnement

Présenté et soutenu par

Antoine Malet

Campus Agro Paris Saclay, Unité MIA Paris-Saclay
Equipe SOLsTIS

Encadrant :

David Makowski
Sophie Donnet

Remerciements

En premier lieu, j'aimerais remercier mes encadrants pour ce stage, Mme Sophie DONNET et M. David MAKOWSKI, pour avoir accepté ma candidature et acceuilli au sein de leur équipe. Je tiens aussi à adresser un mot particulier à mes vaillants collègues de bureau Emré ANAKOK et Caroline COGNOT, pour leur compagnie perpétuelle et leurs très bons conseils.

Je remercie aussi Louis LACOSTE, pour ses excellents conseils en R et en cinématographie, ainsi que François VICTOR, pour ses généreuses explications en statistiques théoriques auxquelles je n'ai pas compris grand-chose.

Table des matières

Table des figures

Liste des Abréviations

ASCII American Standard Code for Information Interchange
API Application Programming Interface
CSS Cascade Style Sheet
DOI Digital Object Identifier
HTML Hypertext Markup Language
IDE De l'anglais, Environnement de Développement Intégré
MA Métaanalyse
MIA Mathématiques et Informatique Appliquée
Rmd R Markdown
RG ResearchGate
RGS2 ResearchGateScrapper2.py
UMR Unité Mixte de Recherche

Glossaire

Bot : Un bot informatique est un agent logiciel automatique ou semi-automatique qui interagit avec des serveurs informatiques sans supervision humaine.

CSS : Langage de code utilisé pour mettre en forme une page web.

JSON : JSON (JavaScript Object Notation) est un format de fichier textuel conçu pour la structuration et l'échange de données (https ://www.hostinger.fr/tutoriels/quest-ce-que-json).

Markdown : Markdown est un langage de balisage léger qui permet de formater du texte de manière simple et rapide. Il utilise des caractères spéciaux pour indiquer les éléments de mise en forme, tels que les titres, les listes, les liens, etc. Les fichiers Markdown peuvent être convertis en HTML pour être affichés sur un site web ou dans un logiciel de traitement de texte (source : https ://bility.fr/definition-markdown/).

N-gram : Suite de mots consécutifs de taille n (une des possibilités de tokenisation). Utile pour comprendre les relations logiques entre les mots. Les bigrammes sont un cas particulier de n-gram (n-gram de longueur 2).

Racinisation (linguistique) : Obtention du radical, par exemple par dépréfixation ou désuffixation (*Exemple :*"enhance", "enhances" et "enhancement" deviennent tous "enhanc").

Réseau de n-grams : Figure permettant de visualiser toutes les relations entre les différents tokens simultanément, plutôt que deux par deux. Cela permet d'aller plus loin que l'analyse de bigrammes séparés les uns des autres.

Sélecteurs CSS : Les sélecteurs définissent les éléments sur lesquelles s'applique un ensemble de règles CSS. Ils peuvent être employés en Web scraping pour cibler et isoler certains éléments d'intérêt.

Token : Unité textuelle souvent réduite, voire ne comprenant qu'un seul mot, issue du processus de tokenisation.

Tokenisation : Processus consistant à découper un texte ou un corpus de textes en unités textuelles plus réduites, comme des mots, des n-grams ou des phrases.

Text-mining : Processus d'analyse textuelle consistant à transformer un texte non structuré en données structurées pour ensuite procéder à l'analyse. Cette pratique repose sur la technologie de « Natural Language Processing » (traitement du langage naturel), permettant aux machines de comprendre et de traiter le langage humain automatiquement (source : https ://datascientest.com/text-mining-definition).

Web scraping : Technique permettant d'extraire automatiquement de grandes quantités d'informations d'un site Web, sans intervention humaine directe, via un script informatique (source : https ://moncoachdata.com/blog/web-scraping-pratique/).

Chapitre 1

Introduction

1.1 Structure d'accueil

Mon stage s'est déroulé au sein de l'Unité MIA (mathématique et informatique appliqués), à l'INRAE du Campus Agro Paris Saclay, sur le plateau de Saclay. L'UMR MIA Paris-Saclay, associée aux tutelles AgroParisTech, INRAE et Université Paris Saclay, regroupe des statisticiens et des informaticiens spécialisés dans la modélisation et l'apprentissage statistique et informatique pour la biologie, l'écologie, l'environnement, l'agronomie et l'agro-alimentaire.

