Université de Toulouse

MASTER 2 GEOMATIQUE

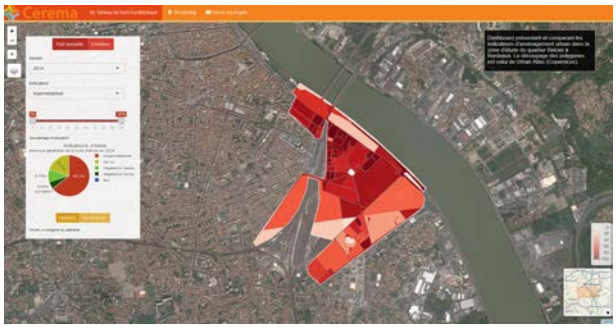
« **Sc**Iences **G**éomatiques en environne**M**ent et **A**ménagement » (**SIGMA**)

http://sigma.univ-toulouse.fr

RAPPORT DE STAGE

Valorisation de données issues d'imageries satellitaires

Développement d'outils de Dataviz



BRUNELLE Tom
CEREMA

Maître de stage : Dominique Hebrard / Amélie Lombard
Enseignant-référent : Laurent Jégou

Septembre 2020

.RÉSUMÉ

Le groupe Satellite, climat, gestion des systèmes d'information (SCGSI) de la délégation toulousaine du Cerema est spécialisée en application satellitaire pour l'observation urbaine et la prévention des risques. Leurs travaux impliquent une importante part de valorisation d'imageries et de résultats de télédétection auprès des services en charge de politiques publiques et des décideurs pour qui la donnée géographique n'est pas familière. La vulgarisation de la donnée et sa déclinaison sous forme d'indicateurs sont alors cruciales et les outils de la data visualisation (dataviz) offrent l'opportunité pour mieux communiquer, accompagner les divers projets et pour valoriser la donnée. C'est dans ce cadre que ce stage s'inscrit.

Un état de l'art des solutions existantes en dataviz a été réalisé pour répondre aux besoins formulés ainsi qu'au manque de connaissances du pôle satellite dans cette discipline. La solution RShiny a été retenue selon des critères et des nécessités internes. Il s'en est suivi une phase de développement d'applications pour valoriser, analyser et comparer les données sur plusieurs projets : suivi d'indicateurs d'aménagement urbain dans le quartier Saint-Jean Belcier à Bordeaux, comparaison et évolution de l'occupation du sol dans le département des Hautes-Pyrénées et analyse des phénomènes d'ilots de chaleur dans la métropole lilloise.

Ce travail a permis d'avoir un regard critique sur les outils de dataviz, sur leur intérêt et leur possible intégration au sein du pôle satellite. RShiny présente une solution solide, mais des limites émergent au regard du contexte interne. Il sera alors intéressant pour eux d'explorer ces solutions en fonction de leurs besoins et contraintes.

.ABSTRACT

The Satellite, Climate, Management and Information Systems (SCGSI) group of the Toulouse delegation of Cerema is specialized in satellite applications for urban observation and risk prevention. Their work involves a significant part of valorization of imagery and remote sensing results with the services in charge of public policies, sometimes with decision-makers for whom the geographic data is not familiar. The simplification of the data and its declination in the form of indicators are crucial and data visualization tools (dataviz) offer the opportunity to better communicate, support the various projects and to valorize the data. It is in this context that this internship fits.

A state of art of existing dataviz solutions was carried out to meet the expressed needs and to respond to the lack of knowledge of the satellite pole in this discipline. The RShiny solution was selected according to internal criteria and requirements. A development phase followed to valorize, analyze and compare data on several projects: monitoring of urban development indicators in the Saint-Jean Belcier district in Bordeaux, comparison and evolution of land use in the Hautes-Pyrénées department and analysis of local climate changes in the Lille metropolis.

This work enables to have a critical look at the dataviz tools, their interest and their possible integration or not within the satellite group. RShiny presents a solid solution, but limits emerge with regard to the internal context. It will be interesting for them to explore these solutions according to their needs and constraints.

REMERCIEMENTS

Je tiens à remercier Amélie Lombard et Dominique Hébrard de m'avoir donné l'opportunité et la chance de travailler avec eux sur la thématique de la dataviz au sein du pôle satellite du Cerema. Ils m'ont épaulé et suivi tout au long du stage malgré les conditions sanitaires exceptionnelles. C'était un réel plaisir de travailler avec eux.

Je remercie également mon enseignant référant Monsieur Laurent Jégou pour son suivi, son implication, ses conseils et sa disponibilité tout au long du stage.

Je tiens à remercier tous mes collègues du Cerema pour leur accueil plus particulièrement Edson Olivares Medina, stagiaire à l'ENSG, avec qui nous avons beaucoup partagé nos travaux et connaissances ainsi que Mathieu Rajerison, du Cerema Méditerranée, pour ses nombreux conseils sur RShiny.

Enfin, je remercie l'ensemble de la promo 2020 SIGMA pour l'année passée ainsi que toute l'équipe enseignante pour la qualité de la formation.

.SOMMAIRES

GLOSSAIRE ET SIGLES UTILES

API Application Programming Interface
Benchmark Analyse des produits et pratiques d'entreprises concurrentes
BI Business Analytics
CEREMA Centre d'étude et d'expertises sur les Risques, l'Environnement, la Mobilité et l'Aménagement
CLC Corine Land Cover
CGDD Commissariat général au développement durable
DALETT Délégation Aménagement Laboratoire Expertise Transports de Toulouse
DDT Direction Départementale des Territoires
DREAL Direction Régionale Environnement Aménagement Logement
DSI Direction des Systèmes d'Information
Dter SO Direction Territoriale Sud-Ouest
DVF Données de valeur foncière
EPA Établissement public à caractère administratif
ICU Ilot de chaleur urbain
IGN Institut Géographique National
LCZ Locate Climate Zone
MCTRCT ministre de la Cohésion des territoires et des Relations avec les collectivités territoriales
MTES Ministre de la transition écologique et solidaire
OCS Occupation du sol
OCSGE Occupation du sol à grande échelle de l'IGN
OGC Open Geospatial Consortium
OSO Occupation du sol du Cesbio
SCGSI Satellite, Climat et Gestion des Systèmes d'Information
SGBD Système de gestion de base de données
SIG Système d'information géographique
WFS Web Feature Service
WMS Web Map Service
WMTS Web Map Tile Service

.INTRODUCTION

La visualisation de données ou dataviz est un ensemble de méthodes interdisciplinaires qui a pour objectif de présenter des données sous une forme visuelle, néanmoins cette discipline ne dispose pas de définition bien établie. Celle que je tente de donner est la suivante : la dataviz est la représentation graphique et spatiale de données statistiques et géographiques dans le but de valoriser et de faciliter la compréhension de données parfois complexes. Il est aussi possible de parler de géo et data visualisation ce qui englobe l'aspect spatial dans les représentations. C'est une discipline qui existe sous différentes formes depuis longtemps, mais elle a pris beaucoup d'ampleur et s'est démocratisée sous l'ère du numérique et de l'open-data. En effet, il y a une plus grande disponibilité d'outils libres et gratuits et le support web permet de rendre les visualisations interactives et dynamiques.

Les objectifs de ces représentations visuelles peuvent être multiples : aide à la communication, à la décision, à l'analyse, à l'interprétation, à la valorisation, etc. La dataviz dépend alors du destinataire et des choix de formes, de sémiologie graphique et de données représentées que fera le développeur. Selon Sidonie Christophe, chercheuse au sein du Laboratoire en sciences et technologies de l'information géographique (LaSTIG), « l'enjeu de la géovisualisation, en tant que champ de recherche interdisciplinaire, est de concevoir des représentations graphiques et des moyens d'interaction avec les données géographiques, pour aider effectivement un utilisateur, à voir, percevoir, comprendre, interpréter, analyser voire prendre des décisions sur un phénomène spatialisé »[1].

Aujourd'hui, devant l'abondance de la donnée, de nombreuses solutions pour réaliser de la dataviz se sont développées. C'est sur cette discipline et sur des missions de valorisation de la donnée que le Centre d'étude et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement (Cerema) m'a engagé en tant que stagiaire.

Le groupe Satellite, Climat, Gestion des Système d'information (SCGSI), dans lequel j'ai évolué durant 6 mois, travaille sur de nombreuses thématiques concernant le littoral, les risques, l'occupation du sol ou encore le climat urbain. Les travaux de l'équipe impliquent une importante part de valorisation d'images satellites et de résultats de télédétection auprès des services en charge de politiques publiques parfois auprès de décideurs pour qui la donnée géographique n'est pas familière. La vulgarisation de la donnée a alors son intérêt. « L'essor de l'open data, le développement de nouveaux capteurs (satellites, aériens ou drones), la multiplication des plates-formes de diffusion offrent une facilité grandissante d'accès à la donnée d'imagerie. D'autre part, le développement des capacités de stockages de données et l'augmentation des puissances de calcul permettent de développer plus facilement les méthodes dites d'intelligence artificielle. Ces deux effets combinés participent à la démultiplication de l'information »[2]. C'est autour de ces enjeux de qualité et d'abondance de la donnée que la volonté d'utiliser des outils de dataviz pour analyser, comparer et valoriser les résultats issus d'imageries satellitaires s'inscrit. Au-delà des livraisons de cartographies ou de fichiers SIG, les outils de la géo et data visualisation offrent l'opportunité pour mieux communiquer et accompagner à l'usage de ces informations.

L'objectif du travail, fournit tout au long de ces 6 mois, est alors de rechercher et de tester des solutions de dataviz adaptées au contexte interne afin de valoriser les résultats de traitement spatial. Comme dit précédemment, les solutions dans cette discipline sont nombreuses et dans un

1 **Sidonie, Christophe.** La géovisualisation kezako ? . *LeMonde.* [En ligne] 18 11 2019. https://www.lemonde.fr/blog/binaire/2019/11/18/la-geovisualisation-kezako/
2 **Cerema.** [En ligne] https://www.cerema.fr/fr/centre-ressources/newsletters/signture/signture-69-artificialisation-sols-sa-mesure/evaluation-qualite-donnees-classification-occupation-du-sol

contexte national interne où aucun outil de dataviz n'est véritablement adopté par le Cerema, le travail préliminaire est primordial. Celui-ci consiste à recueillir les besoins internes de l'équipe et réaliser une bibliographie des outils existants. La définition de critères avec l'équipe de travail a permis de mener une étude préalable sélectionnant les solutions adaptées au cadre du Cerema. Il s'en est suivi une phase de prise en main des solutions retenues et de développement sur divers projets afin de valoriser et faciliter l'appropriation des résultats des travaux du pôle satellite. Cette discipline a fait l'objet d'un deuxième stage au sein du groupe SCGSI du Cerema Sud-ouest qui s'est focalisé sur l'exploration d'un *framework* (ensemble de composants) à savoir Python Dash. Mon travail s'est quant à lui concentré sur le développement de tableaux de bord en R avec la bibliothèque Shiny. Ces deux approches ont permis de mettre en parallèle deux outils de dataviz afin de les comparer, les analyser et de tirer des conclusions sur leurs performances et fonctionnalités. Ces tableaux de bord ont montré leurs intérêts dans la facilité et l'interactivité que l'utilisateur a pour manipuler, visualiser et mettre en valeur des données souvent complexes. Ces deux *packages* reflètent en réalité une divergence qui peut exister au sein du Cerema concernant les outils de dataviz, le stage a tenté d'amener de la clarté dans ce panel de solutions.

Ce rapport présente de manière chronologique les travaux effectués tout au long du stage ainsi qu'une analyse critique de ces technologies tout en prenant en compte le contexte interne du Cerema.

I. Contexte du stage

1. Contexte général

Le stage de fin d'étude du Master 2 SIGMA d'une durée de 6 mois s'est déroulé au sein du département Satellite, Climat et Gestion des Systèmes d'Information (SCGSI) de la Délégation Aménagement Laboratoire Expertise Transports de Toulouse (DALETT) du Cerema. Il a débuté le lundi 2 mars 2020 et s'est terminé le vendredi 28 août 2020.

2. Structure d'accueil

Le Cerema est un établissement public à caractère administratif placé sous la tutelle conjointe du Ministre de la transition écologique et solidaire (MTES) et du ministre de la Cohésion des territoires et des Relations avec les collectivités territoriales (MCTRCT). Les missions du Cerema portent sur de multiples thématiques de l'aménagement du territoire et du développement durable (urbanisme, environnement, infrastructures de transport, gestion des risques…). C'est un acteur majeur agissant au côté des collectivités territoriales et de l'État. Il est composé à travers la France de plusieurs départements territoriaux. La DALETT est la délégation toulousaine du Cerema Sud-Ouest dont la direction se situe à Bordeaux et est spécialisée sur de nombreux domaines comme notamment l'application satellitaire pour l'aménagement du territoire et la prévention des risques. Ces missions sont exercées par le pôle satellite ou SCGSI au sein duquel j'ai effectué mon stage.

L'équipe travaille sur de nombreux sujets en s'appuyant sur l'exploitation d'images satellitaires tels que l'évaluation de la qualité des données de classification de l'occupation du sol ou encore l'évolution du trait de côte. Ce groupe pratique la géomatique, c'est-à-dire, les activités de collecte, traitement et diffusion de l'information géographique.

3. Visualisation de données ou dataviz

La thématique principale du stage est la visualisation de données ou dataviz. Il est nécessaire d'éclaircir cette notion. La dataviz est la représentation de données parfois complexes sous la forme de graphiques, diagrammes, cartographies et autres infographies afin de les rendre plus accessibles et plus lisibles pour le public visé. Cela permet d'analyser et de faciliter la compréhension en rendant la donnée « plus visuelle ». Il existe de nombreuses formes de représentation de la donnée ainsi que de multiples outils pour la mettre en valeur. C'est une discipline qui est en plein essor depuis quelques années et qui ne cessent de se développer du fait de son utilité et de sa praticité dans une pluralité de domaines.

La représentation prend une forme différente selon la donnée et ce que l'on veut montrer. Cela peut être de :

→ l'évolution (graphique en courbes ou *lineplot* par exemple)
→ la comparaison (graphique en bâtons ou *barplot*)
→ la décomposition/analyse (graphique en secteurs ou *piechart*)[3]

Néanmoins, la DataViz ne s'arrête pas à ces 3 types de représentation. Elle peut intégrer des visualisations spatiales par le biais de cartes interactives ou encore d'infographies. L'idée de la visualisation des données est alors de déceler des tendances, des corrélations, des évolutions, des similitudes, des divergences, des aberrations, etc. Son intérêt prend forme dans sa capacité à permettre à l'utilisateur de comprendre simplement et rapidement des données parfois complexes et de retranscrire ces résultats avec un aspect visuel. En effet, comme il est mentionné dans le *Manuel de Data Visualisation*[4] de Jean-Marie Lagnel, « ce qui est invisible dans le tableau prend ainsi forme avec le graphique ».

C'est cette définition de la dataviz qui fait l'objet d'un stage au sein du pôle satellite du Cerema Sud-Ouest. En effet, il y a un intérêt à créer des outils pour permettre une valorisation de leurs projets et de leurs données. Il n'existe pas de solution généralisée au sein du Cerema concernant la *Data Visualisation* et c'est donc pour quoi, une phase importante de ce stage fut de réaliser un recueil du besoin et une bibliographie sur le sujet de la dataviz.

4. Besoin interne : intérêt de la DataViz pour le Cerema

L'objectif du stage est de mettre à disposition de l'équipe des outils de dataviz pour permettre la valorisation des résultats des travaux basés sur le traitement d'images satellites. L'idée est donc de concevoir un outil permettant de visualiser la donnée en la simplifiant et en la vulgarisant. Cela doit servir comme appui d'un raisonnement, base de communication et comme aide à la décision. Les thématiques au sein du Cerema sont larges et les nécessités différentes, il m'a fallu alors recueillir les besoins des agents de l'équipe.

Pour cela, j'ai réalisé des entretiens individuels avec quelques agents du Cerema Sud-Ouest. Grâce à une grille d'entretien (« voir annexe 1 »), j'ai pu éclaircir l'intérêt d'un outil ainsi que les fonctionnalités et spécificités auquel il doit répondre. L'enquête vise à recueillir et comprendre les besoins de l'équipe pour réaliser un outil qui dans l'idéal répondra au mieux à leurs attentes. Au regard des entretiens et de la bibliographie effectuée en parallèle, certains critères se détachent du fait de leur importance. J'ai tenté de les classer du plus au moins

[3] *Webinar #1 DataVIz & Best Practice.* **DataScientest**. 06/05/2020.
[4] **Lagnel, Jean-Marie.** *Manuel de Data Visualisation.* s.l. : Dunod, 2017.

important :

Format du livrable : le fichier de sortie peut être de plusieurs formats : PDF, lien HTML, application sur un serveur. Le choix se restreint lorsqu'il est question d'une production statique ou dynamique. En l'occurrence, les productions sont, pour la plupart des agents, imaginées de façon interactive (carte, graphique, *dashboard*) ce qui élimine certains formats. Il est aussi possible d'imaginer des productions dynamiques et interactives accompagnées de rapports figés présentant des compléments. La question du format est majeure, car les solutions pour créer des outils de dataviz ne possèdent pas toutes ces options concernant le format de l'*output*.

Représentation adoptée : la représentation souhaitée selon le projet est aussi un critère majeur à prendre en compte et cela recoupe avec le format du livrable. En effet, selon la visualisation imaginée, cela nous oriente vers des langages voire des bibliothèques spécifiques. Beaucoup de *packages* existent que cela soit en R, en python ou en JavaScript néanmoins ils n'offrent pas tous les mêmes fonctionnalités. La représentation porte aussi sur le sujet du projet : comparaison de sources, de millésimes, évolution, etc. Tous ces facteurs sont à bien prendre en compte dans le choix à faire.

Format en entrée des données : le format des données à exploiter est important, mais au regard des besoins et des entretiens, il est souvent vectoriel pour les résultats des études et en raster pour les données sources. Nombreuses sont les bibliothèques qui lisent les deux formats, même si cela est à nuancer entre simple visualisation d'un raster et manipulation des données de pixels. La seule difficulté peut être rencontrée sur la taille des fichiers en entrée. En effet, une image satellite à haute-résolution peut ralentir considérablement la fluidité de l'outil et augmenter le temps d'affichage.

Langage/outil choisi : les agents du groupe SCGSI sont familiers avec le langage informatique Python ce qui oriente le choix vers celui-ci même si d'autres langages sont pertinents tels que R ou le JavaScript. En effet, R offre une esthétique et une simplicité d'écriture assez intéressante et le JavaScript dispose de bibliothèques de visualisation très abouties et très variées, mais qui est complexe de par sa syntaxe. Néanmoins, les bibliothèques JavaScript sont utilisées et peuvent être pour certaines intégrées tant sous R que sous Python. La question du choix de l'outil lors des entretiens, n'était pas une des plus importantes, car ce sont les premiers critères évoqués qui décideront de celui-ci.

Communication : le destinataire de la production était selon moi un critère important, finalement cela reste le plus souvent une communication tout d'abord interne puis à destination des partenaires/commanditaires/financeurs et enfin publique. Le champ de communication du produit est donc large et assez similaire selon les agents du Cerema ce qui en fait un critère à ne pas négliger lors de la réalisation de l'outil mais pas majeur.

II. Contraintes et contexte techniques

La période pendant laquelle le stage s'est déroulé a été particulière. La crise sanitaire due au COVID-19 a imposé le travail à distance pendant toute la durée du confinement. Le pôle satellite du Cerema avait mis en place un groupe Slack afin que les employés puissent communiquer pendant le confinement. Les entretiens ainsi que les réunions hebdomadaires avec l'équipe ont été réalisés en visioconférence. La première phase du stage qui consistait à s'approprier le sujet et de dresser un panorama des outils a pu être réalisée sans de véritables difficultés. Les phases suivantes ont été cependant plus contraignantes. En effet, du fait de la non-présence, il était plus difficile de communiquer et de s'expliquer sur des problèmes rencontrés, de transférer et de télécharger des données, etc. Il a été néanmoins possible de réaliser toutes les missions sans être sur le lieu de travail, car je manipulais uniquement des logiciels libres et gratuits tels que Qgis ou RStudio.

Dans ce contexte particulier, un autre stagiaire est arrivé au Cerema courant mai pour m'épauler sur la question de la dataviz. La phase de développement et d'exploration des solutions a alors été coupée en deux : lui exploitant et maniant Python Dash et moi RShiny. Je devais moi-même explorer les deux outils initialement, mais cela m'a permis de me concentrer uniquement sur la dataviz sous R. Il faut aussi préciser que personne dans le service ne détient de compétences solides en R et en dataviz. Le travail de bibliographie a été par conséquent important et la documentation sur internet m'a beaucoup aidé dans le déroulement de ce stage. J'ai pu cependant bénéficier de l'aide et de l'expérience de Mathieu Rajerison de la DterMed (Cerema Méditerranée) sur les questions relatives à RShiny.

