Du laboratoire à Jupyter : La trajectoire d'un instrument logiciel libre de la science ouverte

Emilien Schultz (SESSTIM)

Cet article est une première version de l'histoire du projet Jupyter du 30/11/2023. Il a pour vocation d'initier une réflexion plus collective sur l'histoire des notebooks, qui se poursuit autour du dépôt https://github.com/emilienschultz/history_of_jupyter.

Résumé

Les outils informatiques, des scripts aux logiciels, sont devenus indispensables à la conduite de la recherche contemporaine. Si l'étude des instruments scientifiques a fait l'objet d'une ample littérature des études sociales sur les sciences, de même que les aspects liés aux données numériques, les logiciels sont étrangement absents. Cet article analyse le succès mondial des notebooks computationnels Jupyter pour contribuer à une étude sociale des logiciels sensible aux pratiques des outils numériques en science. Le projet Jupyter, initié en 2001 par un doctorant en physique intéressé par la programmation scientifique a progressivement évolué vers une infrastructure de la science des données mondiale. Je montre l'intérêt de considérer en détails la trajectoire de cet outil en identifiant les conditions toujours négociées du passage d'usages locaux à un rôle d'instrument générique, amenant progressivement son intégration dans une diversité de services au-delà des communautés scientifiques. La diffusion des notebooks computationnels a ainsi opéré non seulement une convergence entre code ouvert et science ouverte, mais aussi entre les pratiques de chercheurs et celles de développeurs informatique, amenant à la stabilisation d'une infrastructure spécifique pour le traitement des données.

Remerciements

Pierre Poulain, Léo Mignot et Mariannig Le Béchec pour leur relecture du manuscrit et les suggestions ; Célya Guson-Daniel pour les réflexions sur les notebooks, ainsi que le GT Notebooks. Matthias Bussonnier pour les échanges depuis de nombreuses années et la possibilité de fréquenter de loin en loin le projet Jupyter. Tous les contributeurs du projet Jupyter.

Introduction

Dans une tribune restée célèbre, Marc Andreessen, figure importante de la Silicon Valley, annonçait que le logiciel était en train de dévorer le monde[1]. S'il fait la liste des secteurs transformés par l'arrivée de nouveaux acteurs du logiciel, il ne mentionne pas la recherche scientifique. Or, comme le remarque un *op-ed* dans Nature, « *behind every great scientific finding of the modern age, there is a computer* » (Perkel, 2021). Cette évolution est à la fois externe et interne : externe, dans la mesure où le développement des technologies numériques et leur commercialisation dépendent largement de logiques éloignées du monde académique ; interne car nombre des idées fondatrices déterminantes de la numérisation de la société viennent de la recherche, ne serait-ce qu'internet ou encore de nombreux langages de programmation[2].

L'extension du numérique dans la société a été scrutée par les sciences sociales (Cardon, 2019), le secteur de la recherche n'y faisant pas défaut. Les transformations autour de la communication numérique des chercheurs que ce soit en termes de publication scientifique (Delfanti, 2016) ou sur les médias sociaux (Desrochers et al., 2018) ont ainsi largement attiré l'attention. Cependant, la rencontre entre STS du numérique et sociologie des pratiques scientifiques reste encore ténue. Cela est d'autant plus surprenant que les STS ont été largement impliquées dans l'émergence des

1 https://a16z.com/2011/08/20/why-software-is-eating-the-world/

2 Pour ne donner que quelques exemples : UNIX est d'abord pensé comme un modèle pour l'enseignement (Kelty, 2008) ; promoteur du logiciel libre, Richard Stallman mène son activité au sein du IA Lab du MIT (Broca et al.,

« *software studies* » ou encore les « *critical code studies* » (Méadel et Sire, 2017), qui ont attiré l'attention sur les aspects matériels liés au numérique. Ces travaux soulignent par exemple les méandres du code et ses conditions d'écriture (Ermoshina, 2017). Les mondes sociaux à l'origine de la production des logiciels ont ainsi pu être détaillés, que ce soit la dense ethnographie de la Silicon Valley (Alexandre, 2023) ou les professionnels de la programmation qui développent et utilisent le logiciel de versionnement Git (Alcaras, 2022). Dans un ouvrage de référence, Christopher Kelty propose ainsi une anthropologie des communautés qui se constituent autour du mouvement du libre pour produire les logiciels iconiques comme EMACS ou GNU-Linux. Pour autant, ces travaux sur les logiciels tendent à privilégier les approches sur les enjeux économiques ou les communautés spécifiques militantes comme les mouvements hackers, et se sont finalement peu concentrés sur les usages du monde scientifique, quand bien même de nombreux universitaires en font partie. Le lien entre numérique et recherche est davantage présent dans les « *information infrastructure studies* » (Bowker et al., 2010), notamment via l'entrée par les données (Leonelli et Tempini, 2022), mettant notamment en évidence l'importance des dispositifs matériels comme les serveurs ou le travail manuel impliqué dans le travail de nettoyage, ou encore l'effet des infrastructures logiciels sur l'activité (Bowker et Star, 1999 ; Vertesi, 2019 ; Plantin et al., 2018). En se concentrant sur les nomenclatures ou les conséquences sur la production des savoirs, les logiciels eux-mêmes sont peu abordés.