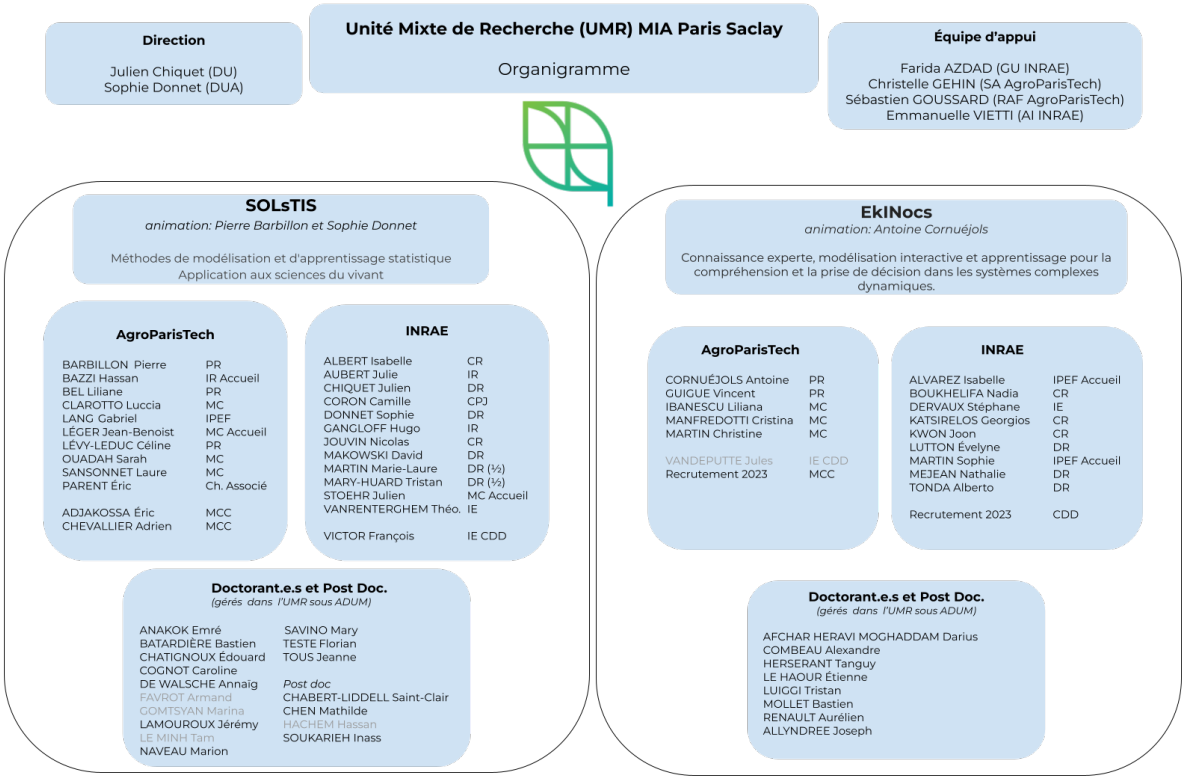

FIGURE 1.1 – Organigramme de l'UMR MIA Paris-Saclay

Les compétences mises en oeuvre portent sur les méthodes d'inférences statistiques (modèles complexes, modèles à variables latentes, inférence bayésienne, apprentissage, sélection de modèle...), et algorithmiques (généralisation, transfert de domaine, représentation des connaissances). L'unité développe des méthodes statistiques et informatiques originales, génériques ou motivées par des problèmes précis en science du vivant. Ses activités s'appuient sur une bonne culture dans les disciplines destinatrices : écologie, environnement, agro-alimentaire, biologie moléculaire et biologie des systèmes. L'unité MIA est dirigée par Julien CHIQUET et Sophie DONNET, et comprends deux équipes distinctes : l'équipe SOLsTIS (Statistical mOdeling and Learning for environnemenT and Life Science) dirigée par Sophie DONNET et Pierre BARBILLON, et l'équipe EkiNocs (Expert Knowledge, INteractive modelINg for understandINg and decisiOn makINg in dINamic Complexe Systems), dirigée par Antoine CORNUÉJOLS.

