III) Réalisation

1) Récupération des données

De nos jours, une façon possible d’automatiser la récupération de données est le web scraping. D’après Wikipédia, le web scraping est une technique d'extraction du contenu de sites web, via un script ou un programme, dans le but de le transformer pour permettre son utilisation dans un autre contexte. Grâce à cette technique, lorsque les pages ont une même structure, il suffit de disposer de tous les liens desquels nous voulons soutirer des données. Afin de produire notre projet final, basé sur la visualisation de données spatiales associées aux banques coopératives en France, nous avons créé deux jeux de données à l’aide de cette méthode.

Tout d’abord nous avons récupéré les données sur toutes les banques présentées auparavant. Cette base de données a été nommée **bdd\_coordonnees\_banques2022***.* Elle recense le nom de la banque, son type, son adresse ainsi que sa longitude et latitude. Ensuite, une deuxième récupération de données été nécessaire. En effet, les données socio-économiques pour chaque zone d'emploi de la France métropolitaine de 2020 ont été récupérées et stockées dans un fichier nommé **bdd\_social\_ze2020**. Toutes les variables obtenus sont présentés dans la partie *Présentation des données.*

Ce qui suit va vous présenter étapes par étapes notre démarche lors de la création de ses deux fichiers. Nous commencerons par expliquer le processus mise en place pour la récupération des coordonnées des banques pour ensuite vous expliquer les données socio-économiques.

Banques

La première tâche à effectuer lors de notre projet était l’une des plus importante : nous devions récupérer les données, sur lesquelles nous devions travailler. Cinq banques en France métropolitaine sont à récupérer ici depuis leur site officiel. Cependant pour la BNP Paribas, nous n'avons pas pu récupérer les données sur leur site donc nous avons dû utiliser un autre site où les données étaient disponibles. Vous trouverez ce lien en *Référence X.* Pour être plus précis, l’objectif premier était d’uniquement récupérer l’adresse de toutes les agences banque par banque.

Comme précisé plus haut, nous avons décidé de faire l’intégralité de notre projet en langage ***R***. Nous avons dû installer différentes librairies (nommé *package* dans ce langage, nous utiliserons cette appellation plus tard dans le rapport). La première librairie utile pour le web scraping est ***rvest***. Elle aide à l’extraction d’information sur des pages web et facilite l’expression des tâches courantes de web scraping. Une deuxième bibliothèque utile pour notre récupération de données est le package ***plyr***. Il regroupe un ensemble d’outils qui permettent de résoudre des problèmes tels que diviser une grosse structure de données en morceaux homogènes, appliquer une fonction à chaque morceau et enfin combiner tous les résultats. Troisièmement, nous avons utilisé la librairie ***dplyr***, elle fournit une grammaire de manipulation de données, fournissant un ensemble de verbes aidant à la résolution de manipulation de structure de données. Un autre package utile a été ***BanR***. Ce package utilise l'api BAN (Base Adresse Nationale). BAN est un jeu de données public des adresses françaises produit par OpenStreetMap, la Poste, l'IGN et Etalab. Ce package comporte plusieurs fonctions pour trouver une longitude et latitude à partir d'une adresse et inversement. Nous allons utiliser la fonction geocode qui permet de récupérer les couples longitude - latitude des adresses des différentes banques. Enfin, vous comprendrez un peu plus loin mais nous avons eu besoin d’un package de manipulation de chaine de caractères, nous nous sommes servis de ***stringr*** ainsi que ***stringi***.

A présent nous avons présentés presque tous les outils nécessaires pour pouvoir commencer le web scraping, un dernier et pas des moindres est l’extension de Google ***selectorGadge*t**. Il facilite la génération et la découverte de sélecteur CSS sur des sites complexes. En plus simple, il permet d’attribuer une espèce d’adresse à un élément remarquable d’un site internet (un titre ou un tableau). Lorsque l’on lance l’extension, une boîte s’ouvre en bas à droite de l’écran, il suffit de cliquer sur l’élément de la page que vous voulez récupérer. Si un sélecteur correspond, un cadre vert apparaitra et nous pouvons récupérer l’adresse de l’élément. Ci-dessous, vous trouverez un exemple pour sélectionner les adresses du Crédit Agricole :