Les deux stages reflètent en réalité une incertitude interne au Cerema concernant la dataviz. En effet, il n'existe pas de décision claire concernant la solution à prioriser dans cette discipline. C'est pourquoi il est possible de visualiser plusieurs *dashboards* issus du Cerema utilisant des techniques différentes. Ces applications ont été développées avant le stage par des agents de différentes délégations territoriales du Cerema :

- Python Dash : Observatoire des marchés immobiliers[5]. Données de cadrage sur le marché de la maison à partir des données de valeur foncière (DVF).
- RShiny : Cartofriches[6]. Outil d'aide au recensement à l'échelle nationale des friches (industrielles, commerciales, d'habitat, tertiaires, etc.), ouvert au grand public via un portail de visualisation sur Internet.

Des contacts ont été pris avec certains agents qui ont travaillé sur ces projets afin d'avoir des informations sur les outils, leurs avantages et limites. Cela a permis de compléter le travail de bibliographie et de visualiser les codes sources pour avoir un aperçu du fonctionnement et de la manière de développer une telle application web.

Une note du Cerema fait ressortir le besoin d'un serveur de data visualisation pour s'adapter aux clients et à la nécessité d'avoir des solutions de livrable autres que les formats 'traditionnels' comme les images et les rapports figés. L'idée est que « les bénéficiaires puissent, sans compétences particulières en traitement de données et sans équipement particulier, mieux s'approprier nos travaux de valorisation de la donnée »[7]. Cette note prône l'utilisation de RShiny

5 Application « Observatoire des marchés immobiliers », http://doc-datafoncier.cerema.fr/dashboard_dvf/
6 Application « RShiny : Cartofriches », https://cartofriches.cerema.fr/cartofriches/
7 Frédéric Berlioz, Christophe Badol, Antoine Lemot, Nicolas Pelé (CE) Mathieu Rajerison (Med), Frédéric Lasseron, Perrine Rutkowski, Antoine Herman (HdF), Bertrand Leroux (O), « Note sur le besoin d'un serveur de Data Visualisation », Cerema, 03/04/2020 (Frédéric Berlioz, 03/04/2020)

et des différents modules R d'autant plus que c'est une solution adoptée par le Commissariat général au développement durable (CGDD).

III. Panoramas des solutions existantes

À l'heure de l'open data et de la profusion de données, les outils de dataviz sont très nombreux et en pleine expansion car utilisés dans de multiples domaines. Le support web rend beaucoup plus accessible l'exploration interactive des données.

Au regard de la large documentation disponible sur internet ou dans certains ouvrages concernant les outils de data visualisation, il a fallu réduire le champ de recherche. Au fur et à mesure de ce travail de bibliographie, il m'a fallu définir des critères et des caractéristiques pour mieux cerner les outils et les attentes. En effet, tous les moyens de réaliser de la visualisation de données ne sont pas adéquats par rapport au cadre du Cerema, aux attentes des agents et à d'autres contraintes telles que le format des fichiers en entrée et en sortie. Il est aussi important de choisir un outil par rapport à sa communauté. Cette dernière se doit d'être active afin de bénéficier d'appui en cas de difficulté et de s'assurer de sa longévité et pérennité.

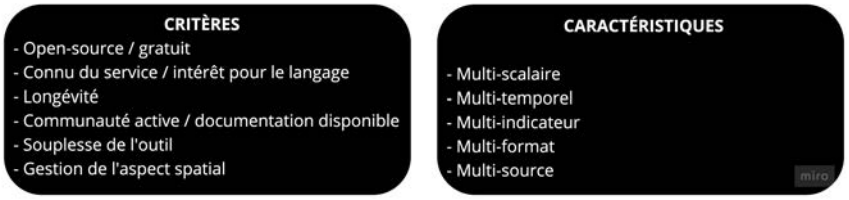


Figure 1: Critères et caractéristiques retenus

Voici un aperçu des outils existants, que j'ai pu voir au cours de ma bibliographie, qui semblent être aboutis pour faire de la dataviz. Je dis bien « semblent être », car il est difficile de juger et comparer des outils sans les manipuler au préalable.

- ArcGis

ArcGis dispose de solutions (payantes car nécessité d'avoir une licence) pour réaliser de la dataviz. Des exemples d'applications développées par Terranis sont proposées. Il y a :

StoryMap : La StoryMap permet de raconter et d'expliquer un sujet en utilisant de multiples supports tout en naviguant et déroulant la page internet. Il y a une interactivité entre le *scroll* de l'utilisateur et l'affichage[8].

*Operations Dashboard :*ArcGis *Dashboard* est un outil permettant de créer des tableaux de bord interactifs mêlant graphiques, cartes et textes. L'outil est configurable et permet entre autres de visualiser et analyser des données en temps réel. L'intérêt est qu'il n'y a aucunement besoin de coder, tout est entièrement paramétrable[9].

- Tableau

[8] Terranis, « Conception d'un observatoire de la végétation urbaine », [En ligne] https://myterranis.maps.arcgis.com/apps/MapSeries/index.html?appid=36e83f66e1204aa0aa9dd851769fbcf3

[9] Terranis, « Synthèse des indicateurs "Biodiversité" liés à la végétation urbaine », [En ligne] https://myterranis.maps.arcgis.com/apps/MapSeries/index.html?appid=36e83f66e1204aa0aa9dd851769fbcf3

Tableau est un logiciel de visualisation de données aussi nommé BI (*Business Intelligence*). C'est un logiciel payant qui permet de concevoir des *dashboards* rapidement et de manière très simplifiée. En effet, tout comme son concurrent Power BI, il fonctionne sur des « glisser-déposer » et ne nécessite donc pas de compétences spéciales.

- R

R est un langage informatique avec une syntaxe relativement simple et intuitive pour des habitués de la programmation. Il dispose d'un *framework* RShiny qui permet de créer des *dashboards* en incorporant des bibliothèques JavaScript tels que Leaflet, plotly, ggplot2, amCharts et autres. Il est facile à prendre en main, assez souple dans les choix de représentation et relativement simple dans sa structure. C'est une solution moderne et fluide pour créer des interfaces graphiques. La communauté est large et active et il n'y a pas de connaissances en CSS, HTML ou JavaScript nécessaires pour réaliser une application Shiny. L'utilisation de R ou du *package* Shiny est gratuite et libre mais la problématique émerge sur la question de l'hébergement de l'application web qui requiert un serveur où R est installé avec l'ensemble des bibliothèques nécessaires. Les applications développées en local sont visualisables uniquement dans RStudio (environnement de développement de R). Enfin, c'est un langage qui ne jouit pas d'une très forte popularité au sein du Cerema et qui est méconnu du groupe SCGSI.

- Python

Python est un langage en pleine expansion depuis quelques années et s'impose comme un des plus populaires. Dans le domaine de la dataviz, il est moins utilisé que d'autres langages mais du fait de sa grande communauté très active, des bibliothèques se développent rapidement. Il possède de nombreuses bibliothèques pour de la représentation graphique telles que : matplotlib, ggplot, plotly, bokeh[10] ou encore de la représentation spatiale : geoplotlib, leaflet, folium. De manière similaire à R, le *framework* Dash permet de créer un *dashboard* interactif tout en incorporant ces différentes bibliothèques. Comme pour les applications Shiny, la difficulté apparaît une nouvelle fois lorsque l'on veut déployer l'application car il est nécessaire d'avoir un serveur pour l'héberger avec python et les *packages* nécessaires installés dessus. Une autre limite apparaît concernant le *package* Dash ou encore Leaflet, qui sont des bibliothèques relativement récentes ne possédant pas encore toutes les fonctionnalités développées sur RShiny.

- JavaScript

JavaScript est un langage très populaire possédant de nombreuses bibliothèques notamment dans le domaine de la visualisation interactive de données. De multiples bibliothèques permettent de créer des *dashboards* et des cartographies/graphiques interactifs. La force de ce langage est qu'il est facile de produire en *output* un lien HTML ouvrable localement. La bibliothèque la plus complète et la plus développée est D3.js. Elle permet une grande souplesse, une interactivité et des créations très esthétiques : comme il est possible de voir dans le tutoriel du module U351_33 du M2 Sigma[11]. Elle est intégrée dans une page web HTML et permet de créer des objets SVG (*Scalable Vector Graphics,* format pour des graphiques vectoriels) dynamiques

[10] Issarane, Hausmane. « Visualisation des données via Python : Les packages à connaître ». *Le DataScientist.* [En ligne] 2019. https://le-datascientist.fr/visualisation-des-donnees-python.

[11] Jégou, Laurent. « Tutoriel sur la représentation graphique statistique avec D3 ». [En ligne] https://ljegou.github.io/d3sigma/.

tout en y liant du CSS[12]. C'est en l'occurrence le langage de base des animations des pages web côté client. Il est intégré grâce à des bibliothèques dans des langages de plus haut niveau comme R et Python dès qu'il faut produire des supports web dynamiques. Néanmoins, le JavaScript est un langage difficile à prendre en main, la syntaxe est longue et comme pour R, ce n'est pas un langage informatique connu du service. Les bibliothèques JavaScript sont utilisées dans le développement d'applications web, car elles sont implémentées dans d'autres langages mais, utilisé seul le JavaScript est moins pratique.

Dans le contexte du Cerema, les outils s'orientent selon l'équipe et moi vers des langages de programmation : en Python, JavaScript ou encore R. L'avantage de ces langages informatiques, c'est qu'il est facile de moduler l'outil en fonction des besoins et des attentes concernant le rendu final : grande souplesse dans la confection du livrable. Chacun de ces langages possède de nombreuses bibliothèques, une grande communauté et beaucoup de documentations et sont pour la plupart toujours en expansion et en développement. Il a néanmoins été important de prendre en considération tous les champs du possible avec notamment des logiciels nécessitant une licence tels que ArcGis ou Tableau.

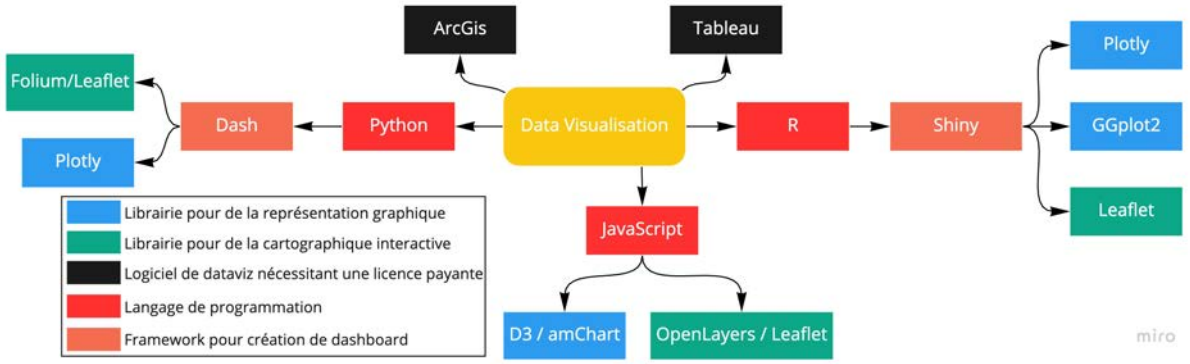


Figure 2: Carte mentale des outils de DataViz

Au regard de la pluralité d'outils, de langages et de l'évolution constante et rapide du monde de la DataViz, le choix est difficile à faire. En effet, ils trouvent tous leur intérêt et leur spécificité quelque part. Par rapport aux critères et caractéristiques énoncés précédemment ainsi qu'au cadre du Cerema, il est possible d'affiner le choix. Ce dernier s'oriente plutôt vers des solutions agiles, souples et esthétiques soutenues par une communauté large et active. L'outil doit aussi bénéficier d'une grande documentation accessible en ligne gratuitement avec un potentiel de visualisation interactive mêlant graphiques, cartes, textes et permettant donc de lire de nombreux formats de données (raster, vectoriel, tableur, etc.). Les critères sont nombreux, mais les solutions restent toujours multiples.

Pour conclure cette bibliographie, la solution Dash semble être une solution intéressante et adaptée en dataviz et cela constitue un outil adéquat au cadre du Cerema. RShiny, quant à lui, présente aussi une solution sérieuse en dataviz car c'est un *package* beaucoup plus avancé et développé en termes de fonctionnalités mais malheureusement il est inconnu pour le service. Enfin, le JavaScript a été plus ou moins écarté par l'équipe du fait de sa complexité. Les autres outils mentionnés dans cette synthèse sont très intéressants mais ne sont pas gratuits. Il est

[12] Antoine Moriceau, Julie Laforêt et Corentin Rey. « Un exemple de géovisualisation de phénomène temporel avec la bibliothèque D3.js ». *Mappemonde.* juillet 2016, Vol. Numéro 118.

nécessaire d'avoir une licence bien que Tableau soit un moyen simple et efficace de mettre en forme la donnée sans savoir coder et qu'ArcGis permette de belles visualisations par des *dashboards* ou des *storymaps*.

Pour compléter ce travail de bibliographie et de recueil des besoins, je me suis appuyé sur un concours de visualisation de données organisé par Toulouse DataViz en mars 2020. En effet, j'ai repris un graphique présentant les logiciels/langages les plus utilisés par les participants de l'année passée (« voir figure 3 »). Il est possible de remarquer que celles qui apparaissent en premières positions sont celles qui ressortent dans la bibliographie.

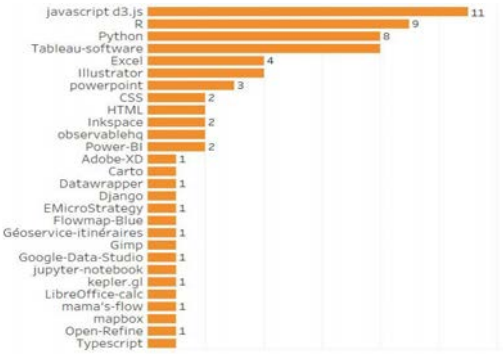


Figure 3: Classement des outils de DataViz dans le cadre du Hackaviz

Source : http://toulouse-dataviz.fr

IV. Présentation des projets et de la commande

Cette partie a pour but de présenter les projets sur lesquels les applications de dataviz ont été développées ainsi que les données utilisées, les fonctionnalités attendues et le *dashboard* final créé.

1. Application « Occupation du sol »

Première application développée sur les données d'OCS face aux usages au niveau communal dans le département des Hautes-Pyrénées.

1.a) Le projet

Le projet porte sur la réalisation d'un panorama des données de consommation d'espace à l'échelle du département des Hautes-Pyrénées : évaluation comparative des indicateurs par rapport à plusieurs sources de données et évolution temporelle de ces indicateurs. L'objectif étant de réaliser un panorama et une analyse de ces données au regard des besoins métiers des services de l'État.

Le comparatif des données d'occupation du sol se fait entre plusieurs sources :

- Corine land cover (CLC) du service Copernicus
- OSO Théia
- Occupation du sol à grande échelle (OCSGE) de l'IGN

- Fichiers Fonciers diffusés par le Cerema

Les diverses classes des différentes sources d'OCS ont été regroupées par le pôle satellite du Cerema avant mon arrivée et elles ont été converties en 4 indicateurs conformément à des travaux sur les indicateurs par un groupe de travail régional en Occitanie :

- Urbain
- Artificiel
- Agricole
- Naturel

L'objectif est alors de créer un outil de visualisation en prenant en compte différents critères :

- Multi-indicateurs (urbain, artificiel, etc.)
- Multi-scalaires (département, communes)
- Multi-temporel (2014, 2016, etc.)
- Multi-source (OSO, CLC, etc.)
- Multi-format (raster, vecteur)

1.b) Les données

Le jeu de données comprend plusieurs indicateurs d'OCS tel que : urbain, artificialisé hors urbain, agricole et naturel. Le but du *dashboard* est alors de comparer les valeurs des indicateurs des différentes sources d'OCS mais aussi de comparer chaque source par rapport à plusieurs millésimes. C'est un tableau de bord qui mêle comparaison, évolution et analyse spatiale.

Les données comprennent :

- Un fichier *shape* des communes des Hautes-Pyrénées : 469 lignes représentants les communes des Hautes-Pyrénées et 86 colonnes : pour chaque source, un millésime minimum et pour chaque millésime 4 indicateurs.

nom	oso09_a1	oso09_a2	oso09_a3	oso09_a4	oso09_a0	oso10_a1	oso10_a2	oso10_a3	oso10_a4	oso10_a0	oso11_a1	oso11_a2	oso11_a
Ancizan	1.36	0	4.20	94.44	0.00	0.91	0	3.61	95.48	0.00	0.60	0	2.03
Tilhouse	4.71	0	28.14	67.15	0.00	4.03	0	32.30	63.67	0.00	4.67	0	32.19
Cauterets	1.35	0	1.64	96.99	0.02	0.34	0	0.86	98.78	0.02	0.52	0	0.69
Arné	10.03	0	64.05	24.92	1.00	12.11	0	67.49	19.39	1.00	11.46	0	67.04
Ozon	7.07	0	42.79	50.15	0.00	7.35	0	44.21	48.44	0.00	7.58	0	42.68
Bagnères-de-Bigorre	4.37	0	7.42	88.22	0.00	3.61	0	6.37	90.03	0.00	3.41	0	6.34
Capvern	9.50	0	34.48	56.02	0.00	9.08	0	36.92	54.00	0.00	11.58	0	37.48
Sarrancolin	1.55	0	4.21	94.24	0.00	1.76	0	3.49	94.75	0.00	1.76	0	4.45
Lugagnan	30.52	0	12.32	57.16	0.00	24.30	0	18.66	57.04	0.00	29.81	0	17.49
Libaros	10.36	0	65.05	24.59	0.00	12.63	0	67.26	20.11	0.00	5.43	0	69.52
Saint-Sever-de-Rustan	13.62	0	52.79	33.53	0.06	5.58	0	65.51	28.84	0.06	5.39	0	63.69
Pouyastruc	10.81	0	58.95	30.24	0.00	10.82	0	60.59	28.59	0.00	8.25	0	60.84
Lourdes	19.05	0	14.28	66.67	0.00	19.09	0	15.98	64.93	0.00	20.98	0	14.87

Figure 4: Données OCS des communes de Hautes-Pyrénées

Correspondance des indicateurs et des données :

a1 → pourcentage de l'indicateur urbain dans la commune
a2 → pourcentage de l'indicateur artificialisé hors urbanisé dans la commune
a3 → pourcentage de l'indicateur agricole dans la commune
a4 → pourcentage de l'indicateur naturel dans la commune

- L'OCSGE de 2013 sur le département des Hautes-Pyrénées au format vectoriel

- L'OSO au format raster pour les millésimes suivants : 2009, 2010, 2011, 2014, 2016, 2017, 2018
- CLC au format vectoriel pour les millésimes suivants : 1990, 2000, 2006, 2012, 2018

1.c) Public visé

C'est un travail qui vise tout d'abord l'équipe de production de ces indicateurs à savoir le pôle satellite du Cerema puis le commanditaire, la Direction Régionale Environnement Aménagement Logement (DREAL) Occitanie et la Direction Départementale des Territoires (DDT) Hautes-Pyrénées avec une communication auprès du groupe de travail régional en Occitanie.

1.d) Fonctionnalités attendues

Le tableau de bord souhaité doit mêler cartes interactives des communes, graphiques d'évolution de l'occupation selon chaque millésime et indicateur, représentations de statistiques tant communales que départementales et affichage des couches spatiales d'OCS (raster et vecteur) dans une carte dynamique. Pour que l'application soit agréable et intuitive, elle se doit d'incorporer de l'interactivité entre les différentes visualisations et laisser à l'utilisateur le choix de jouer sur l'affichage par le biais de sélecteurs par exemple.