Or, ces logiciels sont au cœur de l'équipement des chercheurs. Sensibles aux conditions pratiques de la production des connaissances, l'histoire et la sociologie des sciences et des techniques, et plus généralement les études sur les sciences (STS), ont largement souligné l'importance des instruments dans la production des connaissances (Lamy, 2022). Certains sont devenus emblématiques de la science moderne, du télescope de Galilée à l'accélérateur de particules (Simoulin, 2012), en passant par les détecteurs d'ondes gravitationnelles (Collins, 2004) ou la PCR (Rabinov, 1996). Loin de se limiter à des dispositifs dont la matérialité est évidente (Boaventura et Schultz, 2022), ces instruments incluent évidemment les dispositifs informatiques impliqués dans la production, la collecte, le stockage et la transformation des données (Bowker et Star, 1999). A partir du moment où la séparation entre *hardware* et *software* s'est généralisée, l'ordinateur est largement dépendant des logiciels[3], qui sont légions (Schindler et al., 2022). Aux différentes spécialités sont associés différents logiciels caractéristiques : Matlab pour les physiciens appliqués ; ImageJ pour le traitement d'images des biologistes, R pour les statisticiens ou encore Nvivo pour les sciences sociales qualitatives. Une enquête passée en 2020 en France rend ainsi visible la diversité de ces instruments numériques que les chercheurs peuvent mobiliser dans leur quotidien, allant des logiciels de tableur comme Excel à des programmes « maison » (Le Béchec et al., 2021). Ce paysage est devenu encore plus complexe avec l'avènement des plateformes qui embarquent les logiciels sous la forme de service accessible par internet (Plantin et al., 2018). Qu'ils se présentent sous la forme de morceaux de code informatique (scripts et programmes) ou de logiciels complet, ces entités représentent alors des instruments à part entière de l'activité scientifique, et pour beaucoup sont directement créés par et pour les chercheurs[4], à l'instar des autres instruments et prototypes. Quand ils font l'objet d'études, le focus porte surtout sur les situations de coordination, de mobilisation et d'organisation qu'ils permettent plutôt que sur leur histoire ou leur contexte de

2018) ; le logiciel de versionnement CVS provient d'une initiative académique (Alcaras, 2022) et de nombreux langages de programmation viennent à la base de laboratoires universitaires, par exemple Python (Meertens, 2022).

[3] « Research software plays such a critical role in day to day research that a comprehensive survey reports 90–95% of researchers in the US and the UK rely upon it and more than 60% were unable to continue working if such software stopped functioning » (Hetterick, 2014)

[4] En témoigne l'apparition progressive à partir de 2010 du statut de Research Software Engineer. Des collectifs dédiés sont créés dans des universités responsable de la création des logiciels de recherche : « *Fundamentally, RSEs build software to support scientific research. They generally don't have research questions of their own — they develop the computer tools to help other people to do cool things*.» (Woolston, 2022).

production[5] (Vertesi, 2019; Froger-Lefebvre et al., 2023 ; Shulz, 2021).