En tant que stagiaire, j'ai ainsi pu intégrer SOLsTIS pour mettre au point des méthodes informatiques et statistiques pour l'analyse textuelle d'abstracts d'articles scientifiques. L'unité comprends 63 membres tous statuts et équipes confondus, dont 40 appartiennent à l'équipe SOLsTIS, 19 à l'équipe EkiNocs et 4 membres d'appui.

1.2 Contexte Scientifique

Selon la perception des spécialistes du sujet en Europe, les services écosystémiques rendus par les vers de terre sont multiples :

- Ils permettent une meilleure aération des sols. (PREMIERE REF DU BIBTEX)
- Ils favorisent le recyclage des éléments nutritifs, du carbone, et du phosphore. (DEUXIEME REF DU BIBTEX)
- Ils jouent un rôle important dans le recyclage de la matière organique des sols (3e).
- Leur activité de décomposeurs (minéralisation de la matière organique) favorise la croissance d'autres organismes de l'environment, augmentant de ce fait la productivité des plantes (4e).
- Ils sont essentiels pour la structuration, l'entretien et la productivité des sols, forestiers, prairiaux et agricoles (5e).

Cependant, la perception des vers de terre en Amérique du Nord est très différente, pour ne pas dire opposée. On peut résumer la perception nord-américaine comme suis :

- Les vers de terre exotiques (venus d'Europe et d'Asie) sont des espèces invasives suceptibles de perturber les écosystèmes natifs (6e).
- Leur présence dans les sols modifie durablement les propriétés physico-chimiques des écosystèmes sousterrains (7e).
- Les modifications physico-chimiques induites par les espèces exotiques causent une baisse globale de la biodiversité chez les espèces natives du milieu (8e).
- L'impact de ces espèces exotiques sur les émissions de gaz à effet de serre est également sujet à controverse (9e).

Ce stage s'inscrit dans un projet de recherche visant à étudier la littérature scientifique, par des méthodes de web scraping (sous Python) et de Text Mining (sous R), afin d'analyser la perception du vers de terre dans les 116 abstracts présents dans la base de données et pouvoir proposer une solution argumentée à cette controverse.

1.3 Objectifs de mon travail

Ma mission reposant principalement sur de l'analyse textuelle, mon premier objectif a été de réunir, si possible automatiquement, l'ensemble des métadonnées disponibles à propos des articles mis à ma disposition pour réaliser l'analyse. Ces métadonnées comprennent par exemple l'abstract (le plus important pour le Text-mining), les auteurs, la date de publication, le DOI (*Digital Object Identifier*) ou encore le lien vers la page ResearchGate correspondante. Une fois cette base de données correctement établie et complétée (parfois manuellement), il a été possible de passer à l'étape d'analyse proprement dite, à travers l'élaboration d'un script de Text-mining.

Le script de Text-mining a été conçu dans le but de transformer les données lisibles par un humain dans le corpus de texte (mots, phases, paragraphes) en données statistiques analysables automatiquement par ordinateur. Pour cela, le corpus a donc été transformé en tibble (table de données R) contenant différents types de *tokens* (principalement des mots et des n-grams) aisément quantifiables, et donc, représentables sous formes de figures mathématiques. Ainsi, des informations comme la fréquence globale des mots, les mots les plus fréquents à l'intérieur de chaque métaanalyse, en encore la composition de chaque bigrammes (groupes de deux mots), entre autres, permettent une bonne compréhension générale du sujet, mais aussi de l'avis scientifique émis par chaque groupe d'auteurs au sujet des vers de terre. Par ailleurs, l'aspect quantitatif de l'analyse permet de s'affranchir en partie du ressenti subjectif qui vient inévitablement lors de la lecture et l'interprétation d'un texte. En automatisant totalement le calcul de ressenti, il devient alors possible de tirer des conclusions d'odre statistique, reposant uniquement sur des scores chiffrés calculés informatiquement.

Grâce à ces deux phases de construction d'une base de données et d'analyse textuelle, il devient donc possible de comprendre les idées générales présentes dans un corpus textuel important (ici, 116 abstracts) sans avoir besoin de les lires un par un pour en faire une synthèse collective.