L'application finale développée n'a pas réussi à intégrer toutes les fonctionnalités souhaitées, mais elle permet de répondre en grande partie à la commande et aux attentes en analysant et comparant les évolutions d'occupation du sol entre source et millésime. Voici un aperçu de ce *dashboard* :

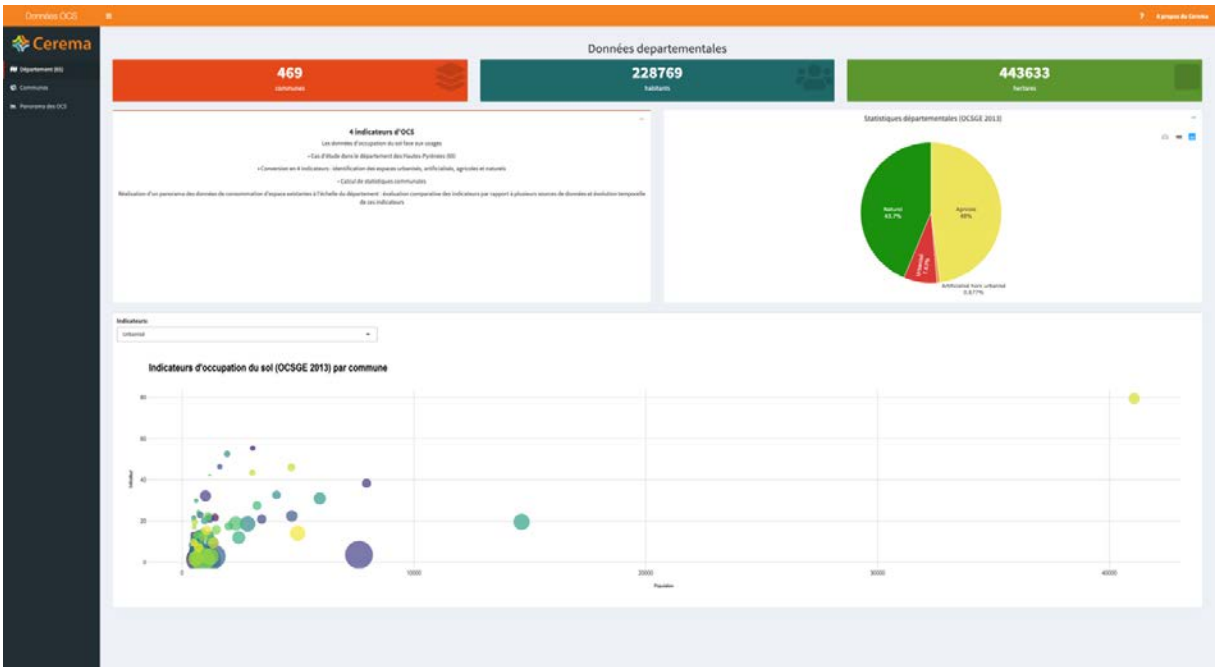


Figure 5 : Application OCS - Onglet Département

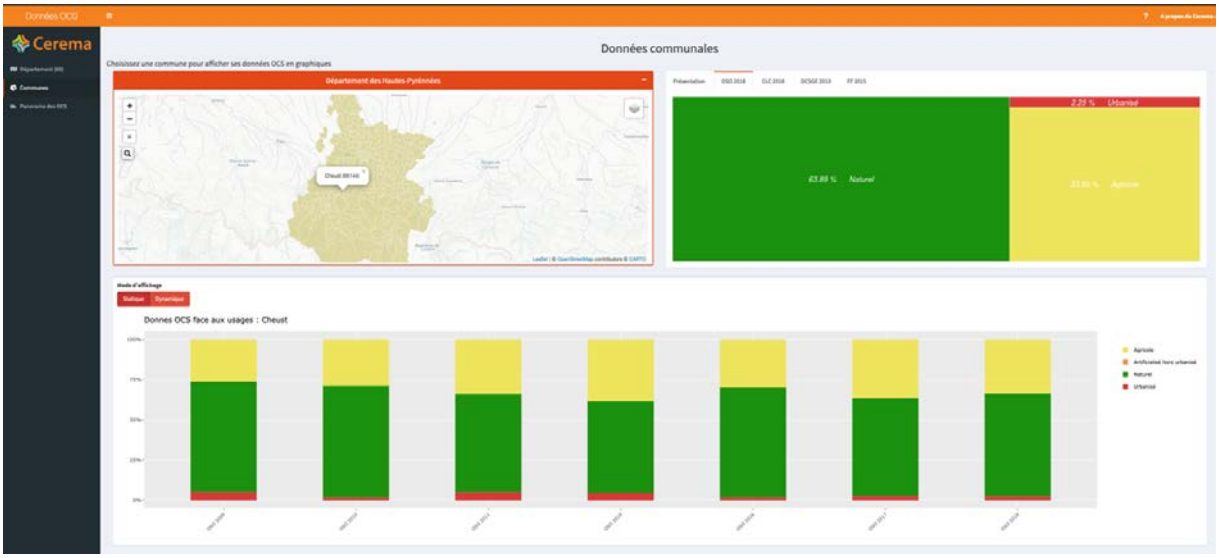


Figure 6 : Application OCS - Onglet Communes

Figure 7 : Application OCS - Onglet Panorama des OCS

L'application développée permet à l'utilisateur de visualiser et prendre connaissance des différences d'évolution et d'occupation qu'il peut exister sur les mesures de ces indicateurs notamment de l'artificialisation via ces bases de données. Cela fait ressortir qu'aucune de ces sources ne mesure les mêmes valeurs à une date donnée ou les mêmes dynamiques d'évolution entre plusieurs millésimes avec même des tendances parfois contraires. « Il n'y a pas de consensus sur la quantification absolue de ce phénomène entre les différentes sources de données disponibles. Ces démarches mettent en lumière le manque d'information sur la qualité d'usage des bases de données. »[13]

Concernant le développement de l'application, l'ensemble des aspects techniques et les limites rencontrées pendant cette phase de production, cela fait l'objet de la quatrième partie de ce rapport.

2. Application Euratlantique

La deuxième application développée porte sur l'évolution des indicateurs d'aménagement urbain dans le cadre du projet Euratlantique à Bordeaux.

[13] Cerema, https://www.cerema.fr/fr/centre-ressources/newsletters/signture/signture-69-artificialisation-sols-sa-mesure/evaluation-qualite-donnees-classification-occupation-du-sol

2.a) *Le projet*

Bordeaux-Euratlantique est une opération d'aménagement sur les communes de Bordeaux, Bègles et Floirac. Avec une programmation de 2 500 000m² de logements, bureaux et équipements publics, le projet prévoit 40 000 nouveaux habitants et 30 000 nouveaux emplois sur ce territoire d'ici fin 2030[14]. L'établissement public à caractère administratif (EPA) Bordeaux Euratlantique est intéressé par la mesure et l'évaluation de son opération à la fois de manière globale mais aussi sur certaines thématiques précises telles que l'imperméabilisation des sols. C'est dans ce contexte que l'EPA a besoin de travailler sur les modalités d'évaluation du projet et a fait un appel d'offre auquel le Cerema a répondu.

La prestation consiste à produire les indicateurs de suivi d'aménagement sur l'ensemble du périmètre d'aménagement de l'EPA Bordeaux Euratlantique, à partir de données satellitaires haute-résolutions du satellite Pléiades :

- Végétation (ainsi qu'une distinction de la végétation haute et basse)
- Surface perméable (correspondant au sol nu, hors végétation)
- Surface imperméable (comprenant le bâti et les surfaces anthropisées imperméables)

2.b) *Les données*

Le jeu de données pour réaliser le *dashboard* comprend :

- Un fichier *shape* représentant les polygones selon le découpage d'Urban Atlas (Copernicus). 86 polygones réparties sur le quartier de Saint-Jean Belcier et 67 colonnes représentant les pourcentages ou surfaces d'occupation des différents indicateurs sur plusieurs millésimes : tous les 2 ans de 2012 à 2020.

	LCZ_2014	LCZ_2016	Difference	id	majority	minority	X2014_nd	S_2014_nd	X2014_imp	S_2014_imp	X2014_sn	S_2014_sn	X2014_vgb	S_2014_vgb	X2014_vgh
1	G	G	OK	NA	2016_eau	2016_eau	0	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	E	E	OK	NA	2016_imp	2016_sn	0	0	68.86	42157.75	0.78	479.00	22.75	13927.25	7.61
3	E	E	OK	NA	2016_imp	2016_vgh	0	0	84.11	1881.25	3.79	84.75	11.89	266.00	0.21
4	9	9	OK	NA	2016_vgh	2016_sn	0	0	18.65	1499.50	8.63	693.75	13.62	1094.75	47.55
5	G	G	OK	NA	2016_vgb	2016_imp	0	0	4.66	86.75	32.55	605.50	39.95	743.25	22.83
6	9	9	OK	NA	2016_imp	2016_vgh	0	0	79.59	997.25	6.76	84.75	10.69	134.00	2.95
7	G	G	OK	NA	2016_vgb	2016_eau	0	0	11.83	1900.00	30.77	4941.75	56.00	8992.75	0.69
8	2	2	OK	NA	2016_imp	2016_vgb	0	0	87.94	23951.75	2.25	613.00	3.23	881.00	6.57
9	2	2	OK	NA	2016_imp	2016_vgb	0	0	84.78	13981.75	4.43	730.50	2.99	493.50	7.80
10	E	E	OK	NA	2016_imp	2016_sn	0	0	78.20	3114.75	0.27	10.75	21.11	841.00	0.41
11	7	7	OK	NA	2016_imp	2016_imp	0	0	100.00	247.50	0.00	0.00	0.00	0.00	0.00
12	E	E	OK	NA	2016_imp	2016_vgh	0	0	67.88	5248.25	10.73	829.50	19.45	1503.75	1.94
13	2	2	OK	NA	2016_imp	2016_vgh	0	0	98.13	9557.75	0.00	0.00	0.00	0.00	1.87

Figure 8: Données des indicateurs d'aménagement urbain, quartier Saint-Jean Belcier, Bordeaux

- Un fichier *shape* représentant la délimitation de la zone d'étude.
- Un fichier raster représentant l'occupation du sol en 2014
- Un fichier raster représentant l'occupation du sol en 2016

14 Bordeaux-Euratlantique, https://www.bordeaux-euratlantique.fr/

J'ai pour compléter le *dashboard* récupéré les coordonnées GPS de chaque projet d'Euratlantique dans le quartier Saint-Jean Belcier ainsi que des compléments d'information : nom et adresse du projet pour en créer un fichier *shape* de ponctuels.

2.c) Public visé

L'EPA Bordeaux Euratlantique est le principal destinataire de la production car c'est le commanditaire.

2.d) Fonctionnalités attendues

Concernant ce *dashboard*, les aspects spatial et temporel sont très importants. Il doit intégrer des fonctionnalités permettant de comparer de manière fluide et intuitive les millésimes entre eux ainsi que de jouer sur l'affichage en fonction du pourcentage d'occupation d'un indicateur sur une zone. Il s'agit d'un tableau de bord qui permet de suivre l'évolution d'indicateurs d'aménagement urbain. Il faut alors arriver à montrer visuellement l'évolution et les changements de manière simple et didactique.

Pour ce tableau de bord, nous avons alors tenté d'y intégrer des curseurs qui permettent de jouer sur l'affichage cartographique, d'utiliser des flux WMS et de développer une page de *storytelling*. Cette dernière est une page que l'on déroule où des informations et des visualisations souvent chronologiques apparaissent au fur et à mesure que la page défile. Voici des captures d'écran de l'application finale :

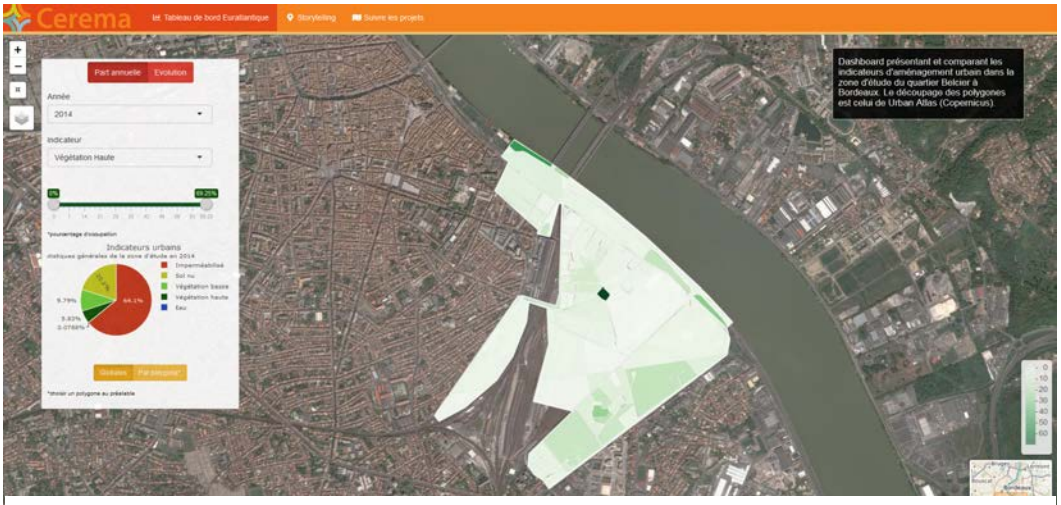


Figure 9 : Application Euratlantique - Onglet DashBoard

Figure 11 : Application Euratlantique - Onglet Storytelling

Figure 10 : Application Euratlantique - Onglet Suivi des projets

Avec l'arrivée du train à grande vitesse à Bordeaux et la volonté de faire rayonner la métropole au niveau européen, une opération d'intérêt national a émergé dans l'objectif de répondre à un projet de renouvellement urbain et de modernisation de la zone centrée autour de la gare. Au regard des nombreux projets éparpillés sur le secteur, différents organismes ont critiqué et montré leurs inquiétudes quant à la forte imperméabilisation que cela pouvait représenter. Le Cerema a pour mission de montrer l'évolution de ces indicateurs d'aménagement. La dataviz a ici un réel intérêt tout d'abord pour l'EPA Bordeaux-Euratlantique afin de suivre ces indicateurs dans le secteur d'aménagement mais il est imaginable que cela pourra être un moyen de communication et de vérification auprès des financeurs, lors de réunions publiques par exemple.

3. Application paramétrable

Lors de ce stage, j'ai tenté de développer une application paramétrable par l'utilisateur afin qu'il insère ses propres jeux de données dans un script générique.

3.a) Le projet

L'idée de cette application est de développer un outil qui permettra à l'utilisateur de paramétrer avec les données et les variables qu'ils souhaitent un tableau de bord. L'application est développée sur RShiny et l'utilisateur n'aura qu'à ouvrir le fichier de paramétrage sur RStudio pour pouvoir modifier comme il le souhaite la visualisation.

3.b) Public visé

Le public visé sont les agents du pôle satellite du Cerema. L'idée est de créer une application facile et rapide d'utilisation pour des agents qui n'ont jamais utilisé RShiny voir R. Une documentation ainsi qu'un guide d'utilisation ont été produits.

3.c) Fonctionnalités attendues

L'application doit permettre de représenter spatialement et graphiquement des données. Elle embarque alors une carte interactive avec différents graphiques qui vont analyser et comparer la donnée : *barchart*, *piechart*, *treemap* (graphique à cases) et *bubblechart* (graphique à bulles).

Voici un aperçu avec un jeu de données sur l'occupation du sol de quelques communes dans les Hautes-Pyrénées.

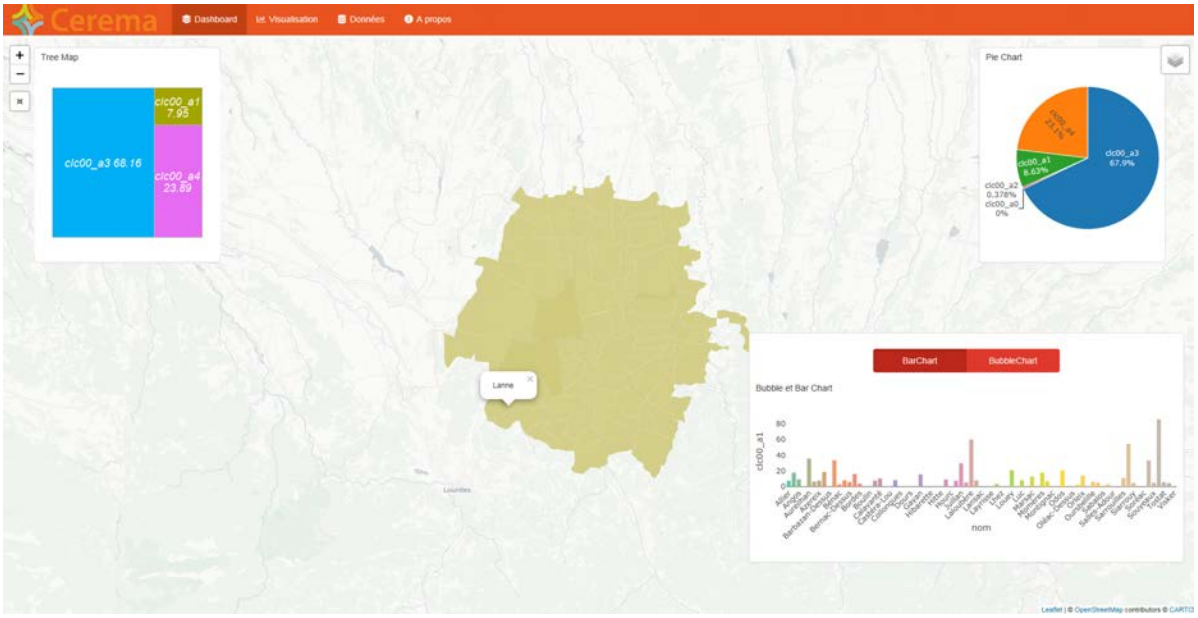


Figure 12 : Application paramétrable - Onglet Dashboard

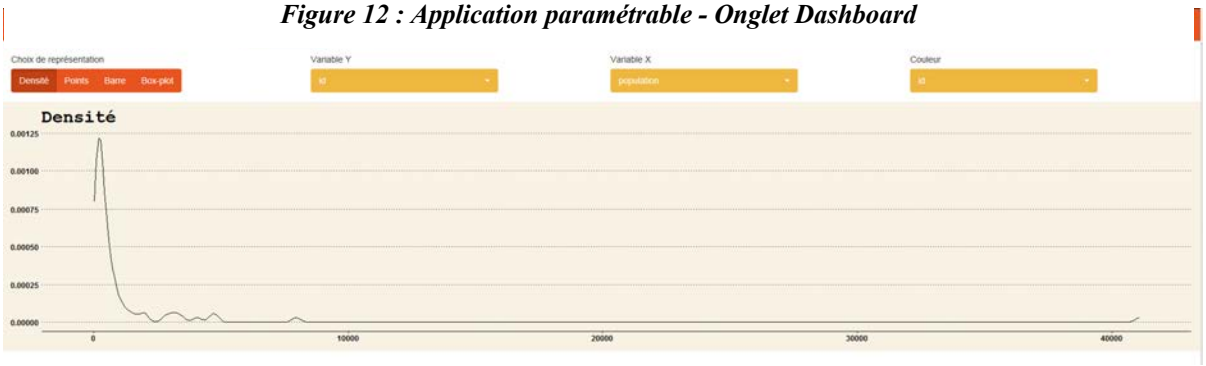


Figure 13 : Application paramétrable - Onglet Visualisation

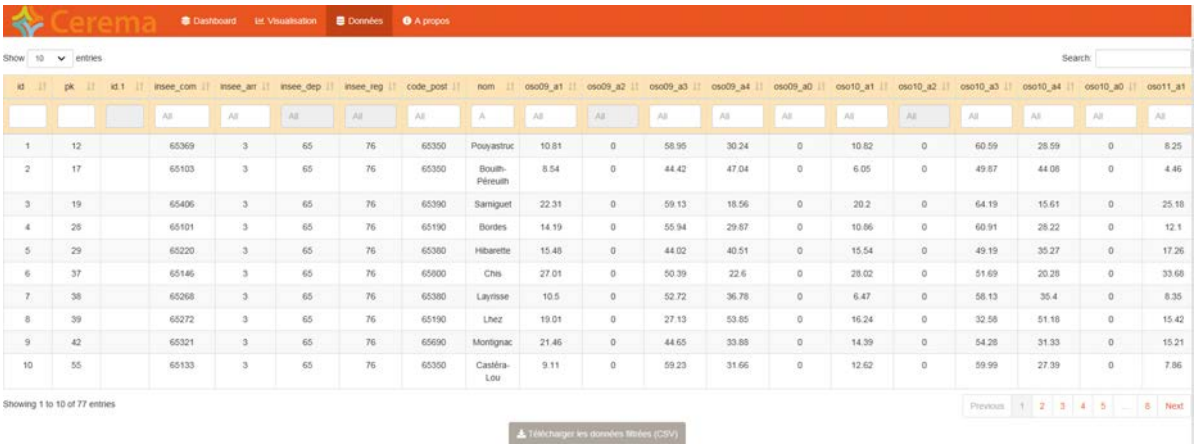


Figure 14 : Application paramétrable - Onglet Données

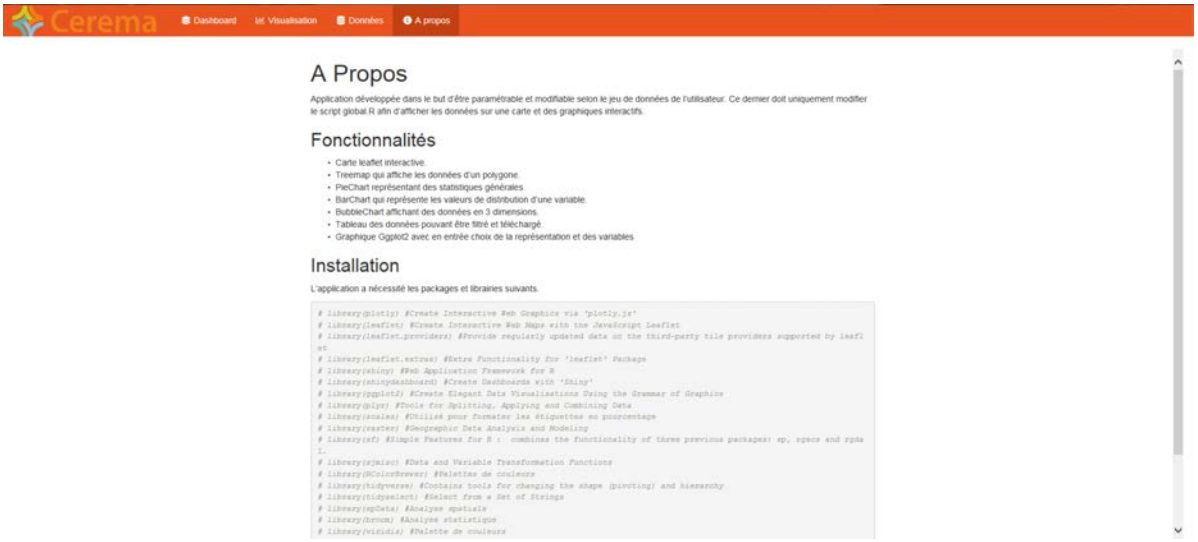


Figure 15 : Application paramétrable - Onglet A Propos

Après avoir présenté l'ensemble des projets, je vais détailler plus particulièrement le développement des applications avec pour chacune d'elles, les fonctionnalités existantes et les limites rencontrées.