Bien entendu, les logiciels peuvent en effet paraître des « objet ennuyeux » (Méadel et Sire, 2017), uniquement digne d'intérêt sous leurs aspects spectaculaires – l'intelligence artificielle – ou intellectualisée – les algorithmes. Ainsi, l'arrivée fracassante des approches qualifiées d'« IA » favorise un regain d'attention sur les algorithmes qui équipent la recherche scientifique (Marcovich et Shinn, 2021; Jaton, 2023). Cette focalisation sur certains aspects du logiciel – sa conceptualisation sous forme d'algorithme ou certaines de ses facettes liés à l'apprentissage automatique - contribue en retour à invisibiliser tous les autres logiciels et programmes qui structurent l'activité de recherche au quotidien. Ceux-ci constituent pourtant les conditions de production des données, de leur analyse, et conditionnent par exemple la reproductibilité des résultats[6][7] (Vertesi, 2019). Dans une des rares études de STS s'attachant à saisir un logiciel scientifique comme objet, Matt Spencer distingue clairement le modèle conceptuel de simulation de son implémentation, et se concentre à documenter la construction d'un logiciel de simulation d'écoulement des fluides (Spencer, 2015). Il montre les enjeux qui se créent autour de la stabilisation d'un programme : l'intégration progressive de nouveaux contributeurs et utilisateurs ; l'évolution au gré des financements ; l'accumulation de vulnérabilités du code dues à une croissance trop désordonnée ; la réécriture de ce code (*refactorisation*) ; et progressivement d'une régulation collective du projet en charge de sa stabilité[8]. Cette étude de cas rend visible qu'à l'instar des autres outils scientifiques, des logiciels émergent dans des contextes situés et peuvent devenir des instruments plus génériques (Shinn, 2000).

Dans cet article, je propose de contribuer à une sociologie des instruments scientifiques à partir d'une étude de cas : les notebooks computationnels[9] du projet Jupyter. Pour les présenter en quelques mots (*cf. partie 1 pour une description plus détaillée*), les notebooks computationnels représentent un format de documents interactifs facilitant la programmation scientifique regroupant à la fois du code, du texte et des résultats comme des visualisations, introduisant de nouvelles pratiques rédactionnelles pour les chercheurs (Sauret, 2022). Ils représente une innovation issue des pratiques de recherche qui a progressivement acquis un statut de commun numérique d'instrument de traitement de données d'envergure mondiale, au-delà même des mondes scientifiques[10]. À ce titre, ils correspondent largement à un instrument générique de traitement des données (Shinn, 2000). Dans un article de Nature de 2018, une chercheuse interrogée pouvait ainsi répondre « *For data scientists, Jupyter has emerged as a de facto standard », says Lorena Barba, a mechanical and aeronautical engineer at George Washington University »* (Perkel, 2018). Dans leurs sillages, ils ont entraînés de nombreuses discussions sur les formats même de la communication scientifique.

---

5 La littérature existante traitant directement les logiciels scientifiques provient donc surtout de l'ingénierie logicielle autour des bonnes - et moins bonnes - pratiques. Ainsi, dans une des premières études devenue influente, les auteurs notent le fossé qui existe entre les pratiques logicielles des chercheurs et ceux du domaine du développement logiciel (Hannay et al., 2009 ; Kanewala et al., 2014 ; Arvanitou et al., 2021).

6 Comme le fait remarquer l'autrice d'une des rares études centrées sur un logiciel dans la recherche, « STS scholars have explored the politics and infrastructural assumptions of bespoke digital tools in the sciences, such as specialist databases and convergent cyberinfrastructures (Ribes and Bowker 2008; Lee et al. 2006). But STS studies of the off-the-shelf software suites are rare, despite the near ubiquity of smartphone smartphone apps, productivity suites, and social media platforms like Facebook or LinkedIn in the laboratories and communities we study (Vertesi 2014; Gillespie 2010) » (Vertesi, 2019)

7 Cependant, les travaux récents autour de la science ouverte conduisent à de nouveau s'intéresser aux logiciels qui deviennent une des briques indispensables de l'ouverture et de la reproductibilité des traitements. Par exemple, des enjeux se posent sur la détection des logiciels utilisés et leurs versions (Schindler et al., 2022).

8 « When [software] was a small/medium sized enterprise, there was a degree of tolerance for idiosyncratic working practices. What mattered was getting good simulations, and the quality of software was less of an issue. But as it grew, this tolerance dwindled » (Spencer, 2015).

9 Computationnel entendu au sens de réalisation de traitement et de calculs, incluant la manipulation de données.

10 La diffusion des innovations logicielles semble absente de la synthèse digitalSTS (Vertesi et Ribes, 2019).

Ainsi, la même année, un article titrait de manière provocatrice « *The scientific paper is obsolete* » (2018). Les notebooks se diffusent par ailleurs comme interfaces dans de nombreuses applications numériques, comme les grands calculateurs ou les entrepôts de données[11] (Thomas et Cholia, 2021).