Chapitre 2

Ressources : pratiques professionnelles, environnement informatique, outils informatiques et statistiques, données

2.1 Environnement informatique

Le matériel qui m'a été fourni par le laboratoire est un ordinateur fixe HP Elite SFF 800 G9 Desktop PC, numéro de série CZC2479ZXS - 4G087AV, avec 31 Go de mémoire vive et un processeur 12th Gen Intel® Core™ i7-12700 × 20, ainsi qu'un processeur graphique Mesa Intel® UHD Graphics 770.

Il est géré par un système d'exploitation Ubuntu 22.04 64-bits (distribution Linux), sous la version 42.9 de GNOME (GNU Network Object Model Environment) fournissant une interface utilisateur ergonomique pour interagir avec le système d'exploitation GNU (*GNU's Not Unix*). Il est ainsi composé d'un noyeau Linux et de GNU, qui ensemble forment le système d'exploitation communément connu sous le nom "Linux". Le système de fenêtrage est "Wayland", un système chargé de l'affichage et du placement des fenêtres durant l'usage du système d'exploitation par l'utiisateur.

2.2 Pratique Professionnelle

2.2.1 Veille bibliographique et technologique

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer REVENIR DESSUS APRES LA RÉDACTION DE LA PARTIE "Discussion".

2.2.2 Bonne pratique de programmation informatique et de developpement logiciel

Pour l'écriture du code Python, j'ai utilisé l'éditeur de code Visual Studio Code (VScode). Pour la conception du script Rmd, j'ai travaillé sous l'IDE RStudio. Conformément aux bonnes pratiques, l'intégralité des scripts (Python et R) ont été commentés en anglais, pour faciliter la relecture du code. Pour tester le fonctionnement de chaque script, deux approches différentes ont été mises en place :

1. Pour le script de Web-scraping en Python, les tests de fonctionnement effectués durant le développement ont été réalisées sur des jeux de données réduits (comprenant le plus souvent seulement les dix premiers articles) afin de pouvoir détecter rapidement si la sortie renvoyée est correcte par rapport à la tâche initiale. Pour la récupération de données via Crossref, des tests ont aussi été réalisées, notamment pour l'exploration des structures JSON renvoyées par la fonction works() de l'API. Une fois les données recherchées obtenues en sortie de tests ponctuels, il a été possible de généraliser la méthode, en l'appliquant dans l'ensemble du script principal. Enfin, afin d'éviter la régression du code (perte de fonctionnalité), j'ai souvent travaillé sur deux scripts identiques mais distincts, le premier servant de script principal, tandis que le deuxième était davantage un support pour le développement de nouvelles fonctionnalités. De cette façon, le script principal n'était mis à jour que lorsque le script secondaire était considéré comme fonctionnel. Un dépôt GitHub personnel aurait aussi pu jouer ce rôle, mais comme je travaillais seul sur cette partie, je n'en ai pas ressenti l'utlité.
2. Pour le script de Text-mining en Rmd sous RStudio, la plupart des tests de fonctionnement au cours du dévéloppement ont été réalisés dans la console R directement, afin d'éviter d'ajouter de nouveaux objets inutiles "test" à l'environnement R. Les processus n'ont été intégrés au script Rmd proprement dit qu'une fois leur fonctionnement testé et validé dans la console. Dans cette démarche, le fonctionnement de l'environnement R a été très utile, car cela a permis de réemployer certains objets stockés en mémoire sans avoir à les redéfinir seulement pour effectuer le test.

2.2.3 Communication des travaux

En concertation avec mes encadrants, j'ai aussi utilisé un dépôt GitHub spécialement mis en place pour le projet entre eux et moi, où mon travail de chaque jour a pu être sauvegardé grâce à un mécanisme de Push/Pull. De cette façon, mes encadrants ont pu facilement suivre l'évolution de mon travail et évaluer la qualité des solutions proposées.

Pour faciliter l'utilisation de cet outil, le logiciel GitHub Desktop m'a été présenté, une interface utilisateur graphique facilitant grandement la visualisation et l'usage du dépôt GitHub mis en place pour le projet.

Grâce à la fonctionnalité Knitr de Rmd, j'ai pu rendre compte de ma progression quotidienne en produisant automatiquement un fichier rapport au format HTML, directement issu de mon code (figures produites sous R, titres et intérprétation rédigées en Markdown ou HTML).