V. Phase de développement et manipulation des outils

Avant de développer les outils, nous avons avec l'équipe défini les fonctionnalités importantes et prioritaires et celles qui l'étaient moins. Cela s'est fait au regard de la bibliographie, d'exemples d'applications et des besoins de l'équipe. Les *dashboards* développés ont alors permis de tester chaque bibliothèque choisie à savoir Python Dash et RShiny. Edson Olivares Medina, autre stagiaire, a entrepris de développer des outils sous Dash et quant à moi je me suis concentré sur RShiny.

L'idée était de définir les fonctionnalités qui semblaient prioritaires dans un *dashboard* telles que l'affichage de fichiers vecteurs, de cartes dynamiques ou des graphiques (« voir tableau 1 ») : ce travail a été réalisé au cours de la phase de bibliographie et pendant la prise en main des outils. Dans cette même idée de vouloir connaître les limites et avantages, nous avons développé en premier lieu une application qui tente de prendre en compte l'ensemble des fonctionnalités prioritaires dégagées. C'est dans cette logique que s'inscrit l'application sur l'occupation du sol en Hautes-Pyrénées. L'application Euratlantique tente elle d'explorer d'autres fonctionnalités et diverses méthodes de dataviz qui n'ont pas été réalisées dans la 1ère application.

Tableau 1 : Tableau des fonctionnalités et des priorités

Fonctionnalités	Précisions	Priorité*
Lecture de fichiers vecteurs		1
Lecture de fichiers rasters		5
Lecture WMS/WMTS		2
Lecture WFS		2
Calcul de statistiques	Calculer de nouvelles données par rapport aux données initiales	3
Affichage de graphiques	Statique et dynamique	2
Cartographie dynamique		2
Enregistrement des résultats	Export PDF/PNG, téléchargement des données (csv, etc.)	5
Gestion des données volumineuses		4
Export en HTML		4
Curseurs altérant l'affichage		5
Géovisualisation temporelle		5
Ajout de couches au projet	L'utilisateur peut uploader lui-même les données	2

* Ordre de priorité des fonctionnalités à tester : 1 étant à tester en premier et 5 en dernier.

Avant de présenter les applications développées, il est important de prendre connaissance du *package* Shiny et de son fonctionnement.

1. Présentation de RShiny

Shiny est un *package* R, développé par RStudio, qui permet la création d'applications web interactives sur lesquelles il est possible de réaliser toutes les analyses / actions disponibles sous R. Sa grande force est le fait qu'il n'y a absolument pas besoin de connaître ni HTML, ni CSS, ni JavaScript : tout se fait directement en R.

Après avoir installé et chargé le *package* Shiny dans R, il faut créer l'application Shiny qui sera composée de deux fichiers : un fichier « ui.R » et un fichier « server.R ». Il est notamment possible d'avoir un fichier « global.R » ainsi que des dossiers « www » et « data ». Les fonctionnalités et intérêts de ces fichiers/dossiers sont détaillés dans la figure 16.

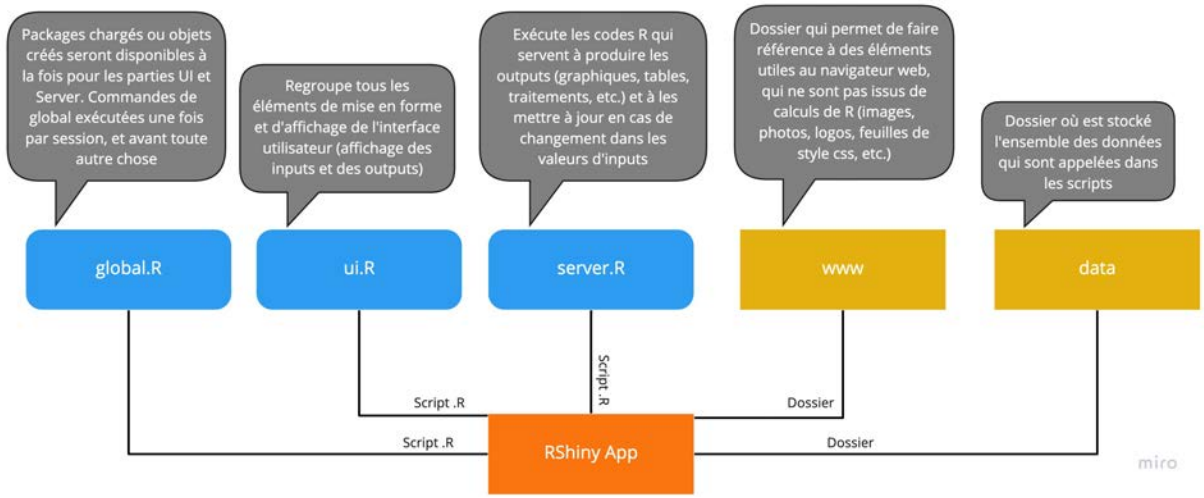


Figure 16: Structure d'une application Shiny

Concernant la conception des interfaces des applications, j'ai été fortement inspiré et aidé des nombreuses applications Shiny qu'il est possible de visualiser sur le site du *package*[15]. L'avantage est que pour la plupart de ces exemples, le code source est à disposition. Les choix et interfaces graphiques sont très différents ce qui m'a donné un large panel d'exemples. Cela m'a été d'une grande aide dans la conception de ces applications notamment au début dans la compréhension et dans la prise en main de Shiny. De plus, des outils interactifs réalisés par la DREAL Pays de la Loire m'ont permis d'avoir un aperçu concret des fonctionnalités et de l'intérêt de ces applications dans la mise en valeur des données[16].

Les applications développées ont été, dans un premier temps, visualisées en local puis nous avons trouvé une solution d'hébergement relativement pérenne. En effet, le Cerema Méditerranée a mis à disposition du stage leur serveur payant shiny.io qui permet d'héberger un nombre illimité d'application. La question se pose sur le format du livrable et sur l'utilisation de serveurs commerciaux pour des applications à destination de services publics utilisant des données partiellement non publiques. Je reviens sur ces enjeux dans la 3ème sous partie.

2. Les applications

2.a) Panorama des diverses sources d'OCS en Hautes-Pyrénées

La première prise en main sur RShiny s'est réalisée sur le jeu de données présenté dans la partie précédente à savoir l'Occupation du sol (OCS) dans le département des Hautes-Pyrénées (65). Pour développer un tel outil, nous avons listé tout au long de cette phase de test les fonctionnalités importantes et à tester dans le *dashboard* pour en connaître les limites et avantages.

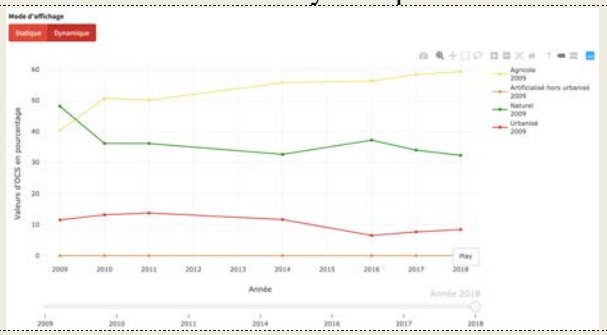
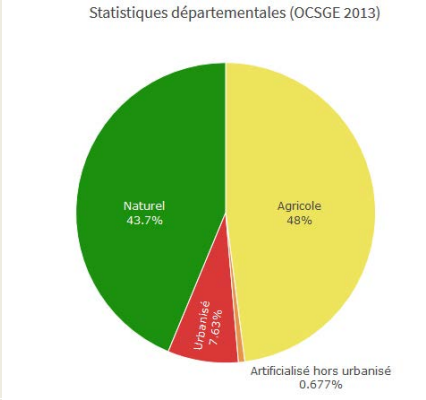
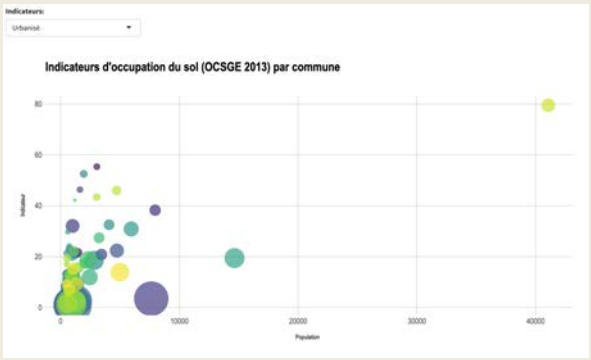
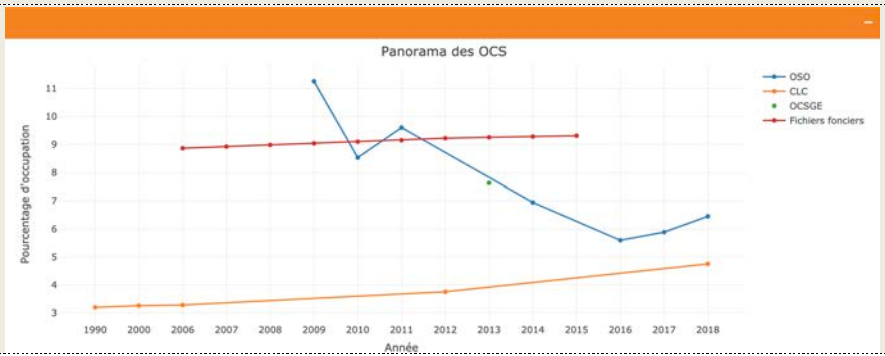
[15] "Gallery", Shiny.RStudio, https://shiny.rstudio.com/gallery/
[16] DREAL Pays de la Loire, « Les pesticides dans les cours d'eau des Pays de la Loire », http://apps.datalab.pays-de-la-loire.developpement-durable.gouv.fr/qualite-des-eaux/

2.a.1 Fonctionnalités

Pour réaliser ce *dashboard*, je suis parti de presque rien en termes de compétences. J'avais seulement une formation sur R grâce au module vu au premier semestre du Master mais aucune sur RShiny. Dans un premier temps, le développement s'attardait essentiellement sur de l'affichage de cartes et graphiques simples. L'avantage du *package* Shiny est qu'il est très bien documenté sur internet et qu'il dispose d'une communauté très active sur de nombreux forums. Prendre en main l'outil est alors assez aisé d'autant plus qu'il n'y a aucune compétence à avoir dans un langage de type web.

Concernant l'affichage de graphique, j'ai opté pour la bibliothèque plotly.R qui s'appuie sur le package open-source plotly.js. L'avantage de cette bibliothèque est qu'elle rend les graphiques interactifs et dynamiques. Le panel de représentation est très large (« voir tableau 2 ») et la documentation concernant le script et le paramétrage de ces graphiques y est plutôt complète.

Tableau 2 : Représentations 'Plotly' utilisées dans l'application OCS

<table>
<tr><td>Linechart dynamique</td><td>Barchart</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Piechart</td><td>Bubblechart</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Linechart</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

L'avantage de cette bibliothèque est qu'il est aussi possible de convertir des figures statiques réalisées avec la bibliothèque ggplot2 en figures interactives (« ggplotly() ») ce qui offre beaucoup de possibilités.

Autre bibliothèque utilisée pour créer un *treemap* ou carte à case est Treemapify qui fonctionne sur ggplot2 et n'est pas interactif (« voir figure 17 ») :

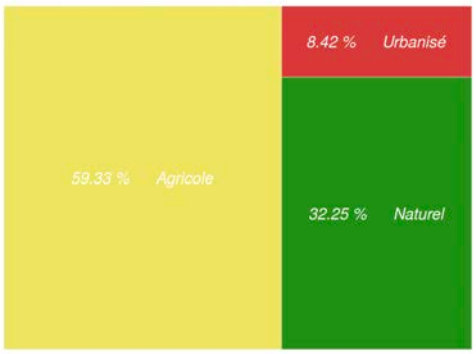


Figure 17: Treemap ou graphique à cases

Ces diverses représentations graphiques permettent d'analyser plus précisément les données. En effet, il est possible de décomposer des statistiques départementales avec le *piechart* ou communales avec le *Treemap*, de visualiser les communes en fonction de plusieurs dimensions comme dans le *Bubblechart* (taille, couleurs, abscisse, ordonné), de comparer les occupations du sol à différentes années ou encore de voir l'évolution des usages (*Linechart*).

Cela permet de se rendre compte que cette étude comparative sur la question de la mesure de l'artificialisation via plusieurs bases de données (Corine Land Cover, OSO, OCSGE et les fichiers fonciers), ne fait pas apparaître les mêmes valeurs à une date donnée, ni sur les dynamiques d'évolution, où des désaccords sur les tendances peuvent même être observés.

Concernant la cartographie interactive, Leaflet a été choisi car c'est la bibliothèque JavaScript open-source la plus populaire (« exemple de représentation dans la figure 18 »). Elle dispose de nombreuses fonctionnalités et est très bien documentée. Les polygones des communes de Hautes-Pyrénées sont chargés grâce au *package* SF qui s'avère être une bibliothèque pour gérer des données spatiales, plus rapide que Rgdal. Il est important de savoir que les données en entrée doivent être en système de référence WGS84 EPSG 4326 sur une carte leaflet. L'interface de programmation (API) Leaflet possède de nombreuses fonctionnalités telles que la possibilité de modifier les fonds de carte, d'afficher les labels et/ou *popups*, de refixer l'affichage par rapport au zoom par défaut, de sélectionner une commune dans une barre de recherche, etc.

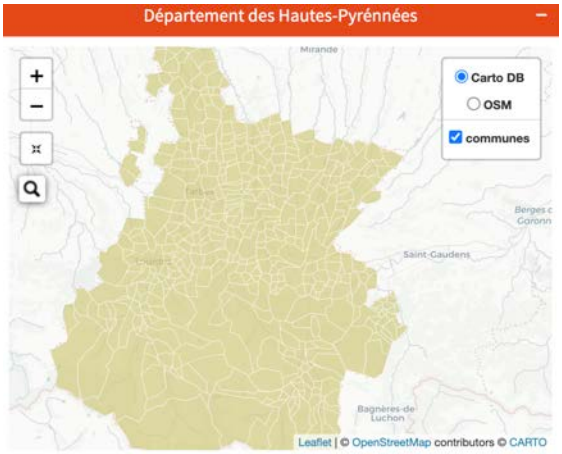


Figure 18: Carte Leaflet interactive

Concernant maintenant les fonctionnalités propres à Shiny, l'application en utilise plusieurs. En effet pour gérer la réactivité, il existe plusieurs fonctions :

- « reactiveVal » : construit un objet réactif c'est-à-dire qu'il prend une valeur comme une variable mais dès que la valeur change, tous les éléments qui dépendaient de cette valeur vont être modifiés.

```
rv <- reactiveVal()
```

- « observeEvent » : destinée à être exécuté lorsqu'une variable réactive est "déclenchée" et est censée avoir des effets secondaires. Exécution d'une action en réponse à un événement.

```
observeEvent(input$mapcom_shape_click, {
  rv(input$mapcom_shape_click$id)
  })
```

('mapcom' correspond à l'identifiant de l'*output* de la carte leaflet)

Ces deux fonctions sont utilisées dans le *dashboard* pour récupérer l'identifiant du polygone lorsque l'on clique dessus. L'action 'clique' va avoir pour effet de récupérer cet identifiant et de le mettre dans une variable réactive qui changera à chaque nouvelle action.

- « selectInput » : liste de sélection créé dans l'ui.R avec un identifiant et les variables à choisir. La valeur choisie est ensuite récupérée dans le fichier server.R grâce à cet identifiant :

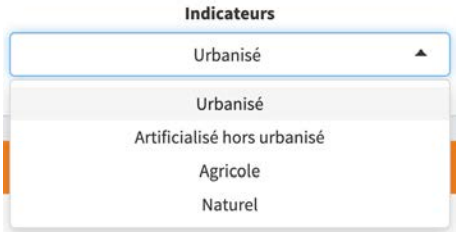


Figure 19: Sélecteur déroulant (Shiny Widgets)

Figure 20: Checkbox (ShinyWidgets)

- « checkbox » :

La pluralité de fonctionnalités de Shiny est une vraie force du *package* tout comme le fait qu'il est possible d'y intégrer n'importe quelle bibliothèque disponible sur R pour de la représentation graphique. La conception d'une application implique de concevoir l'interface et les fonctions réactives qui vont la modifier pour répondre aux demandes de l'utilisateur (« voir figures 19 et 20 »). C'est ce qui est appelé en informatique une programmation évènementielle (« voir figure 21 »). En l'occurrence, le programme est défini par des réactions à des évènements que va produire l'utilisateur : sélection, clique, changer d'onglet, etc.

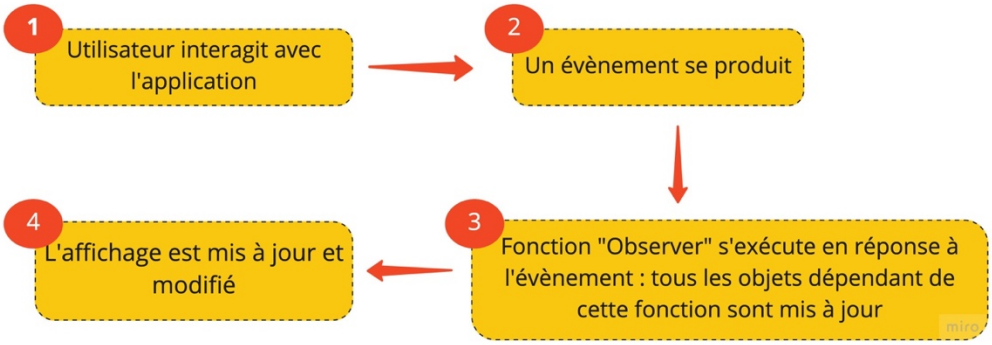


Figure 21 : Programmation évènementielle dans un interface graphique

Néanmoins cette première application sur l'occupation du sol n'en est pas à sa première version, car j'ai dû faire face à de multiples problèmes notamment concernant la latence de l'outil.

2.a.2 Limites

La première version complète de l'application tentait d'afficher des fichiers de type raster. La bibliothèque Raster sous R permet de les charger et Leaflet grâce à la fonction « addRasterImage » permet de les afficher sur une carte interactive. L'idée était de visualiser uniquement la commune sélectionnée dans la carte du département ainsi que la couche d'occupation du sol liée à l'onglet sélectionné au préalable. La « figure 22 » est une planche explicative du fonctionnement de l'application et des réactions que l'utilisateur provoque par ses choix.