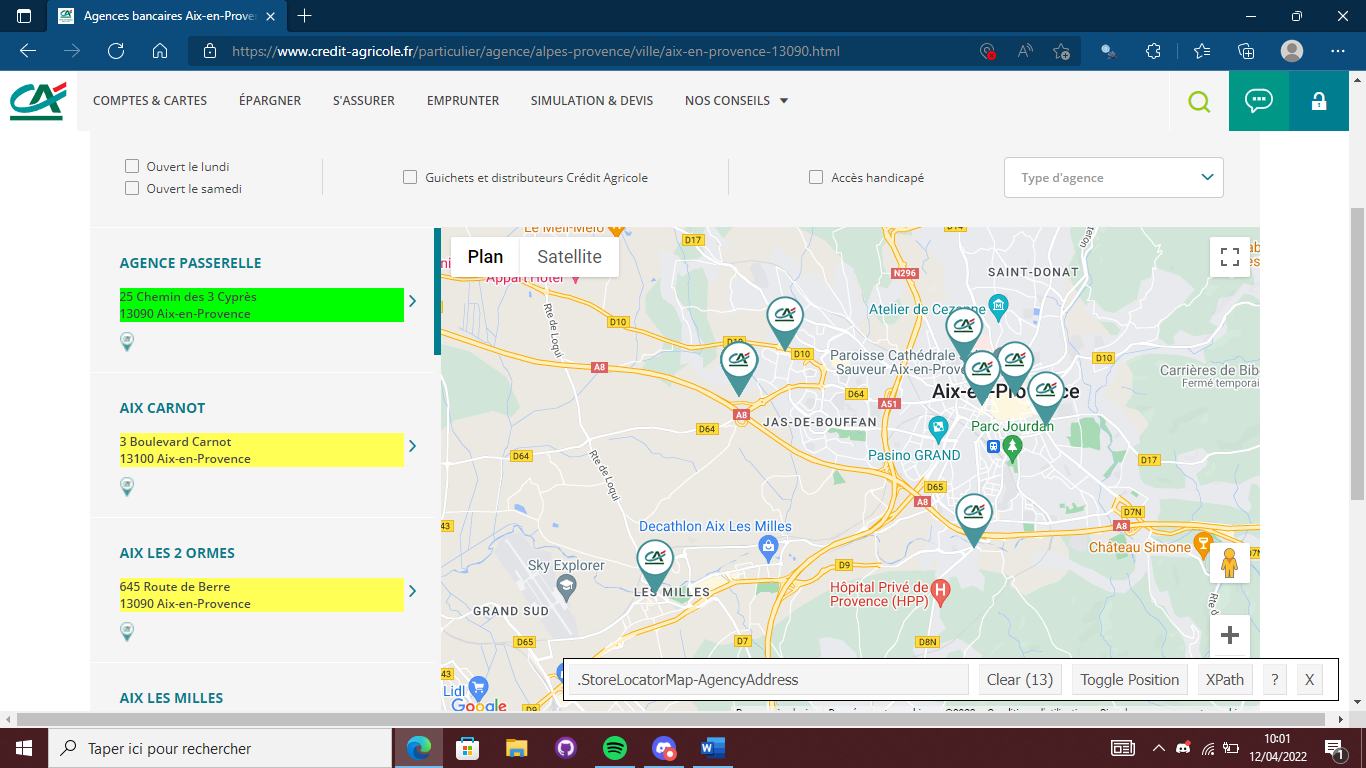


Figure X : Exemple de récuperation d’adresse pour le Crédit Agricole à l’aide de l’outil selectorGagnet

Sur l’exemple ci-dessus, vous pouvez voir que la première adresse (en vert) a été sectionnée, les adresses qui suivent (en jaune) sont mises en évidence car l’extension reconnaît que ce sélecteur est du même type que celui sélectionné en vert. Dans la boîte en bas à droite, on retrouve le nom de l’adresse sélectionné : « .StoreLocatorMap-AgencyAddress » et le nombre de sélecteur similaire disponible sur cette page (ici 13). Ainsi en utilisant ce nom nous allons pouvoir récupérer toutes les informations sélectionnées en vert et jaune, c’est-à-dire les 13 adresses.

Le site du Crédit Agricole présente les régions puis dans chaque région les villes par ordre alphabétique où contiennent au moins une agence, les villes avec des arrondissements sont détaillées (Paris 01 par exemple). L’url est sous la forme suivante :

**https://www.credit-agricole.fr/particulier/agence/**region**/ville/**ville**-**code postal**.html.** Nous avons dû utiliser un fichier annexe contenant les codes postaux associés aux villes de France. Sur le site certaines villes ont plusieurs codes postaux ce qui entrainent des doublons d’adresses.

Les sites de la Société Générale et de la Banque Populaire étaient faits de la même façon. Les urls étaient de la forme :

**https://agences.**nom de la banque**.fr/banque-assurance/agences-**departement**-**numero du département. Nous avons fait un web scraping pour récupérer la liste de chaque département ainsi que leur numéro. A partir de ce lien, nous pouvons récupérer toutes les adresses d’un département.

Le site du Crédit Mutuel est composé d’une liste de département puis des villes contenant une agence dans un département sélectionné. Dans une troisième page imbriquée nous avons une liste d’adresses d’agences de la ville choisie ainsi que d’autres aux alentours. Ça ne pose pas de problèmes en revanche cela entraine une quantité très importante de doublons d’adresses.

Le Crédit Mutuel de Bretagne est composé de plusieurs pages imbriquées aussi, les noms des départements puis le nom des villes avec leur code postal et enfin sur une troisième page l’adresse exacte de l’agence. Nous avons dû formater l’url pour accéder à la page des villes et à celle de l’adresse.

Pour la BNP Paribas, le site utilisé lui aussi divise les agences en département. Si le nombre d’agences est important, la page est divisée en plusieurs pages. L’url est sous la forme suivante :

**https://www.moneyvox.fr/pratique/agences/bnp-paribas/**numero du département**/** s’il y a plusieurs pages il faut rajouter un « numéro de la page**/** ».

Après avoir réalisé ce web scraping nous avons fait un premier tri des adresses car nous avions des doublons à cause du format du site ou alors que l’agence était partagée en différente parties (par exemple : l’agence à Montélimar 23 rue Raymond Daujat qui a une partie pour les entreprises et apparait donc deux fois dans les adresses récupérées).

A partir des adresses, nous avons utilisé le package *BanR* qui permet de convertir une adresse française en longitude et latitude. Pour chaque banque il a fallu modifier des adresses manuellement car elles ne donnaient pas de résultats avec la fonction du package. Nous réalisons un deuxième tri des adresses avec les couples longitudes, latitudes car certaines adresses changeaient seulement de code postal mais désignées la même agence. Les longitudes et latitudes étaient donc identiques. Nous obtenons après cela 14704 agences différentes.