Les réunions avec mes encadrants, le plus souvent hebdomadaires, ont été fixées par échange de mails ou bien organisées de vive voix sur site. Elles m'ont permis de rester bien focalisé sur la mission en me fournissant des objectifs hebdomadaires clairs et précis, tout en permettant à mes encadrants d'être régulièrement informés de ma progression par rapport aux objectifs fixés.

2.3 Outils informatiques et statistiques pour les différentes phases de vos travaux

2.3.1 Récupération des abstracts et des métadonnées avec Python

Pour l'élaboration du script Python, j'ai surtout utilisé le package Python Habanero Crossref pour requêter (en utilisant une API) la base de données Crossref, qui contient une grande quantité de métadonnées (donnés sur les articles en tant que tel, comme l'abstract, les

auteurs, etc.), afin de récupérer les informations voulue à propos d'un ensemble de titres d'articles scientifiques défini au préalable sur le sujet des vers de terre. Pour compléter les données récupérées (la base de Crossref comprenant de nombreuses données manquantes), j'ai aussi développé en parallèle un module pour récupérer les données d'intérêt dans le code source de ResearchGate (web scraping), comme le *DOI (Digital Object Identifier)* de chaque publication, la date de chargement sur la base de RG ou encore le lien vers la page RG correspondante. Pour parvenir à cette solution, voici la liste des modules qui ont été utilisés :

Pandas : Pandas est un module qui permet de manipuler facilement des tableaux de données avec des étiquettes de variables (colonnes) et d'individus (lignes). Il est notamment utilisé dans le script pour exporter les résulats issus du code Python vers un fichier CSV(coma separated values), lisible et modifiable à l'aide d'outils de bureautique courants comme LibreOffice Calc ou Excel.

Numpy : Package conçu pour le calcul scientifique avec Python. Il est très utile pour l'algèbre (comme par exemple pour la manipulation de matrices), et son implémentation en C, C++ et Fortran en fait un outil de calcul rapide et efficace pour l'analyse de données et le calcul scientifique. Dans le code, cela dit, il sert simplement à l'indexage lors de la création du DataFrame de résultats.

Itertools : Module implémentant des outils Python pour maîtriser plus subtilement les itérations. La méthode employée dans le code est *zip_longest*, qui permet de créer un DataFrame à partir d'une liste de listes de tailles potentiellement différentes. La plus longue sera employée en référence (longest), et toutes les autres seront ajustées à cette longueur par l'ajout d'une *fillvalue* ("null", dans le code). Dans mon travail, elle a servi à créer le DataFrame requis à partir de listes de données de tailles pas forcément égales (à cause des valeurs manquantes).

Habanero : Module client de bas niveau pour interroger l'API Crossref, une base de données contenant les métadonnées des articles de tous les membres (des informations comme le titre, le nom d'auteur, le DOI etc.). Dans le code Python, elle est employée surtout pour rechercher les noms d'auteurs, les autres champs testés n'étant pas assez fiables pour automatiser complètement la récupération d'informations. Crossref est codé comme une **classe** du module Habanero, comprenant les méthodes works(), members(), prefixes(), funders(), journal(), type() et licence(). Dans le code Python, seule la méthode works() a été employée pour envoyer une requête à partir du titre de chaque article.

Unidecode : Module contenant entre autres la fonction éponyme *unidecode* (employée dans le script) conçue pour transformer les chaînes de caractères contenant des caractères non-ASCII (comme par exemple des idéogrammes chinois) pour les traduire en chaînes de caractères contenant uniquement des caractères ASCII. Dans le code, la fonction *unidecode* est employée pour rendre l'affichage des noms d'auteur contenant des caractères non-ASCII. Certaines corrections sont imparfaites et retirent quelques lettres.

RGS2 : Sous-module codé localement à partir d'un exemple trouvé en ligne [1] , par la suite adapté pour récupérer directement les informations voulues dans le code source du site scientifique ResearchGate, les autres options potentielles (comme Google Scholar) ayant souvent un système de détection et de blocage des bots. Seule la deuxième version du sous-module a été retenue dans le projet final. Il dépend des modules suivants :

1. Module **Parsel**, fonction *Selector* : Module facilitant l'extraction des données pour les formats HTML, JSON et XML. Dans le code, il est utilisé pour trouver les

1. citation url à mettre ici (scrape publications from RG)

données recherchées directement dans le code source de la page (web scraping) en s'appuyant sur des sélecteurs CSS.