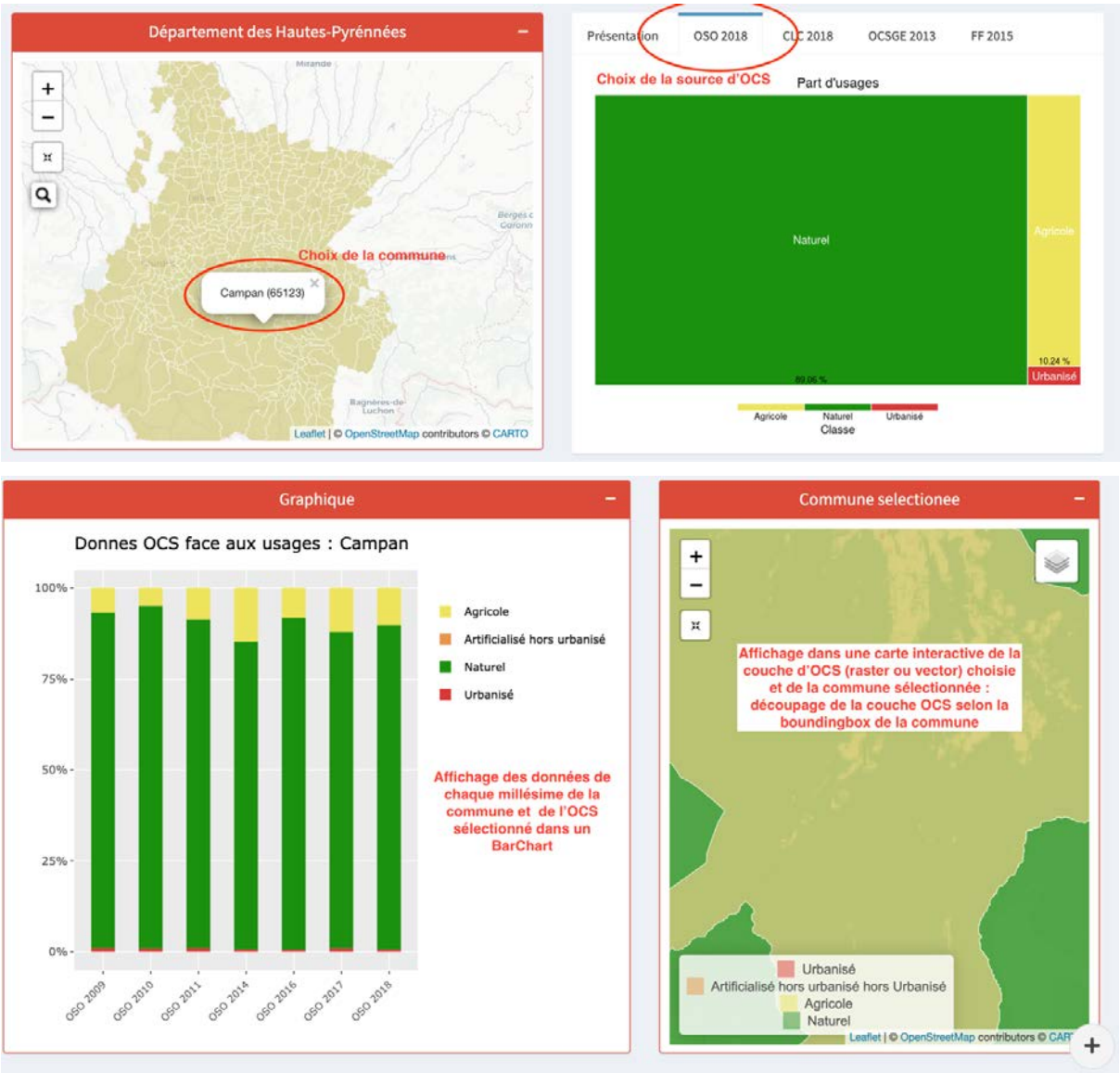


Figure 22: Affichage interactif dans la V1 de l'application OCS

Dans le script cela se traduit par :

```
communes_sf <- read_sf(dsn="data/ocs_db.sqlite", layer="stats_co_dept65") %>%
filter(insee_com == rv())
   coords <- st_coordinates(communes_sf$geom)
   lng <- mean(coords[,1])
   lat <- mean(coords[,2])
```

Pour la commune sélectionnée, on crée une nouvelle variable *communes_sf*. La fonction « st_coordinates » permet de récupérer les coordonnées sous forme matricielle et on crée deux nouvelles variables pour la longitude et la latitude.

```
oso_2018 <-  crop(raster("data/oso18_agregate_4326.tif"), extent(communes_sf), snap="out")
```

La fonction « crop » de la bibliothèque Raster permet de découper un fichier raster selon une étendue. En l'occurrence, on découpe l'OSO de 2018 selon les coordonnées de la commune sélectionnée.

```
leaflet() %>%
    addProviderTiles(providers$CartoDB.Positron, group = "Carto DB") %>%
    setView(lng, lat, zoom = 13.2) %>%
```

Ensuite dans Leaflet, on spécifie les coordonnées par défaut de la carte selon la latitude et la longitude que l'on a récupéré de la commune sélectionnée.

J'ai réussi à développer les fonctionnalités et visualisations souhaitées mais l'application est confrontée à de fortes latences voire à des déconnexions de l'application en local. Cela provient de l'affichage des rasters et des vecteurs notamment l'OCSGE qui est très lourd (beaucoup de polygones). Pour tenter de palier ce problème, j'ai essayé plusieurs méthodes :

→ *Simplifier la géométrie et augmenter la taille des pixels*

La fonction « aggregate » du *package* Raster permet de réduire la résolution d'une image raster. L'agrégation regroupe des zones rectangulaires pour créer des cellules plus grandes basée avec des valeurs calculées sur la moyenne des zones.

```
oso_2018 <- raster::aggregate(oso_2018)
```

Concernant les vecteurs, la bibliothèque Rmapshaper dispose d'une fonction « ms_simplify » qui permet de simplifier la géométrie.

```
layer_simplify <- ms_simplify(vecteur1)
```

Dans ces deux cas, la taille des fichiers s'est considérablement réduite et cela a permis de réduire le délai d'affichage mais il reste toujours trop long.

→ *Utiliser une base de données SQLite*

La latence pouvait alors venir d'autre part, par exemple dans la manière de requêter les données ou la manière dont elles sont stockées. Le pôle satellite du Cerema ne possède pas de Systèmes de gestion de base de données (SGBD) sur serveur comme PostGreSQL ou autres pouvant être utilisées dans le cadre du stage. Nous avons alors essayé d'utiliser une solution locale de SGBD à savoir SQLite.

J'ai créé une base de données SQLite via Qgis où j'ai pu intégrer directement les couches spatiales au sein de la base de données à savoir les couches vecteurs et les *dataframes*. À savoir que SQLite ne permet pas de stocker des fichiers de type raster. Il est possible aussi de faire l'ensemble de ces manipulations directement depuis RStudio notamment grâce aux bibliothèques DBI et RSQLite. C'est aussi ces dernières qui permettent de charger la base de données SQLite et de requêter des données avec le module « dbGetQuery » en effectuant des requêtes en SQL comme ci-dessous :

```
df_com_pivot <- dbGetQuery(dbConnect(RSQLite::SQLite(), bdd_sqlite), 'SELECT source,
annee, value, classe, nom FROM df_com_pivot')
```

Cela n'a pas réduit pour autant le délai d'affichage. Pour comprendre où l'application Shiny passe son temps, j'ai utilisé deux autres bibliothèques.

→ *Profilage de l'application*

Le *package* Profvis est un outil pour faire du profilage c'est-à-dire pour comprendre comment R passe son temps afin de faciliter l'optimisation du programme et donc de l'application. « La plupart des utilisateurs de R ont eu des moments où nous voulions que notre code s'exécute plus rapidement. Cependant, il n'est pas toujours clair comment y parvenir. Une approche courante consiste à s'appuyer sur l'intuition et sur la sagesse de la communauté R au sens large pour accélérer le code R. Un inconvénient est que cela peut conduire à se concentrer sur l'optimisation des éléments qui ne prennent en fait qu'une petite partie du temps de fonctionnement global »[17].

Profvis permet de connaître, pour chaque fonction dans le script, le temps passé par R pour sortir le résultat. Ci-dessous, le temps passé sur la 1ère version de l'application OCS qui affichait les rasters et utilisait une base de données SQLite : à droite la mémoire utilisée et le temps passé (chiffres tout à droite).

Figure 23: Profvis, une bibliothèque pour déterminer où R passe son temps

On peut remarquer que ce ne sont pas forcément les cartes qui demandent le plus de temps d'affichage bien qu'ils soient tous en haut de la liste. En effet, les graphiques utilisant Plotly semble utiliser beaucoup de mémoire et nécessiter un délai d'affichage conséquent.

Cette bibliothèque est intéressante pour voir où Shiny passe son temps. En effet, elle a permis de comprendre que le *package* Plotly peut amener à des latences. Elle demande beaucoup plus de temps d'affichage et de mémoire du fait de son interactivité. L'exemple ci-dessous montre à quel point la fonction « ggplotly » augmente considérablement le temps passé. Ces tests ont été réalisés en local uniquement avec un seul utilisateur manipulant le tableau de bord.

17 https://rstudio.github.io/profvis/

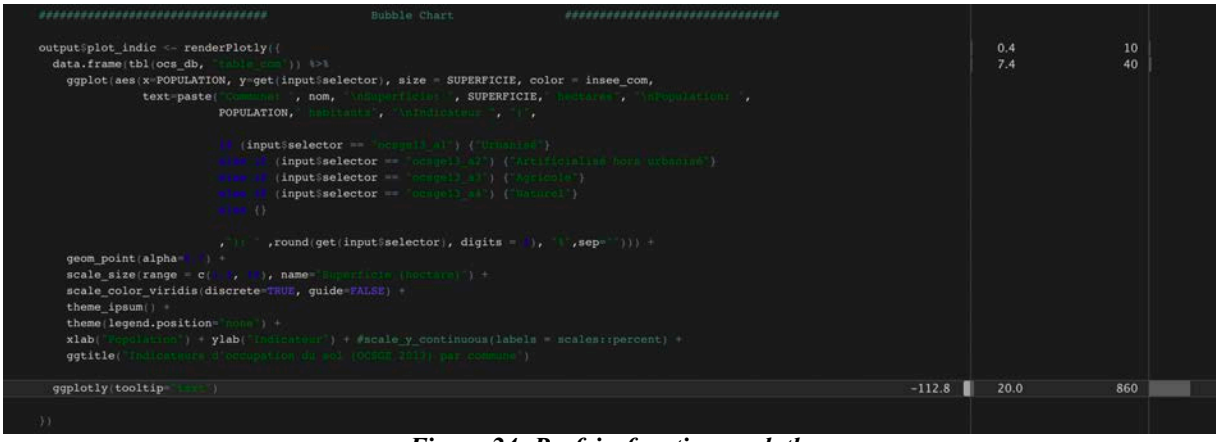


Figure 24: Profvis, fonction ggplotly

La latence générale de l'application peut venir du fait d'affichage de plusieurs graphiques issus de la bibliothèque « Plotly ». Néanmoins, il semble que ce profilage ne calcule pas ou ne permet pas de se rendre compte du temps d'affichage réel des cartes interactives.

Figure 25: Profvis, comparaison des cartes

→ *output$mapdept* : affiche les polygones des communes
→ *output$mapcom* : affiche les polygones des communes avec valeurs réactives
→ *output$mapcom2* : affiche le polygone de la commune sélectionnée + source d'OCS sélectionnée (raster et vecteur).

Ce que ce *package* ne prend pas en compte dans le temps passé est le calcul et les manipulations spatiales. En réalité, ce qui ralentit l'application est le chargement des différentes couches d'OCS que ça soit raster ou vecteur et toutes les opérations que cela nécessite : récupération des coordonnées de la commune sélectionnée, découpage de la source d'OCS par rapport aux coordonnées de la commune puis affichage de ces couches dans une carte Leaflet. Tout ce traitement doit être reproduit pour chaque nouvelle commune sélectionnée. Cette carte s'avérait être une source des latences car nous avons décidé de l'enlever ce qui a permis une meilleure fluidité mais pas ce que nous espérions. En effet, un délai d'affichage persiste toujours mais cela vient, selon moi, de l'affichage cumulé de graphiques « Plotly ».

Une autre bibliothèque qui m'a aidé dans la construction de l'application est TicToc qui permet de mesurer le temps d'exécution de commandes dans R. Voici un exemple sur des commandes pour requêter des données dans une base de données SQLite. L'idée est d'optimiser le temps d'exécution du script en choisissant les commandes les plus performantes :

```
tic("connect")
df_com_pivot <- dbGetQuery(dbConnect(RSQLite::SQLite(), bdd_sqlite), 'SELECT source,
annee, value, classe, nom FROM df_com_pivot')
df_com_pivot
toc()
connect: 0.085 sec elapsed

tic("noconnect")
data_frame2 <- data.frame(tbl(sqlite, "df_com_pivot")) #création d'un dataframe
```

data_frame2
toc()
noconnect: 0.237 sec elapsed

L'application OCS a permis d'éclaircir les fonctionnalités possibles et les limites de RShiny et de R plus généralement. En effet, bien que de nombreuses fonctionnalités soient implémentées dans le *package*, des limites persistent notamment concernant les jeux de données volumineux, le stockage des données et l'affichage de multiples cartes et/ou graphiques. Ces limites énoncées sont en réalité des problèmes de performance.

Pour aller plus loin, nous avons décidé de partir sur un autre jeu de données : projet de Bordeaux Euratlantique. Il était question de changer complètement l'affichage du *dashboard* pour tester de nouvelles fonctionnalités et visualisations et se confronter à d'autres limites potentielles.

2.b) Euratlantique

Ce *dashboard* cherche à explorer d'autres types de visualisations comme notamment le *storytelling* ainsi que d'autres fonctionnalités comme les curseurs. Après avoir présenté le projet dans la partie précédente et notamment l'aperçu final de l'application, il est décrit dans cette partie avec plus de détails les fonctionnalités implémentées ainsi que les limites rencontrées.

2.b.1 Fonctionnalités

- Thème du *dashboard*

Pour ce tableau de bord, j'ai opté pour un modèle graphique différent de celui de base de RShiny (« shinydashboard ») : Bootstrap. C'est un framework open-source comprenant des codes HTML, CSS et JavaScript utiles à la conception du design de sites et d'applications web facilitant l'aspect *responsive* et la réactivité ainsi que l'implémentation de multiples composants prédéfinis.

Structure Bootstrap :

```
ui <- bootstrapPage(

  navbarPage(
    windowTitle = "Euratlantique", # Nom de la page web

```

Comparaison avec la structure Shinydashboard :

```
shinyUI <- shinyUI (dashboardPage(

  skin = "red",

  # En tête de la page
  dashboardHeader(title = "Données OCS",
```

Puis on importe les fichiers de styles et le fichier HTML qui appelle les bibliothèques CSS et JS.

```
27              div(class="outer",
28                  # on rattache le fichier css "style" à notre script
29                  tags$head(includeCSS("www/styles.css")),
30                  tags$head(includeHTML("www/librairies.html")),
```

Le thème Bootstrap permet d'avoir plus la main sur l'apparence et le style du *dashboard* cependant, cela nécessite de toucher un peu plus le CSS dans l'application. Il permet notamment de créer les fenêtres déplaçables sur la carte comme dans le 1er onglet (« figure 26 »).

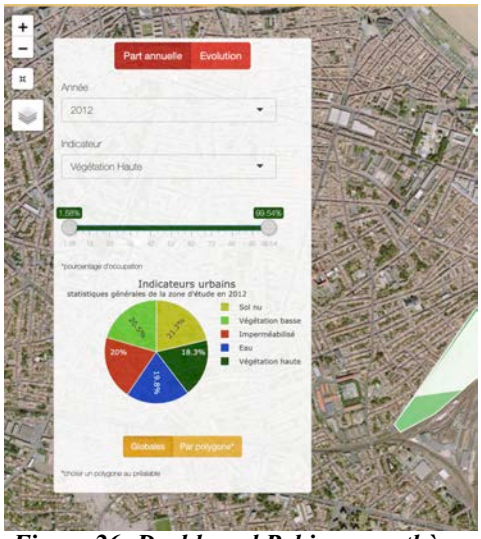


Figure 26: Dashboard Rshiny avec thème Bootstrap

- Des sélecteurs pour laisser le choix à l'utilisateur des données à afficher

Pour sélectionner les données à afficher, deux sélecteurs ont été créés :

- Choix de l'année :

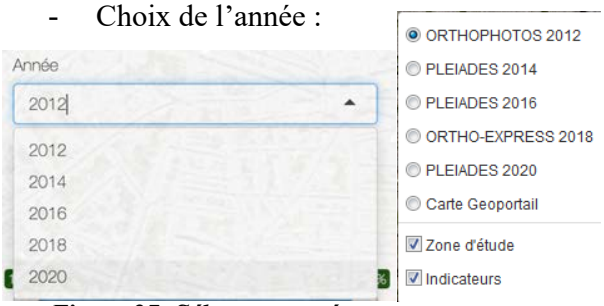


Figure 27: Sélecteur année

Le choix de l'année modifie le fond de carte pour afficher une image satellite à la date sélectionnée.

- Choix de l'indicateur :

Figure 28: Sélecteur indicateur

Le choix de l'indicateur permet de choisir les données à afficher sur la carte. Cela modifie la couleur du curseur, des polygones et met à jour la légende.

- Un curseur pour altérer l'affichage de la carte

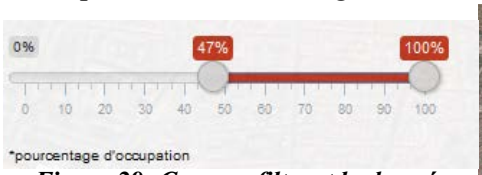


Figure 29: Curseur filtrant la donnée affichée

Le 'sliderInput' de Shiny est une fonction qui prend en entrée une valeur minimale et une valeur maximale. Selon l'indicateur sélectionné, ces valeurs changent et se mettent à jour. Le curseur agit comme filtre du jeu de données en entrée. Le jeu de données devient alors une valeur réactive en fonction de l'année choisie, de l'indicateur et du champ du curseur. Il va se mettre à jour à chaque variation du *slider* en prenant uniquement les valeurs égales ou supérieures à la borne 'minimum' et égales ou inférieures à la borne 'maximum'.

Création du *slider* dans le fichier ui.R :

```
sliderInput("range", "", min = min(bdx_polygones$X2016_vgh), max = max(bdx_polygones$X2016_vgh),
            value = range(bdx_polygones$X2016_vgh), round = -2, sep = "" , step = 1, post = "%", ticks = TRUE)
```

Données réactives en fonction du *slider* dans le fichier server.R :

```
filteredData <- reactive({
  bdx_polygones[bdx_polygones$X2016_vgh >= input$range[1] & bdx_polygones$X2016_vgh <= input$range[2],]
```

Cette fonctionnalité apporte vraiment son intérêt dans l'application. Cela permet de visualiser rapidement et simplement les zones qui se sont le plus imperméabilisées ou celles qui sont très végétalisées par exemple à une année donnée.

- Du dynamisme quand la page défile

Une nouvelle fonctionnalité exploré dans ce *dashboard* est ce que l'on appelle le *storytelling* ou *scrollytelling* ou encore ce qu'appelle ArcGis la *storymap*. L'idée est de dérouler une page avec des informations et des visualisations qui s'affichent au fur et à mesure du '*scroll'* (déroulement de la page). Il n'existe pas vraiment de bibliothèques dans RShiny pour réaliser ce type de visualisation de manière interactive à la manière d'une parallaxe où une représentation (une carte par exemple) évolue (ajout de markers par exemple) au fur et à mesure que la page descend. Néanmoins, nous avons créé une page où l'affichage est chronologique avec l'état des lieux de chaque année sur la zone d'étude en rajoutant du dynamisme lorsque l'on arrive à hauteur d'une visualisation. Ce dynamisme est possible grâce à la bibliothèque Shinyanimate et sa fonction « addScrollAnim » ou encore « addHoverAnim ».

```
1364  observe(startAnim(session, 'storymap', 'fadeInUp'))   # animation se déclenche quand on voit l'élément
1365  observe(addScrollAnim(session, 'linechart_1416', 'fadeIn')) # animation lorsque l'on scroll
1366  observe(addHoverAnim(session, 'belcierprojet', 'pulse')) # animation quand souris passe sur l'objet
```

- Des cartes synchronisées

Pour permettre de comparer l'évolution des indicateurs d'aménagement urbain ainsi que de visualiser les réalités terrains avec les résultats des études, des cartes interactives leaflet synchronisées ont été implémentées dans le 2ème et 3ème onglet mais de différentes manières.

- La 1ère carte synchronisée se situe à la dernière ligne du 2ème onglet :

Figure 30: Synchronisation de cartes leaflet

Deux cartes dans deux *outputs* différents sont créées : 'map_raster1' et 'map_raster2'. L'idée est de synchroniser la carte 'map_raster1' par rapport au niveau de zoom et coordonnées de 'map_raster2'. Pour cela, on récupère les coordonnées des limites de la carte (*bounding box*). Ensuite, grâce à la fonction *leafletProxy* qui modifie une carte qui est déjà en cours d'exécution dans la page, la situation et le zoom de la 2ème carte se mettent à jour automatiquement et le tout dans une fonction *observe* de Shiny qui déclenche une action à chaque fois que ses données en entrées changent.

```
1291  # Permet de synchroniser le niveau de zoom et la localisation d'une carte avec l'autre
1292  observe({
1293    coords <- input$map_raster2_bounds
1294    if (!is.null(coords)) {
1295      leafletProxy("map_raster1") %>%
1296        fitBounds(coords$west,
1297                  coords$south,
1298                  coords$east,
1299                  coords$north)
1300    }
1301  })
```

Cela permet de comparer les résultats des traitements d'images satellites réalisés par l'équipe SCGSI (rasters) par rapport à la réalité terrain (fonds de carte orthophotos et Pléiades) et au regard de l'évolution des projets dans le secteur d'aménagement.

- La 2ème fonction utilisée pour synchroniser des cartes se situe dans le 3ème onglet :

Figure 31: Synchronisation de multiples cartes pour visualiser l'évolution de l'OCS

Pour réaliser ces 5 cartes synchronisées, j'ai utilisé une fonction différente de celle de l'onglet précédent. En effet, dans ce cas-là j'ai utilisé la bibliothèque Leafsync qui permet de synchroniser plusieurs cartes Leaflet. L'ensemble des cartes doivent se situer dans un même *output*. Ci-dessous quelques lignes de code montrant la manière de créer ces cartes :

```
1472  # Output qui créé plusieurs cartes leaflet et qui les synchronise
1473  output$synced_maps <- renderUI({
1474    if(is.null(reactvalue())){
1475      HTML(paste(""))
1476    }
1477    else{
1478      # On filtre la donnée des markers par rapport au clickEvent (valeur réactive)
1479      # On récupère la latitude et longitude du marker que l'on a sélectionné et on centre la vue sur ces coordonnées
1480    markers_filtered <- markers %>%
1481      filter(name == reactvalue())
1482    m12 <- leaflet(markers_filtered) %>%
1483      setView(lng = markers_filtered$lng, lat =markers_filtered$lat, zoom = 17.5) %>%
1484      addWMSTiles(url_sat_wms
1485                                  ,layers = layer_sat_2012, group =group_sat_2012) %>%
1486    addLayersControl(
1487      baseGroups = c(group_sat_2012),
1488      position = "topright",
1489      options = layersControlOptions(collapsed = FALSE))
1490
1491    m14 <- leaflet(markers_filtered) %>%
1492      setView(lng = markers_filtered$lng, lat =markers_filtered$lat, zoom = 17.5) %>%
1493      addWMSTiles(url_sat_wms
1494                                  ,layers = layer_sat_2014, group =group_sat_2014) %>%
1495      addLayersControl(
1496        baseGroups = c(group_sat_2014),
1497        position = "topright",
1498        options = layersControlOptions(collapsed = FALSE))
```

Ensuite il suffit d'utiliser la fonction *sync* en mettant en paramètre l'ensemble des variables créant les cartes leaflet.

```
1526      sync(m12, m14, m16, m18, m20, ncol = 5)
```

Cet onglet permet de visualiser aisément l'évolution de l'occupation du sol à différentes dates. C'est un moyen de suivre les projets et de réaliser du *monitoring* sur la zone d'étude.

- Flux WMS (clé de l'IGN)

Pour le *dashboard* Euratlantique, nous avons utilisé des flux WMS issus de Géoportail et de l'IGN afin d'afficher les imageries satellites en fond des cartes interactives. Nous nous sommes dotés d'une clé IGN par référer qui est la solution adaptée dans le cas d'une application web. En effet, elle fonctionne uniquement sur l'URL de la page utilisant les ressources à savoir l'adresse du serveur payant shiny.io de Cerema Méditerranée (je reviens en détail dans la 3ème sous-partie sur cette question d'hébergement). Cela nous donne accès aux imageries Pléiades et aux orthophotos uniquement sur les applications ayant ce nom de domaine.

2.b.2 Limites

Globalement, nous avons réussi à implémenter l'ensemble des fonctionnalités souhaitées sur ce *dashboard*. Cela nous a permis en plus d'explorer de nouvelles façons et manières de visualiser les données. Je me suis aidé d'exemples disponibles sur le site Shiny notamment « SuperZip example » développée par Joe Cheng[18]. Certaines limites ont néanmoins été rencontrées mais cela rejoint finalement les problèmes rencontrés dans l'application précédente.

Le *package* « Profvis » nous permet de bien visualiser le problème.

▼ runApp	-4385.0	4450.1	20020
▶ output$map_raster1	-3965.1	4108.0	12770
▶ output$piechart	-18.3	19.3	380
▶ output$treemap_12	-8.5	8.4	330
▶ output$treemap_16	-3.8	1.4	270
▶ output$treemap_14	0	1.6	270
▶ <observer>	0	7.1	230
▶ output$treemap_20	-3.4	1.4	210
▶ output$treemap_18	0	1.5	190
▶ output$linechart_1416	-4.3	3.9	180

Il est facile de remarquer les valeurs extravagantes du temps et de la mémoire de l'*output* 'map_raster1' qui correspond à la carte Leaflet du 2ème onglet affichant des couches rasters (« figure 32 »).

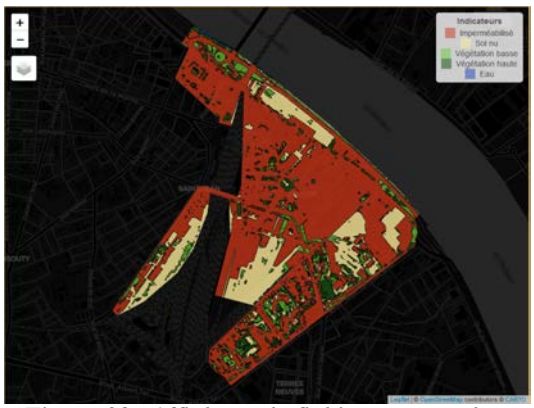


Figure 32: Affichage de fichiers rasters dans Leaflet

Il est question ici d'un problème de taille et de format du fichier en entrée. C'est une problématique à laquelle nous avions déjà été confrontées dans l'application précédente. Malgré avoir réduit la taille des pixels de l'image avec la fonction *aggregate* de la bibliothèque « raster »,

[18] Joe Cheng, « SuperZip example », https://shiny.rstudio.com/gallery/superzip-example.html

l'affichage est lent. Dans RShiny, l'affichage des *outputs* se fait une fois que toute la page est chargée. Il est facile de vérifier que lorsque l'on enlève les couches rasters dans l'affichage, la page se génère sans délai. Le problème vient de ces fichiers en entrée.

Pour pallier ce problème, la solution serait de transformer ces couches en flux WMS. Cela permettrait à l'application de ne pas charger ni de garder en mémoire les rasters. Il faut savoir aussi que l'application ne charge pour l'instant que 2 occupations du sol en raster et qu'elle est supposée en héberger 5, 1 pour chaque millésime. La transformation de couches en flux WMS nécessite un serveur géographique dédié, je reviens sur cette problématique dans la dernière grande partie.

Cela constitue la principale difficulté de l'application. Il faut néanmoins préciser que le script de l'application est long bien qu'il ne soit pas très compliqué à comprendre et qu'il soit composé de beaucoup de répétitions de code du fait des nombreuses conditions dans l'application.

2.c) Application paramétrable

Cette application est un outil souhaité par le pôle satellite du Cerema. En effet, un outil qui est paramétrable permettrait aux agents de réaliser de la dataviz rapidement et sans avoir à connaître le langage en question. Il est entendu ici par paramétrable une application généralisée qui est déjà codée contenant un fichier de paramètres que l'utilisateur devra remplir afin d'y insérer ses propres données.

2.c.1 Fonctionnalités

L'outil doit permettre d'afficher interactivement de la cartographie ainsi que divers graphiques permettant de visualiser la donnée.

Pour guider l'utilisateur dans ce *dashboard*, j'ai intégré dans l'application un module d'aide et d'introduction qui s'affiche lorsque la page est chargée grâce à la bibliothèque rintrojs (« voir figure 33 »). Cette fonctionnalité présente chaque composant du *dashboard*.

Figure 33: Module d'aide Rshiny

Dans la partie serveur, voici les lignes de code qui permettent d'intégrer ce module dans l'application. On définit la boîte de dialogue qui s'affiche à l'ouverture de l'application ainsi que les boutons pour passer à l'interaction suivante ou précédente.

```
######## Boîte s'affichant à l'ouverture de l'application pour guider l'utilisateur
#show intro modal
observeEvent("", {
  showModal(modalDialog(
    h4("Application paramétrable par l'utilisateur qui permet d'afficher une carte interactive Leaflet contenant des polygones.
       La sélection d'un polygone permet de modifier l'affichage du TreeMap. Les autres graphiques permettent de visualiser les données en utilisant
       plusieurs représentations."),
    easyClose = TRUE,
    footer = tagList(
      actionButton(inputId = "intro", label = "Tour d'introduction", icon = icon("info-circle"))
    )
  ))
})

observeEvent(input$intro,{
  removeModal()
})

# show intro tour
observeEvent(input$intro,
             introjs(session, options = list("nextLabel" = "Suivant",
                                             "prevLabel" = "Précédent",
                                             "doneLabel" = "C'est parti."))
)
```

Le texte qui est ensuite affiché dans chaque boîte de dialogue est issu d'un fichier csv « intro » que l'on charge dans le script global.R (« voir figure 34 »).

```
###Module Tour Introduction (ne pas modifier)
intro <- read_csv("intro.csv")
```

step	text
1	Carte leaflet interactive . Choisissez un polygone. Le choix du polygone altère l'affichage des graphiques.
2	Représentation en Treemap . La taille de chaque case est proportionnelle à la valeur que l'utilisateur a choisi en input. Il faut sélectionner un polygone au préalable
3	Représentation en PieChart . La taille de chaque part du camembert est proportionnelle à la valeur que l'utilisateur a choisi en input.
4	Bouton dit "Radio" qui permet de générer soit un BarChart soit un BubbleChart.
5	Représentation en BarChart , Graphique en bâton qui représente les valeurs de distribution d'une variable quantitative. Représentation en BubbleChart . Graphique à bulles qui permet d'afficher 3 dimensions : X, Y et la taille.

Figure 34: Fichier csv pour le module d'aide

Pour choisir ensuite la variable décrite par le module, la fonction 'introbox' définit à quelle étape et quelle ligne du fichier 'intro.csv' la représentation en question correspond.

```
introBox(data.step = 2, data.intro = intro$text[2],
         plotOutput(outputId = "tree_map", height = "300", width = "300"))
```

Concernant le paramétrage, voici un aperçu du script de global.R :

```
############################    JEU DE DONNEES    ############################

## ATTENTION : les données vectorielles doivent être au format 4326 WGS 84 ##
vecteur1 <-  "E:/Stage/App_formation_correction/Data/poly_ocs_hautespyrenees.shp" #indiquer le nom et l'extension du fichier vectoriel


  vecteur1 <- sf::read_sf(vecteur1)
  ident <- rownames(vecteur1)  # création d'un identifiant unique
  vecteur1 <- cbind(id=ident, vecteur1)
  dataframe1 <-  as_tibble(vecteur1) #création d'un dataframe
  # vecteur1 <-  rgdal::spTransform(vecteur1, CRS("+proj=longlat +datum=WGS84"))



# le dataframe sur lequel est basé le dashboard est alors soit la conversion d'un vecteur en dataframe soit de l'import d'un tableau CSV

############################    VALUE BOX    ############################
# box avec valeurs uniques
#box 1 --> nombre de lignes dans le dataframe
nbre_col <- "communes" # (descriptif de la donnée : à quoi correspond une ligne dans le dataframe ?)
#box 2 --> somme d'une colonne du dataframe
label_col <- "superficie" # nom de la colonne à choisir (somme de toutes les lignes)
sum_col <- "superficie (hectares)" # (descriptif de la donnée, label à afficher derrière la donnée)
#box 3 --> somme d'une autre colonne du dataframe
label_col2 <- "population" # nom de la colonne à choisir (somme de toutes les lignes)
sum_col2 <- "habitants" # (descriptif de la donnée, label à afficher derrière la donnée)
```

L'utilisateur doit charger les données et définir ensuite pour chaque représentation les colonnes à sélectionner. Il dispose aussi de plusieurs choix concernant la cartographie interactive :

- une carte avec une couche de polygones
- une carte avec une couche de polygones et un flux WMS
- une carte thématique choroplèthe basée sur une colonne du fichier vecteur : l'utilisateur doit spécifier une colonne du fichier *shape*. Les couleurs seront en fonction des valeurs de cette colonne : cela pose un problème de discrétisation mais j'ai jugé qu'il ne fallait pas aller plus loin dans le paramétrage afin de simplifier l'application. Cela permet juste de discrétiser la couche de polygones afin de différencier les polygones par des couleurs.

L'utilisateur n'a pas la main sur la palette ni la discrétisation mais peut visualiser simplement (mais pas de manière poussée), les différentes catégories ou rangs.

Côté global.R, cela se traduit par :

```
##########################   CARTE LEAFLET   ##########################
# copier coller le type de carte souhaitée dans la variable type avec les différents choix indiqués ci-dessous
type <- "POLYGONE"
#au choix :
# - "WMS_POLYG" (1 flux WMS + 1 couche polygones)
# - "POLYGONE" (1 couche polygones)
# - "CHOROPLETHE" (1 couche polygones avec couleurs basées sur une colonne)
label_vect1 <- "nom" #nom de la colonne à utiliser comme label du polygone Z qui s'affiche en Popup et en passant la souris dessus
id_vect1 <- "insee_com" #identifiant unique d'un polygone



TITRE_MAP1 <- "Communes des Hautes-Pyrénées" #titre du box contenant la carte leaflet

# SI type = WMS_POLYG alors remplir les deux variables suivantes
WMS <- ""   #URL du WMS ou WFS
WMS_layer <- "" #Nom du layer à requêter

# SI type = CHOROPLETHE alors remplir la variable suivante
factor_color <- "S_2016_imp" #nom de la colonne altérant la couleur
```

Côté server.R, cela se traduit par :

```
  output$map_general <- renderLeaflet({
    if (type == "POLYGONE"  ) {
      general_map(vecteur1,vecteur1[[label_vect1]], id_vect1)
    }
    else if (type == "WMS_POLYG"  ) {
      wms_map(vecteur1,wms, wmslayer, vecteur1[[label_vect1]], id_vect1)
    }
    else if (type == "CHOROPLETHE") {
      choropleth_map(vecteur1, vecteur1[[label_vect1]], vecteur1[[factor_color]], id_vect1)
    }
  })
```

L'*output* fait appel à des fonctions qui sont définies dans le fichier fonctions.R. Ce dernier crée des fonctions permettant la création de graphiques et de cartes. Il suffit de le charger en l'appelant dans le fichier global :

```
source('fonctions.R')
source('librairies.R')
```

Le deuxième onglet « Visualisation » permet à l'utilisateur de choisir un type de graphique parmi 4 (densité, point, barre, *boxplot* ou boîte à moustaches) ainsi que les variables (x, y et couleur pour certain graphique). Cela a été réalisé dans l'optique de donner à l'utilisateur plus d'autonomie dans le choix de représentation.

Le troisième onglet « Données » permet de visualiser les données dans un tableau. Il est possible de les filtrer et de télécharger ensuite ce jeu de données filtré en csv.

Enfin, le dernier onglet « A propos » est l'incorporation d'un fichier RMarkDown transformé en HTML pour décrire l'application. Le RMarkDown est un moyen de générer des documents de manière dynamique en mélangeant du texte mis en forme et des résultats produits par du code R.

2.c.2 Limites

L'application a été essayée avec plusieurs jeux de données et semblent fonctionner mais selon moi la dataviz est difficilement paramétrable ou du moins il est difficile de faire une visualisation poussée, élaborée et originale avec en entrée des paramètres modifiables par l'utilisateur. Voilà pour moi les principales limites :

- Les jeux de données peuvent être formatées de différentes manières. En paramétrant l'application, il est nécessaire que les données soient dans un format similaire.
- Ce que l'on souhaite montrer est intrésèquement lié à la donnée et chaque utilisateur va envisager une visualisation différente.

- Les représentations graphiques sont différentes si l'on traite de l'évolution, de la comparaison ou de l'analyse.
- Il est difficile de prévoir tous les cas de figures lorsque l'on développe une telle application.
- Les cartes choroplèthes posent des problèmes de discrétisation.

Pour moi les langages informatiques se détachent des logiciels de dataviz par la liberté que le développeur possède. En effet, la souplesse du code est le principal avantage et permet de créer des outils taillés sur mesure. En paramétrant une application, on perd ces avantages.

3. Le déploiement

La question du déploiement sur un serveur est importante notamment car les applications RShiny ne sont pas convertissables en fichier .HTML visualisable en local. Elles doivent être déployées sinon la seule possibilité de les visualiser est sur RStudio. Pour l'hébergement, nous avons deux solutions : la première est un serveur shinyapp.io que Cerema Méditerranée nous partage et la seconde, c'est l'obtention d'un serveur privé au sein du pôle satellite.

3.a) Serveur Cerema Méditerranée

Le Cerema Méditerranée a adopté la solution de RShiny avec un serveur Shiny.IO : nombre d'applications illimité et optimisation des performances (affichage plus rapide). Néanmoins la licence n'est pas pérenne et la possibilité qu'elle ne soit pas reconduite l'année prochaine existe. Une discussion sur le réseau média d'entreprise du Cerema 'MonCerema' pour pérenniser RShiny a vu le jour mais la Direction des systèmes d'information du Cerema (DSI) n'est pas d'accord. La question est donc en suspens. Cerema Méditerranée plaide pour un serveur Shiny Pro pour un coût de 3000€/an. Concernant la protection des données, il est questionnable le fait qu'un service public utilise un serveur commercial étranger pour l'hébergement d'applications utilisant parfois des données non publiques. En réalité, la question ne s'est pas réellement posée dans l'équipe dans le cadre du stage et des applications développées. La principale problématique a porté sur le format de livraison. J'y reviens dans la sous partie suivante.

Nous avons tout de même profité du serveur et nous avons pu héberger nos applications en ligne. Le *package* 'rsconnect' déploie les applications RShiny sur le service shinyapps.io. Ce dernier est un service de déploiement sur le cloud, il possède une version gratuite et payante et il n'y a aucune installation requise. L'avantage est que tout est paramétré pour héberger une application Shiny cependant nous n'avions pas la main sur le serveur et donc sur les performances et paramètres de celui-ci.

La licence standard qui coûte environ 1000€ par an donne les avantages suivants :

- Un espace avec gestion des accès, une optimisation des performances par rapport à la licence gratuite
- Un nombre illimité d'applications
- 2000 heures actives par mois
- Un support personnalisé par e-mail

Avec cette licence, le Cerema ne dispose toutefois que d'un seul compte administrateur permettant de gérer les applications déployées.

Une licence pro est cependant demandée par les contributeurs de la note portant sur la nécessité d'avoir un serveur de data visualisation. Celle-ci offre plus de garanties fonctionnelles, pour un coût d'environ 3 000 euros par an :

- Des noms de domaines personnalisés
- 10 000 heures actives par mois
- Un partage des comptes administrateurs

Le Commissariat général au développement durable (CGDD) a quant à lui choisi d'adopter cette licence.

3.b) Serveur privé

Dans le cadre du stage, il a été possible de se doter d'un serveur privé pour le déploiement des applications développées sous Dash car le Cerema Sud-Ouest ne disposait d'aucun serveur pour les héberger. L'intégration de l'application Dash a été mise en place par un informaticien du Cerema. Il était question d'intégrer les applications Shiny dans ce serveur mais cela nécessitait l'installation des modules et bibliothèques R et RShiny par cet informaticien qui n'a pas pu le réaliser durant la durée du stage. Je n'ai pas pu alors tester le déploiement des applications sur ce serveur.

La problématique de l'hébergement rejoint celle du livrable. En effet, lorsqu'une application est réalisée et qu'elle doit être transmise au commanditaire, comment celle-ci doit-elle être livrée ? Est-il préférable d'héberger l'application sur un serveur propre au Cerema avec un coût annuel pour le serveur et une pérennité instable ou est-il plus adapté de donner le code source et l'ensemble des données au commanditaire qui devra s'occuper lui-même de l'hébergement et de la maintenance de l'application ? La dernière solution semble être plus cohérente et moins contraignante mais cela reste une question en suspens au sein du pôle satellite du Cerema.

VI. Synthèse et comparaison des solutions : limites, avantages et intérêts

Après avoir manipulé, testé et assimilé les outils, il a été important de comparer les solutions de dataviz et d'avoir un regard critique sur leurs fonctionnalités et leur intérêt au sein du Cerema.

1. Comparaison des fonctionnalités

Cette partie compare les deux bibliothèques Dash et Shiny et analyse leurs avantages et atouts.

1.a) Application sur les ilots de chaleur urbains dans la métropole de Lille

La question de la dataviz au sein du pôle satellite du Cerema a fait l'objet de deux stages. L'un s'est concentré sur RShiny à savoir moi quand l'autre s'est focalisé sur Python Dash. Ce sont les deux bibliothèques qui ont été retenues lors de la phase de bibliographie réalisées en début de stage. Après avoir pris en main les outils et développé des tableaux de bord sur plusieurs jeux de données, l'idée est de comparer ces deux solutions. Pour cela, un *dashboard* sur les îlots de chaleur urbain à Lille a été réalisé tant sous RShiny que sous Python Dash afin d'en comparer les fonctionnalités et les performances. Chacun de ces *dashboards* possèdent exactement les

mêmes données et les mêmes fonctionnalités. La figure 35 présente des captures d'écran : à gauche le *dashboard* réalisé sur RShiny et à droite celui réalisé sur Python Dash.