2. Module **playwright.sync_api**, fonction *sync_playwright* : Module permettant de lancer une session navigateur depuis un script Python. Dans le code, il est utilisé pour se rendre sur le site de ResearchGate via une session Chromium, un navigateur libre développé par Google.

3. Module **re** : Module fournissant des opérations sur les expressions rationnelles utilisable dans un code Python, ce qui peut s'avérer nécessaire pour sélectionner très précisément les informations voulues dans une structure de données complexes. Dans le code Python, il est utilisé pour filtrer les résultats HTML bruts issus du Web scraping.

4. Module **time**, fonction *sleep* : Module fournissant différentes fonctions liées au temps. Dans le script, la fonction "sleep(t)" est utilisée pour forcer le système à ne rien faire pendant t secondes, évitant de cette façon de surcharcher le serveur cible de requêtes trop rapides et trop nombreuses.

2.3.2 Text Mining avec R

Pour réaliser le text-mining, j'ai utilisé un script R (développé sous Rstudio en Rmd) pour traiter le texte et l'analyser sous forme de figures. L'approche choisie est une approche "tidy" reposant sur la *tokenisation* du corpus de texte en unitées textuelles (appelées *"tokens"*) plus petite, comme des mots, des phrases ou des *n-grams*. J'ai pour cela pu m'inspirer du livre rédigé par Julia Slige (data scientist) et David Robinson (Directeur de Data Scientist de la plateforme Heap), disponible en ligne à l'adresse : https ://www.tidytextmining.com/. Mon travail a donc consisté à adapter les codes R montrés en exemple sur des livres à mes propres données (abstracts d'articles scientifiques), structurées différemment. Les analyses portaient par exemple sur la fréquence des mots, au global et pour chaque MA, ou encore sur l'analyse de sentiment (Pour répondre à la question : Quel est le sentiment global exprimé par le texte à la lecture ?), reposant sur l'attribution d'un score positif (+1) ou négatif (-1) à chaque mot du corpus. Cette attribution de sentiment a pu être réalisée grâce à un dictionnaire R conçu pour relier un token donné à la valence (positive ou négative) qui lui correspond. Afin de mieux visualiser les résultats, différentes figures ont été réalisées, comme des diagrammes en barre ou des nuages de mots.

Afin de filtrer les mots d'intérêt seulement, deux stratégies ont été employées : Premièrement, un filtrage brut de tous les mots de liaisons sans rapport direct avec le sujet (nommés "stop words" en text-mining, des mots tels que "the", "and", "is", "of" etc. en anglais) pour ne conserver que les mots sur lesquels les analyses pourront donner des résultats scientifiques significatifs (savoir que le mot "le" est le plus fréquent dans un corpus de texte français ne signifie rien sur le plan biologique). Deuxièmement, une autre approche a été de considérer la fréquence de chaque mot par rapport au nombre total de mots présents dans le corpus (n mot / N mots). De cette façon, les mots les plus fréquents perdent une partie de leur poids statistique, tandis que les mots plus rares en gagnent. On peut ainsi visualiser facilement les mots les plus importants d'un texte, sans même avoir besoin de modifier les données au préalable avec une liste de stop words. Cela permet de conserver le texte dans son ensemble, évitant ainsi un potentiel biais pouvant perturber l'analyse. Le script R développé repose sur les librairies suivantes :

Librairie dplyr : Librairie R conçue pour faciliter la manipulation de larges jeux de données (DataFrame et Tibble), avec des fonctions spacialisées comme *mutate* (ajout de variables), *select* (sélectionner les variables à partir de leurs noms), *filter* (filtrer des cellules selon leur valeur), *summarise* (résumer les informations d'un tibble dans un format très synthétique) et *arrange* (pour réordonner les lignes selon l'ordre / la variable voulue.)

Librairie ggplot2 : Librairie R pour créer déclarativement des graphiques divers et variés (barplots, histogrammes, scatter plots, etc.)

Librairie tidytext : Librairie R conçue pour faciliter l'analyse de texte (Silge, Julia, and David Robinson. 2016. "tidytext : Text Mining and Analysis Using Tidy Data Principles in R." [2]), se fondant sur le paradigme d'analyse de données "tidy", où chaque variable est une colonne, chaque observation une ligne et chaque ensemble d'observations est un tableau. La fonction la plus utilisée dans le cadre de cette analyse est *unnest_tokens*, qui permet de transformer un texte donné en une tables d'unités textuelles plus réduites (tokens), comme des phrases ou des mots.