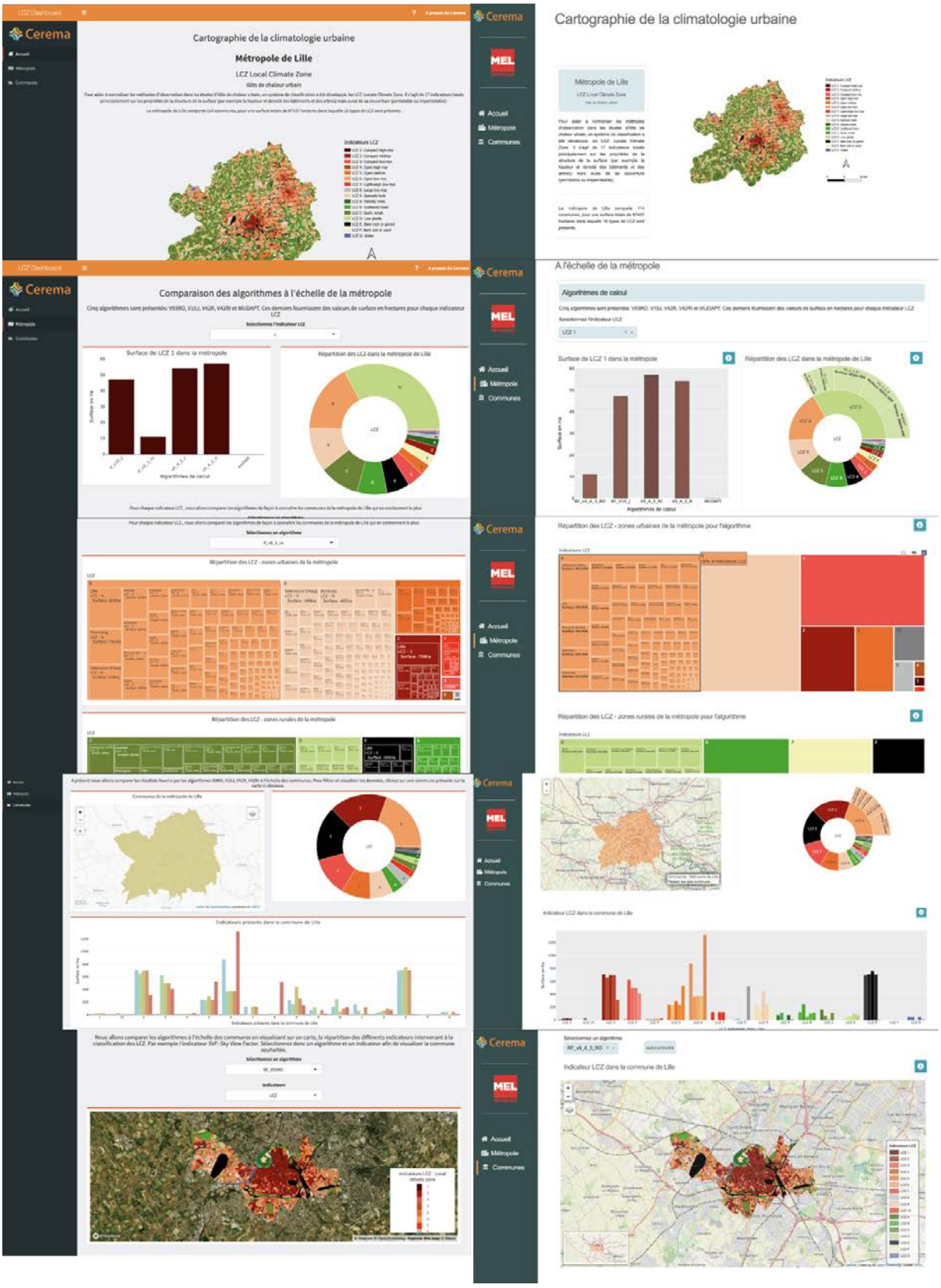


Figure 35 : Planche comparative Dashboard LCZ Shiny (gauche) et Dash (droite)

Les deux applications disposent des mêmes fonctionnalités et il a été possible de faire tout ce qui a été réalisé sous Dash sur RShiny. En termes de fonctionnalité, les applications se valent du fait qu'elles sont basées essentiellement sur la bibliothèque Plotly qui est disponible tant sous R que sous Dash. La différence remarquable est sur la performance. En effet, Python Dash semble mieux gérer les données volumineuses quant à R, il présente des petites latences lorsqu'il faut générer plusieurs graphiques Plotly.

Néanmoins, Leaflet sous Python présente de nombreuses limites comparé à Leaflet sous R. Toutes interactions venant de Leaflet vers Python Dash étaient possibles, mais ce n'était pas le cas dans le sens inverse. En effet, il n'est pas possible d'avoir une pleine interactivité entre la carte et les composantes de Dash. De plus, Leaflet gère uniquement du Geojson en format d'entrée, ce qui nécessite au préalable de convertir les *shapefiles*. Le fichier de 100mo contenant l'ensemble des *Local Climate Zones* (LCZ) ou zones climatiques locales de la métropole ralentissait considérablement l'affichage sur l'API cartographique. Pour pallier à ce problème, une solution a été proposée :

- découpage par communes
- passage en GeoJson et EPSG 4326
- création de fichiers HTML

Nous avons conclu que le *framework* Dash n'est pas assez mature pour les besoins du Cerema. Cet outil a été créé en 2017 et son côté cartographique n'est pas assez développé pour cette étude. En effet sur RShiny, aucune des manipulations préalablement expliquées n'a été nécessaires pour l'affichage des polygones des LCZ. L'utilisation de la bibliothèque MapDeck sous R permet d'afficher rapidement des gros volumes de données. Afin que le chargement se fasse plus rapidement, une intersection est réalisée sur la commune sélectionnée au préalable : on découpe l'ensemble des LCZ selon le polygone de la commune.

Ces deux applications permettent de comparer assez facilement les outils du fait qu'elles utilisent le même jeu de données et adoptent les mêmes représentations. Il est clair que RShiny semble à l'heure actuelle plus abouti et performant du fait de ses fonctionnalités implémentées et des nombreuses bibliothèques disponibles sur R. L'analyse et la manipulation de données spatiales sont bien gérées du fait des nombreux *packages* : Rgdal, SF, Rmapshaper, etc. d'autant plus qu'il existe plusieurs API cartographiques telles que Leaflet, MapBox et des dérivés de celles-ci MapDeck et LeafGl. D'un point de vue extérieur, le résultat est similaire mais d'un point de vue technique, Dash présente beaucoup plus de limites notamment sur la simplicité du code, sa reproductibilité, le temps de développement, etc.

1.b) Tableau des fonctionnalités

L'idée est de montrer dans le tableau 3 les capacités de chaque outil à gérer chacune d'entre elles. En rouge, les fonctionnalités ne sont pas gérées, en orange, la solution présente des limites et en vert elle répond parfaitement aux attentes.

Tableau 3 : Comparaison des fonctionnalités

	Outils	
	R	**Python**
Lecture de fichiers vecteurs	Lecture et manipulation : Rgdal et SF	Lecture *shapefile* : Geopandas
Lecture de fichiers rasters	Lecture et manipulation : Raster	Lecture : module rasterio puis export en jpeg: module Image
Lecture WMS	Affichage Flux WMS (leaflet : addWMSTiles)	Affichage de flux WMS (WMS Tile Layer)
Lecture WFS	Pas de lecture WFS	Pas de lecture WFS
Affichage de graphiques	GGPlot, Plotly	Plotly
Cartographie dynamique	Leaflet / Mapbox (necessite Token) Mapdeck / LeafGl (affichage de données lourdes, pas de réactivité)	Folium, Dash_leaflet (très récent), Mapbox (besoin d'une clé car clé développeur seulement gratuite jusqu'à 5000 requêtes)
Enregistrement de résultats	Plotly permet de sauvegarder les graphiques Fonctionnalité Shiny pour télécharger les données	Sauvegarde des graphes Plotly
Traitement des données	Bibliothèque : dplyr / tidyr	Module Pandas (pour csv, excel) et module Geopandas (pour geojson)
Gros volume de données	Latence à gérer des gros volumes de données dans RShiny car sont stockés en mémoire	Calculs statistiques rapides. Cependant pour la partie cartographique, des limites pour Leaflet en raison du format Geojson
Export en HTML / Déploiement	Déploiement sur serveur shiny.io Fichiers RMarkDown dynamiques en HTML	Pas d'export en HTML. Déploiement sur serveur WSGI (serveur web pour python)
Curseurs altérant l'affichage	SliderInput de Shiny	Slider et RangeSlider fonctionnant très bien entre les composants de Dash. plus laborieux pour créer le lien entre une couche Geojson de Leaflet et les Slider.
Ajout de couches au projet	FileInput : csv, shp, etc.	Dash core componente Upload

<u>Comparaison Leaflet / LeafGl / MapDeck</u>

Pour comparer les performances des différentes bibliothèques de cartographie interactive, j'ai réalisé un simple test d'affichage d'un jeu de données assez lourd : 44 775 polygones pour représenter les LCZ de la métropole de Lille.

- Leaflet : simple test d'affichage des LCZ dans une carte leaflet sous shiny : ça met 50 secondes à s'afficher sans modifier la géométrie. En simplifiant la géométrie, on arrive à 30 secondes.

- LeafGl (« figure 36 ») : avec la couche simplifiée, le temps d'affichage est de 3 à 4 secondes. Très rapide, application et navigation fluide mais moins d'options d'affichage et d'esthétique (bords, couleurs, highlight options, etc.). Sans simplification : 22 secondes. Application qui reste fluide dans l'affichage et la navigation. La problématique peut venir du fait qu'il n'y a pas la possibilité d'avoir d'interactions avec des composantes shiny.

Figure 36: Carte interactive produite avec LeafGl

- MapDeck (« figure 37 ») : affichage très rapide (3 à 5 secondes) mais très peu de fonctionnalités : pas d'interactions avec des composantes shiny.

Figure 37: Carte interactive produite avec MapDeck

MapDeck utilise en réalité MapboxGl (bibliothèque JavaScript pour les cartes interactives) et Deck.Gl (bibliothèque JavaScript qui utilise 'WebGl' pour visualiser de grands ensembles de données). LeafGl utilise lui aussi WebGl pour afficher de larges jeux de données.

Export en HTML

Il n'est pas possible de transformer une application RShiny entièrement en un lien .HTML. Néanmoins, il est possible de créer des rapports interactifs disposant de fonctionnalités dynamiques avec le RMarkDown, un langage de balisage léger[19]. Il existe aussi des modèles graphiques (*templates*) pour créer des tableaux de bord dynamiques en RMarkDown tel que Flexdashboard. Il permet d'organiser les composants du *dashboard* et d'utiliser des widgets HTML pour la visualisation de données. La figure 38 présente un exemple de rapport réalisé en RMarkDown.

Ceci peut être intéressant pour le service car ils sont familiers avec le MarkDown sous python. Cela reste assez similaire sauf que les parties de code sont en R. Un script fonctions.R a été réalisé par mes soins afin qu'ils aient juste à appeler les fonctions et à remplir les arguments pour créer des graphiques sous Plotly et des cartes sous Leaflet.

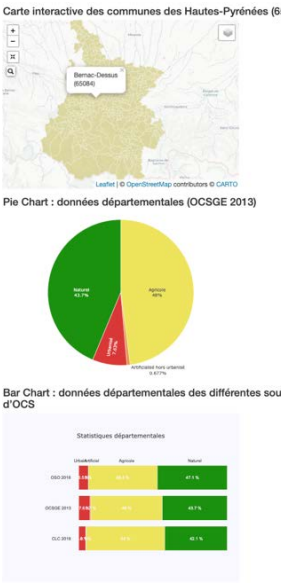


Figure 38: Rapport interactif RMarkDown

1.c) *Comparaison des atouts et des limites*

RShiny est un outil adapté pour réaliser de la dataviz. Il existe depuis maintenant 10 ans et est en constante évolution depuis. Cette bibliothèque permet de générer facilement et rapidement des *dashboards* sans avoir à coder ou même connaître les langages de programmation web. Le *framework* de RShiny est simple à comprendre et à mettre en place, sa documentation est très riche et sa communauté active, de plus il existe de nombreuses applications RShiny avec le code source à disposition. Par rapport aux missions qui m'ont été données, j'ai réussi à développer la quasi-totalité des critères souhaités (graphique interactif, affichage raster, vecteur, flux, etc.) si ce n'est l'affichage de Web Feature Service (WFS). En effet, Leaflet ne permet pas d'afficher de flux WFS qui a l'avantage de pouvoir être dynamique et interactif avec l'utilisateur. La seule solution était de rentrer le flux WFS dans une variable afin de l'afficher spatialement sauf que cela revient exactement à la même situation que d'avoir un fichier en local.

Le problème du WFS tentait de répondre à une problématique plus globale, celle de la latence de l'application et de la difficulté à gérer de larges et volumineuses données. En effet, les

[19] « Introduction to interactive documents ». *Shiny from RStudio.* [En ligne] http://shiny.rstudio-staging.com/articles/interactive-docs.html.

limites de RShiny se sont rapidement vues lorsqu'il s'agissait d'afficher plusieurs vecteurs ou plusieurs rasters. Le temps d'affichage était long parfois très long ce qui n'est pas souhaitable en dataviz. Des bibliothèques comme LeafGl ou MapDeck permettent un affichage très rapide et très fluide de milliers de polygones mais contrainte en termes d'esthétisme, de fonctionnalités et de réactivité avec les composantes de Shiny.

Il n'existe pas encore de bibliothèques complètes pour réaliser des *StoryMaps* (comme celles d'ArcGis par exemple) mais il est possible de créer des pages avec des animations pour rendre la visualisation dynamique et plus agréable pour l'utilisateur. J'ai par ailleurs intégré des fichiers rasters dans des cartes Leaflet, ce qui a rendu l'affichage de la page *storytelling* beaucoup plus lent malgré la simplification de l'image (réduction de la taille des pixels : de 61mo à 700ko). Une solution serait alors de déposer sur Cerema.data (plateforme open data du Cerema) les fichiers rasters et de les appeler par des flux WMS. L'affichage serait probablement beaucoup plus rapide. Je reviens dans la partie suivante sur la solution que peut offrir ces flux.

Tableau 4 : Limites et avantages de Shiny et Dash

	R Shiny	Python Dash
Atouts	**Popularité :** Communauté large et documentation fournie. **Performance :** Adapté pour la DataViz et traitement de données. **Fonctionnalité :** *Framework* Shiny bien fourni : fonctionnalités permettant de l'interactivité et de la réactivité. Possibilité de sécuriser l'application par un identifiant et mot-de-passe **Technique :** Pas de compétences CSS/JS/HTML requises mais possibilité de modifier le CSS si besoin.	**Popularité :** Connu du service, intégration dans la chaîne. Communauté large et très active. **Performance** : Adapté pour la dataviz et traitement de données **Fonctionnalité :** *Framework* bien fourni qui permet un dynamisme entre les objets très large. Système d'authentification intégré et possibilité d'intégrer du code html (.html avec Javascript etc) également. **Technique:** Pas de connaissances obligatoires en HTML5 , des bases en CSS sont un plus.
Limites	**Popularité :** Faible popularité interne au service. **Performance :** Faible capacité à gérer de larges données/rasters : affichage qui peut être très long. **Fonctionnalité :** Pas de flux WFS sur Leaflet. Pas de bibliothèque vraiment implémentée pour réaliser du scrollytelling notamment concernant la cartographie. **Financier :** Serveur payant mais possibilité d'héberger 5 applications gratuitement sur shinyapp.io avec une limite de 25h actives (pas de protection)	**Performance :** Côté cartographique, c'est assez limité en termes de fonctionnalités avec Dash-leaflet. Difficulté à gérer spatialement des données très lourdes. **Fonctionnalité :** Le module Folium permet de créer facilement des cartes Leaflet avec toutes les fonctionnalités proposées par Leaflet JavaScript mais aucune interaction avec les composants de Dash (seulement de la visualisation et interaction à l'intérieur de la carte). Dash-leaflet très jeune encore (3 mois) **Financier:** Serveur payant mais possibilité d'héberger 5 applications maximum avec Heroku gratuitement

Dans le contexte du Cerema et du pôle satellite, la solution RShiny est intéressante car elle offre la quasi-totalité des fonctionnalités souhaitées dans une dataviz. Seulement quelques bibliothèques sont nécessaires telles que Leaflet ou Plotly et la prise en main du *package* Shiny est assez rapide. Néanmoins cela requiert une certaine formation ou un temps d'apprentissage à ne pas négliger d'autant plus qu'aucun des agents n'est familier avec le langage R. De par mes présentations devant le service des outils développés et des formations RShiny que j'ai réalisé devant des agents du Cerema, l'idée fut de partager les compétences acquises afin qu'ils aient quelques clés en main pour pouvoir développer eux-mêmes des outils. J'ai débuté le stage sans réelles compétences en RShiny et en dataviz, j'ai tout de même manipulé l'outil pendant de

nombreuses heures avant d'être suffisamment à l'aise pour développer rapidement des visualisations.

Dans le cadre du Cerema, cela reste une solution qui est difficile à adopter pour des chargés d'étude car elle nécessite du temps. La solution d'un logiciel payant serait alors une solution potentiellement intéressante notamment sur le temps gagné à manier l'outil malgré le coût de la licence ou alors d'avoir un développeur dans l'équipe disposant des compétences adéquates. La note sur le serveur de dataviz posté sur l'intranet de l'entreprise MonCerema propose une double solution selon le niveau de compétences des utilisateurs et d'exigence fonctionnelle des projets : RShiny pour les utilisateurs avancés et une solution payante comme ESRI *OpenDashboard* ou Tableau pour la réalisation de tableaux de bord de manière intuitive et simple.

2. L'intérêt des services web dans la DataViz

Tout au long de ce stage et du développement des applications, un problème a été récurrent sur RShiny : la latence et la performance de l'outil. C'est un problème qui nous a empêché d'insérer certaines visualisations dans des applications du fait du temps d'attente pour afficher ou des déconnexions fréquentes. Comme il est dit dans les parties précédentes, cela vient surtout de la taille et des formats des fichiers en entrée. Une solution intéressante et envisageable serait de passer par des services web tels que le *Web Map Service* (WMS), un standard de l'Open Geospatial Consortium (OGC). C'est un moyen d'obtenir des cartes/données géoréférencées à partir d'un serveur de données cartographiques. Il produit des cartes sous la forme d'une image. Ci-dessous les principales requêtes possibles :

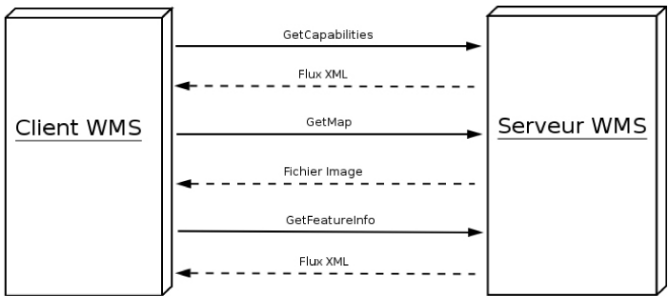


Figure 39: Requête WMS

Source : https://georezo.net/wiki/main/standards/wms

GetCapabilities : retourne les métadonnées du service (couches proposées, projections associées, auteur…)
GetMap : retourne une carte (généralement dans un format d'image) selon les paramètres demandés.
GetFeatureInfo : retourne les informations sur un objet représenté sur la carte[20].

L'idée est alors de remplacer les fichiers rasters ou vecteurs volumineux qui n'ont pas besoin d'être interactifs par des flux WMS. L'application ne stocke pas ces flux, elle les requête directement d'un serveur cartographique ce qui ne fait pas ralentir l'outil.

20 "WMS",Géorezo, https://georezo.net/wiki/main/standards/wms

Dans le cas de l'application OCS, cela nous permettrait d'afficher les occupations du sol (OSO, OCSGE, CLC) au niveau de la commune, sans faire ralentir l'application.

Pour Euratlantique, grâce à une clé IGN qui fonctionne par référer (uniquement valable sur une URL spécifique), nous avons pu bénéficier des flux d'imageries satellites à haute résolution mis à disposition par l'IGN : Pléiades, orthophotos, etc. et cela sans réduire les performances de l'application.

Autre fonctionnalité qui serait intéressante mais qui n'est pas permise par Leaflet est l'appel de flux *Web Feature Service* (WFS). Le WFS permet d'accéder aux objets vecteurs de la table. Cette capacité n'est pas permise ni par Leaflet ni par les autres services de cartographies interactives telles que Mapbox.

Et enfin dernier type de flux qui serait intéressant mais qui n'est pas permis par Leaflet est le *Web Map Tile Service* (WMTS) qui fournit des mosaïques (généralement de taille 256 × 256 pixels), tandis que le WMS délivre une image par requête. Le principal avantage des mosaïques est qu'elles peuvent être pré-rendues côté serveur et mises en cache côté client. Cela réduira le temps d'attente pour les données et la bande passante. Les tuiles vectorielles sont servies en découpant les géométries dans la *bounding box* des vignettes, puis simplifiées pour correspondre au niveau de zoom afin d'éviter une complexité inutile à des niveaux de zoom inférieurs. Ces géométries et caractéristiques sont également traitées pour faciliter le style. C'est une solution intéressante car les flux sont moins lourds et il est possible de régler le style en local tout en utilisant des fonctions d'affichage rapide comme WebGl mais cela nécessite un serveur approprié[21].

Les services Web ont un réel intérêt dans les outils développés. De plus, le Cerema dispose de solutions pour cataloguer et héberger la donnée : Prodige et CeremaData. CeremaData est une plateforme open data et met à disposition du public des données numériques produites par le Cerema et ses partenaires.[22] Prodige est quant à lui un logiciel qui permet de cataloguer des données, de créer les métadonnées associées, de produire des services au bénéfice des partenaires et du grand public.[23] Il est alors possible de déverser de la donnée sur CeremaData grâce à Prodige et d'en faire des flux WMS ce qui serait la solution la plus intéressante pour résoudre les divers problèmes rencontrés. Néanmoins cette solution n'a pas pu être exploité durant mon stage car cela ne rentrait pas dans mes missions.

3. Essai d'un logiciel nécessitant une licence : Tableau

Tableau est un logiciel payant permettant de faire de la dataviz. L'idée est de se rendre compte rapidement, après s'être bien approprié le sujet, la manière de faire, du potentiel, des capacités et des limites d'un tel outil. Après avoir testé des langages de programmation, le but de ce travail est de prendre connaissance d'autres options existantes bien qu'elles soient payantes. En effet, les limites des langages de programmation cités auparavant sont la nécessité de connaître le langage en question et de savoir le manipuler. Un logiciel comme celui-ci permet d'avoir aucune

[21] SIG : Mettre en place des tuiles vectorielles. *Makina Corpus.* [En ligne] https://makina-corpus.com/blog/metier/2020/sig-mettre-en-place-des-tuiles-vectorielles.

[22] **Cerema.** [En ligne] https://www.cerema.fr/fr/actualites/ceremadata-plateforme-open-data-du-cerema-est-ligne

[23] **Cerema.** [En ligne] https://prodige.cerema.fr/

connaissance en la matière et de tout de même réussir à créer de la dataviz[24]. Les gains peuvent être réalisés en termes de temps passé à développer une application et d'investissement à s'approprier le logiciel.

Voici un aperçu de quelques manipulations réalisées avec la version gratuite de Tableau :

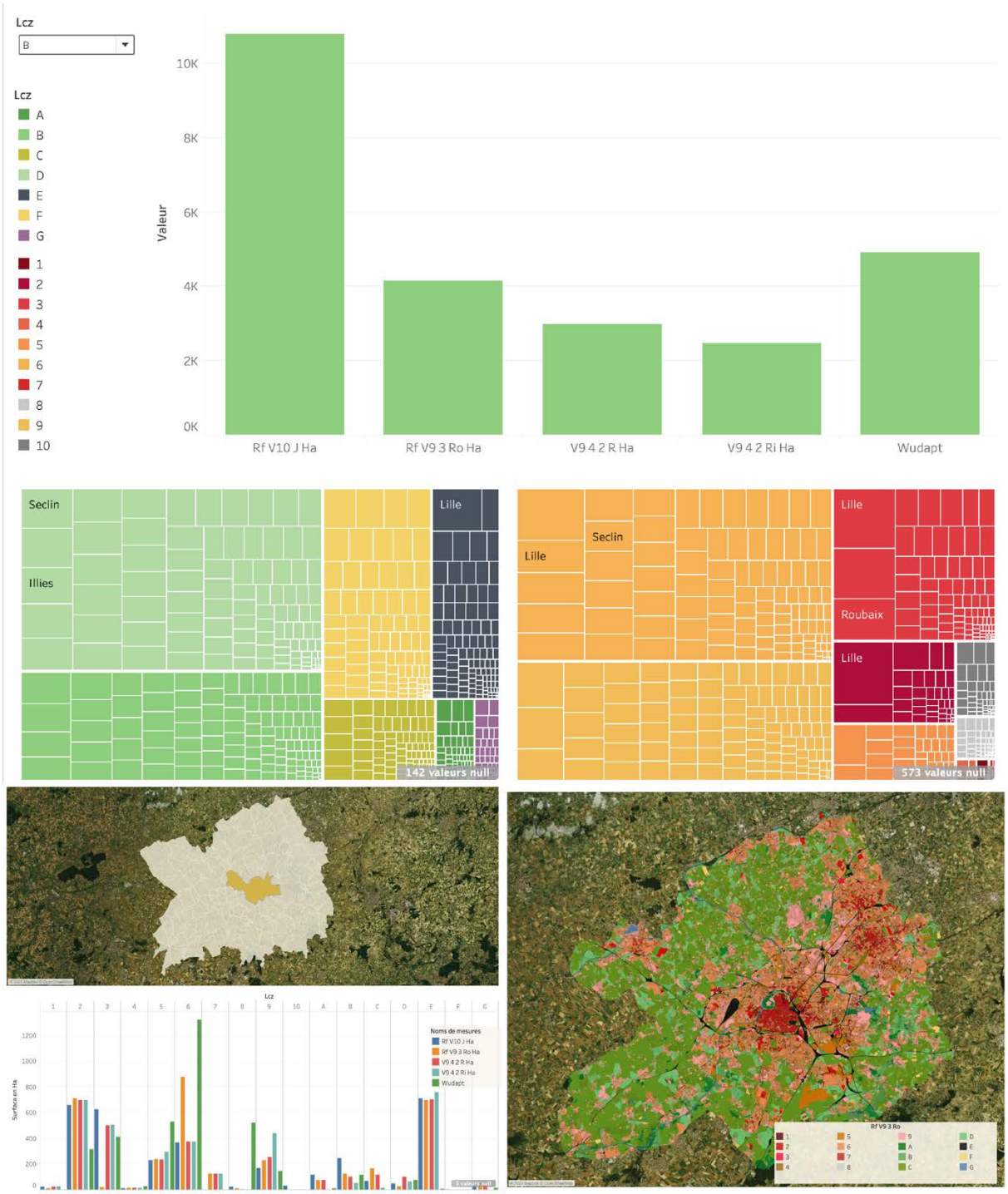


Figure 40 : Planche Dashboard LCZ - Tableau BI

[24] « Michelle Malizia Catalano, Porcia Vaughn & Joshua Been. Using Maps to Promote Data-Driven Decision-Making: One Library's Experience in Data Visualization Instruction », *Medical Reference Services Quarterly,* 2017

Les fonctionnalités sont nombreuses. L'aspect spatial n'est pas très poussé mais les fonctionnalités existantes permettent déjà de nombreuses visualisations. Le glisser-déposer facilite grandement la gestion des données et le logiciel est dans l'ensemble assez intuitif.

Concernant le déploiement et l'export de l'application, il est possible d'enregistrer les tableaux de bord en version figée via PowerPoint sinon l'unique manière de déployer les applications est d'utiliser Tableau Server qui est un service payant de Tableau.

Livrable

Le stage a fait l'objet de plusieurs livrables :

- Bibliographie détaillée des outils existants et synthèse des besoins de l'équipe
- Guide d'utilisation de R et de Shiny : présentation du fonctionnement du *package* avec un exemple d'application très simple.
- Documentation de l'application OCS : présentation du code, des singularités et spécificités et guide de maintenance de l'application.
- Documentation de l'application Euratlantique : présentation du code, des spécificités et guide de maintenance de l'application : l'application et la documentation permettent aux agents de mettre à jour l'application c'est-à-dire de changer de périmètre d'étude, d'ajouter les données des millésimes manquants, de modifier les flux WMS, de compléter le texte, etc.
- Documentation de l'application paramétrable : présentation du code et du paramétrage de l'application par l'utilisateur.
- 3 applications hébergées sur serveur : Euratlantique, l'occupation du sol en Hautes-Pyrénées et LCZ dans la métropole de Lille.
- L'ensemble des scripts et données utilisées dans le développement de ces applications
- Un rapport général reprenant l'ensemble des travaux du stage.

En plus de ces documents, j'ai réalisé deux présentations orales devant des agents du Cerema en fin de stage pour présenter mes travaux à savoir la bibliographie, le recueil du besoin et le développement des applications. L'idée était d'avoir un regard critique sur le travail réalisé et de discuter de l'intérêt de la dataviz et de la possible incorporation des outils dans les travaux des agents du Cerema.

Enfin, après chaque présentation orale, une session de formation à RShiny fut réalisée par mes soins. L'idée fut de montrer le fonctionnement de R et de Shiny et de décrire, avec l'aide d'une application très simple développée pour la formation, le script. Cela a été réalisé dans le but de leur partager la manière dont l'interface graphique est créée, la réactivité est gérée et les objets affichés.

Planning effectif (Gantt)

Voici le planning de mon stage, c'est-à-dire la manière dont les différentes phases se sont déroulées dans le temps :

- En vert : le travail préalable : bibliographie et recueil du besoin
- En jaune : la phase de développement
- En bleu : la documentation des travaux et présentation des résultats

Figure 41: Diagramme de Gantt

Conclusion et perspectives

L'objectif du stage fut de développer des outils de géo et data visualisation pour valoriser les résultats de traitement d'imageries satellitaires. La phase préalable à la production de ces outils a été très importante du fait que la thématique de la dataviz est relativement nouvelle au sein du pôle satellite de la DALETT. En effet, il a fallu, de par la bibliographie et les entretiens, déterminer les solutions les plus pertinentes dans le cadre du Cerema par rapport aux fonctionnalités souhaitées et aux critères établis. C'est dans cette étape préliminaire que deux *frameworks* ont été retenus : RShiny et Python Dash, des *packages* permettant la création de tableaux de bord. Ils ont été sélectionnés selon des critères communs tels que la gratuité d'utilisation, l'open-source et la large documentation et communauté.

La représentation et la valorisation de données issues d'imageries spatiales a porté sur plusieurs thématiques et projets. Cela a permis de déceler les avantages et les limites des méthodes choisies, de dresser un panorama des fonctionnalités existantes et de tirer des conclusions générales.

La première application développée sous RShiny par mes soins porte sur l'occupation du sol dans les communes des Hautes-Pyrénées. L'objectif du projet est de mener un panorama des données de consommation d'espace en convertissant en 4 indicateurs plusieurs sources d'OCS à savoir CLC, l'OSO, l'OCSGE et les fichiers fonciers pour en calculer des statistiques communales. L'objectif de la dataviz développée est de représenter, de comparer et visualiser l'évolution selon plusieurs sources et millésimes. L'idée est de donner une représentation à des données tabulaires. L'outil permet alors à l'utilisateur d'afficher rapidement et facilement des données à la commune, au département, selon un millésime et une source. Il est apparu qu'aucune base de données ne mesure les mêmes valeurs à une date donnée, ni sur les dynamiques

d'évolution où des tendances contraires peuvent être observées. Cela était remarquable dans les métadonnées, mais la dataviz permet d'accentuer ces différences.

La deuxième application développée sur RShiny tente d'explorer d'autres fonctionnalités sur un autre projet : Bordeaux-Euratlantique. La mission du Cerema est de produire des indicateurs de suivi d'aménagements (végétation, surface imperméable et perméable) à partir de données satellitaires à haute-résolution du satellite Pléiades. Dans cette prestation, un outil de dataviz a été proposé. Par cet outil, l'objectif est de visualiser facilement les changements et évolutions de ces indicateurs. Pour ce faire, j'ai opté pour un visualiseur essentiellement spatial avec des sélecteurs des indicateurs, des années, etc. L'utilisateur a davantage de possibilités afin d'altérer l'affichage et de visualiser ce qu'il souhaite. Au regard des nombreux projets éparpillés sur le secteur, différents organismes ont critiqué et montré leurs inquiétudes quant à la forte imperméabilisation que cela pouvait représenter. La dataviz a ici un réel intérêt tout d'abord pour l'EPA Bordeaux-Euratlantique afin de suivre ces indicateurs dans le secteur d'aménagement, mais il est imaginable que cela pourra être un moyen de communication et de vérification auprès des financeurs ou lors de réunions publiques par exemple.

Ces applications ont permis de déceler les limites et les avantages de RShiny. Le principal problème rencontré fut celui de la latence de l'application et de la gestion des gros volumes de données. Hormis cela, le *package* gère plutôt bien l'ensemble des fonctionnalités retenues préalablement. La comparaison avec Dash par le biais d'une application sur les LCZ de la métropole de Lille a permis de mettre en avant le fait que Shiny est une solution pour le moment bien plus avancée et aboutie que Python Dash. Pour pallier aux problèmes rencontrés, la solution pour alléger les applications et augmenter les performances, serait de requêter des flux WMS d'un serveur tel que la plateforme open-source CeremaData qui offre la possibilité de pousser des données vecteurs et rasters et d'en faire des flux web. J'ai créé pour l'ensemble des outils développés des documentations pour les utiliser, les adapter et les mettre à jour. Un guide d'utilisation RShiny a aussi été rédigé afin de comprendre le fonctionnement du *package*, mais aussi de permettre à l'équipe de créer une application très simple.

RShiny semble être une solution adaptée en termes de fonctionnalités par rapport aux besoins du pôle satellite du Cerema. Néanmoins, cela demande un certain temps de développement et de certaines compétences en R. Bien qu'une application paramétrable par l'utilisateur ait été développée ce qui veut dire qu'il n'a besoin que de configurer quelques paramètres pour avoir en sortie un *dashboard*, cela n'est pas assez abouti pour dépendre de celle-ci. En effet, il est selon moi difficile à développer une application qui réponde aux besoins des utilisateurs. La visualisation et donc le script s'altère en fonction du jeu de données, du public visé et de ce que l'on souhaite montrer. Il est difficile de prendre en compte tous ces critères. Dans le contexte du pôle satellite du Cerema, les principales limites que présentent RShiny sont en réalité dans le fait qu'il faut développer l'application et que pour l'instant aucune personne ne dispose des compétences adéquates. Les solutions de dataviz payantes présentent peut-être alors une solution plus adéquate à l'environnement interne du Cerema. Une double offre serait aussi une alternative intéressante en fonction du niveau des utilisateurs et des exigences des projets.

À la suite de ce stage, j'ai permis d'explorer la discipline de la dataviz et de porter à connaissance des agents les intérêts et les limites des outils. Le Cerema ne disposera pas à mon départ d'agents pleinement formés pour réaliser de la dataviz que cela soit sur RShiny ou Python Dash néanmoins, ils ont les connaissances théoriques et un bagage technique via les scripts pour

décider d'une solution à adopter. Le Cerema va prochainement se doter de licences ArcGis, la solution des dashboards et *storymaps* de ce logiciel est peut-être alors la solution pour des agents sans véritables compétences en dataviz. Des agents du Cerema souhaite l'achat d'un serveur pro Shiny ce qui poussera probablement, si achat fait, à réaliser des formations à l'outil en interne. Concernant les outils développées par mes travaux, ils seront récupérés par les agents du pôle satellite et adaptés à l'évolution du projet et de la commande grâce aux documentations que j'ai réalisées. Ces dernières permettent de mettre à jour les données, modifier les flux WMS, rajouter du texte et changer de zone d'étude lorsque les résultats des traitements d'images satellites auront été réalisés par les agents. En définitive, les documentations vont leur permettre de récupérer les travaux, de les améliorer/compléter et de les livrer aux commanditaires par la suite. Concernant plus globalement la *data visualisation* au sein du pôle satellite, cela dépend des solutions adoptées au niveau régional voir national, de si les licences ArcGis disposent de fonctionnalités de tableau de bord et de *storymap* ainsi que de la volonté des agents à se former à des outils de dataviz.

Ce stage m'a permis d'acquérir beaucoup de nouvelles compétences. En effet, la dataviz m'était une thématique assez méconnue et cela m'a donné l'opportunité de me familiariser avec les outils et d'apprendre la représentation et la valorisation de la donnée. Cela m'a permis de mettre en œuvre les compétences acquises en R et à les conforter. Plus spécifiquement, l'expérience m'a donné l'opportunité de découvrir le *package* Shiny et toutes les bibliothèques utilisées dans les applications. Le stage m'a fait découvrir le fonctionnement et la gestion d'une structure de la fonction publique étatique et de connaître plus en détail les travaux du Cerema notamment du pôle satellite. Du fait du contexte exceptionnel lié à la crise sanitaire, j'ai acquis beaucoup d'autonomie, mais j'ai aussi développé ma capacité à communiquer et à travailler en équipe. Enfin, ce stage m'a permis de développer des compétences en data et géo visualisation, domaine que nous n'avons pas réellement abordé lors de la formation. Cela complète parfaitement, de par l'apprentissage professionnel et technique, la formation reçue à SIGMA, mais cela conforte surtout la capitalisation de connaissances et de compétences de plusieurs UE du premier semestre. Ce fut une expérience riche, formatrice et constructive qui m'a permis d'avoir un aperçu concret du fonctionnement d'un établissement public. Le Cerema ne disposant pas de réelles solutions de dataviz, je suis fier d'avoir pu contribuer à l'exploration de cette discipline et d'avoir pu montrer l'intérêt qu'elle peut avoir dans leurs projets.

Table des matières

.Annexe

Annexe 1 : Grille d'entretien

Grille d'entretien pour la réalisation d'un outil de data visualisation au sein du Cerema. L'enquête vise à recueillir et comprendre les besoins de l'équipe en termes de dataviz.

Connaissances en Data Visualisation ?	
Quels types de données représenter ? (format)	
Dans quel but et pour qui ? (communication interne, partenaires, etc.)	
Qu'est-ce que l'on veut représenter ? (Corrélation, différence, évolution, etc.) Quel est le message à communiquer ?	
Cas d'application sur un projet spécifique ?	
Quelle représentation adopter ? (*dashboard*, carte interactive, infographie, etc.)	
Avec quels outils ? (Python, R, JavaScript, etc.)	
Livrable sous quel format ? (PDF, lien HMTL, etc.)	
Est-ce que la dataviz est pertinente dans votre travail ? Seriez-vous amené à l'utiliser ?	
Si oui, comment visualiser vous l'outil ? (insertion de données brutes, modification du code, glisser-déposer, etc.)	

.Bibliographie et webographie

1. Cerema. [En ligne] https://prodige.cerema.fr/ .
2. Georezo. [En ligne] https://georezo.net/wiki/main/standards/wms.
3. Cerema. [En ligne] https://www.cerema.fr/fr/actualites/ceremadata-plateforme-open-data-du-cerema-est-ligne .
4. RStudio. [En ligne] https://rstudio.github.io/profvis/.
5. Cerema. Application « Observatoire des marchés immobiliers ». [En ligne] http://doc-datafoncier.cerema.fr/dashboard_dvf/.
6. Application « RShiny : Cartofriches ». [En ligne] https://cartofriches.cerema.fr/cartofriches/
7. Frédéric Berlioz, Christophe Badol, Antoine Lemot, Nicolas Pelé, Mathieu Rajerison, Frédéric Lasseron, Perrine Rutkowski, Antoine Herman, Bertrand Leroux. *Note sur le besoin d'un serveur de Data Visualisation* . Cerema. 03/04/2020.
8. Sidonie, Christophe. "La géovisualisation kezako ?" . *LeMonde.* [En ligne] 18 11 2019. https://www.lemonde.fr/blog/binaire/2019/11/18/la-geovisualisation-kezako/.
9. Cerema. [En ligne] https://www.cerema.fr/fr/centre-ressources/newsletters/signture/signture-69-artificialisation-sols-sa-mesure/evaluation-qualite-donnees-classification-occupation-du-sol.
10. Lagnel, Jean-Marie. *Manuel de Data Visualisation.* s.l. : Dunod, 2017.
11. Gallery. *Shiny.RStudio.* [En ligne] https://shiny.rstudio.com/gallery/.
12. Michelle Malizia Catalano, Porcia Vaughn & Joshua Been. "Using Maps to Promote Data-Driven Decision-Making: One Library's Experience in Data Visualization Instruction". *Medical Reference Services Quarterly.* 2017.
13. *Webinar #1 DataVIz & Best Practice.* DataScientest. 06/05/2020.
14. "SIG : Mettre en place des tuiles vectorielles". *Makina Corpus.* [En ligne] https://makina-corpus.com/blog/metier/2020/sig-mettre-en-place-des-tuiles-vectorielles.
15. Yau, Nathan. *Data Visualisation, de l'extraction des données à leur représentation graphique.* s.l. : Eyrolles, 2011.
16. Terranis. "Conception d'un observatoire de la végétation urbaine". [En ligne] https://myterranis.maps.arcgis.com/apps/MapSeries/index.html?appid=36e83f66e1204aa0aa9dd851769fbcf3.
17. "Synthèse des indicateurs "Biodiversité" liés à la végétation urbaine" . [En ligne] https://myterranis.maps.arcgis.com/apps/MapSeries/index.html?appid=36e83f66e1204aa0aa9dd851769fbcf3.
18. Antoine Moriceau, Julie Laforêt et Corentin Rey. "Un exemple de géovisualisation de phénomène temporel avec la bibliothèque D3.js". *Mappemonde.* juillet 2016, Vol. Numéro 118.
19. Jégou, Laurent. Tutoriel sur la représentation graphique statistique avec D3. [En ligne] https://ljegou.github.io/d3sigma/.
20. *Toulouse DataViz.* [En ligne] http://toulouse-dataviz.fr/.
21. "Introduction to interactive documents". *Shiny from RStudio.* [En ligne] http://shiny.rstudio-staging.com/articles/interactive-docs.html.
22. Issarane, Hausmane. "Visualisation des données via Python : Les packages à connaître". *Le DataScientist.* [En ligne] 2019. https://le-datascientist.fr/visualisation-des-donnees-python.

.Table des tableaux

.Table des illustrations