Librairie Knitr : Librairie R conçue pour récupérer automatiquement l'output d'un code R (par exemple, pour produire une figure) afin de l'inclure dans un autre document (par exemple, format Word, HTML ou PDF) qui contiendra aussi la prose écrite par l'auteur du document, souvent pour interpréter ou commenter des résultats. Cette librairie permet notamment de moduler plus finement l'affichage des résultats, en permettant par exemple de ne pas afficher certaines figures dans un format de sortie donné (pour masquer une figure interactive au format HTML que l'on ne souhaite pas forcément voir apparaître dans un document PDF, par exemple).

Librairie SnowballC : Librairie R implémentant Snowball, un langage conçu pour gérer les chaînes de caractères, les nombres entiers et les booléens. Dans le script R de Text Mining, il a surtout servi à transformer les tokens pour ne garder que la racine de chaque mot (processus de *racination*), afin de ne compter qu'un seul exemplaire de chaque mot (si, pour un humain, "enhance" et "enhancement" sont deux mots ayant approximativement le même sens, informatiquement ce sont deux chaînes de caractères distinctes).

Librairie grid : Librarie R implémentant les fonctions graphiques primitives qui sous-tendent le package ggplot2. Elles permettent de modifier certains détails des graphes produits grâce à ggplot2, offrant ainsi un meilleur contrôle du rendu visuel du résulat obtenu. Dans le script, il est employé, par exemple, pour spécifier exactement l'aspect des flèches composant le réseau de bigrammes.

Librairie ggraph : Librairie R formant une extension de ggplot2, conçue pour permettre de supporter les structures de données relationnelles comme les réseaux, les graphes et les arbres. Dans le script, elle est notamment employée pour produire les réseaux de bigrammes.

Librairie igraph : fonction *graph_from_data_frame() :* Cette fonction appartient à la librairie igraph. Elle crée un objet graphe à partir d'un data frame, où le data frame représente les arêtes entre les nœuds.

Librairie egg : fonction *ggarrange() :* Librairie servant à organiser différents graphes sur une seule et même figure. Dans le script, c'est l'une des librairies employées pour comparer les quatre MA entre elles.

2. ref Silge/Robinson à mettre à la place

Librairie tidyr : Librairie R implémentant des méthodes utiles pour manipuler des objets de type "tidy". Elle comprend des fonction telles que *pivot_wider* et *pivot_longer* (pour convertir le dataFrame d'un format "wide" en format "long"), ou encore *bind_rows()*, pour combiner des DataFrames par lignes.

2.4 Données

2.4.1 Données 1

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac,

2.4.2 Données 2

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac,

2.4.3 Données 3

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac,

Chapitre 3

Résultats

Chapitre 4

Discussion

Au cours de ce stage, j'ai commencé par développer un programme Python pour le web scraping sur la base de données médicale PubMed, avant de découvrir que ce n'était pas selon dont nous avions besoin. J'ai donc réussi à m'adapter pour faire fonctionner mon code (en interrogeant une autre base de données), mais il reste encore à nettoyer les données extraites par le code (collectées directement en brut dans un fichier CSV généré par le script). Rétrospectivement, je ne suis pas certain que cette approche automatisée soit vraiment rentable en temps, même si je suis satisfait d'avoir commencé à apprendre comment développer ce type d'approches automatisées.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Chapitre 5

Conclusion

Elle vise, à reformuler les objectifs visés, énoncer les résultats essentiels obtenus, à replacer le travail dans son contexte scientifique et à faire ressortir leur importance théorique, pratique, tech- nique ou économique. Elle peut ouvrir de nouvelles perspectives ou hypothèses qui seront le point de départ de nouveaux travaux. Il n'y a pas a priori d'appel à des références.

[**?**]

Résumé

à faire à la fin

Mots-clés de référencement type MESH : Lumbricus terrestris, Soil, Ecosystem, Introduced species, Meta-Analysis.

Mots-clés des acquis techniques : Web-scraping, Text-mining, dplyr, R Markdown, GitHub.