Je vais m'intéresser à trois aspects principaux : (1) Comment émerge un nouveau programme situé dans une activité de recherche ? (2) Comment un tel programme devient progressivement un logiciel plus générique ? (3) Comment il vient à occuper une place de *standard* pour devenir une infrastructure, entendue comme « *a broad category referring to pervasive enabling resources in network form* » (Bowker et al., 2010) ? Répondre à ces questions nécessite à la fois de mieux caractériser les contours de ce qu'est un logiciel, détailler les enjeux de la rencontre entre open source et open science, et prendre au sérieux la dimension processuelle de la stabilisation des dispositifs en dépliant l'histoire longue. L'intérêt de se concentrer sur les notebooks Jupyter est alors double, au-delà de l'importance qu'ils prennent dans le domaine du traitement des données : d'une part, ils permettent d'interroger le tournant « données » de la recherche scientifique actuelle, notamment autour de la recomposition des publics impliqués et des nouveaux enjeux de reproductibilité ; l'autre est que leur trajectoire permet de réunir un ensemble d'interrogation, liant les aspects concernant la programmation au logiciel, le logiciel aux infrastructures, et les infrastructures aux plateformes.

La première partie clarifie les propriétés des notebooks computationnels et leur rôle dans l'implémentation de la programmation lettrée pour le traitement de données au cœur des enjeux actuels de la science ouverte. La deuxième partie revient sur l'origine académique des premiers notebooks ancrés dans la dynamique open source des prototypes d'instruments logiciels portée par la communauté de la programmation scientifique en langage Python. La troisième partie s'intéresse au désencastrement progressif du projet Jupyter non seulement de sa communauté d'origine mais du secteur même de la recherche et sa transformation en infrastructure générique du traitement des données, notamment via sa plateformisation sous différents services (Plantin et al., 2018).

> Cet article repose sur une perspective phénoménologique du logiciel[12] triangulant différentes sources. Je vais tout d'abord recourir au corpus constitué par l'ensemble des déclarations des différents membres du projet et la littérature qui se développe notamment dans les computer science autour de l'usage des notebooks. Je m'appuie aussi sur une ethnographie participative engagée à partir de 2015 par un séjour dans l'équipe du projet Jupyter, et une utilisation régulière de ces outils dans le cadre de mon activité de recherche en sciences sociales. J'ai ainsi participé à la grande conférence Jupytercon2023 organisée à Paris, et j'ai construit en collaboration avec HumaNum et Datactivist des démonstrateurs pour les SHS. Dans le cadre de cette ethnographie participative, j'ai eu l'occasion de conduire une enquête par questionnaire auprès des communautés d'Humanités Numériques. Enfin, des données extraites des dépôts publics du projet permettent de suivre l'évolution du code au fil des années. Une remarque méthodologique doit être faite : cet article entend contribuer à la sociologie, et à ce titre différentes dimensions techniques sont volontairement invisibilisées. Néanmoins, pour comprendre les enjeux autour des notebooks, il est nécessaire de présenter certaines dimensions ayant trait à la programmation.

1. De la programmation scientifique aux notebooks computationnels

1.1 Qu'est-ce qu'un notebook ?

[11] « Jupyter is proving to have a transformative impact on modern computational science by enabling a new integrative mode of interactive supercomputing, where code, analysis, and data all come together under a single visual interface that can seamlessly access powerful hardware resources ». Par exemple le Health Data Hub français https://www.health-data-hub.fr/offre-technologique

[12] Dans le sens où je ne cherche pas à descendre au niveau du code mais considère le logiciel caractérisé par les fonctions qu'il permet de réaliser. La perspective temporelle longue envisagée ici ne permet pas de rentrer dans une analyse détaillée du code.

Le projet Jupyter est devenu en l'espace d'une dizaine d'année non seulement la référence des formats de *notebooks computationnels*, passant des usages de la programmation Python à une infrastructure du traitement de données mondiale[13]. La diffusion progressive de cette innovation numérique s'appuie sur un écosystème composé de différents logiciels interdépendants, de formats, et de services qui se sont progressivement mis en place. Avec des millions d'utilisateurs à travers le monde[14], notamment du domaine de la *data science*, cette réussite exceptionnelle a transformé les pratiques à la fois individuelles et collectives (Quaranta et al., 2022). Ils ont introduit de nouveaux styles de programmation (Grotov et al., 2022).

Pour comprendre ce qu'est un notebook computationnel actuellement, il faut déplier plusieurs couches. Ce terme désigne un format permettant l'exécution de langage de programmation (que ce soit le langage Python, R ou d'autres) dans une interface qui contient à la fois le code, les résultats, et des éléments de narration (du texte, des images, mais aussi des éléments interactifs)[15]. Le projet Jupyter les présente comme « *a shareable document that combines computer code, plain language descriptions, data, rich visualizations like 3D models, charts, graphs and figures, and interactive controls. A notebook, along with an editor (like JupyterLab), provides a fast interactive environment for prototyping and explaining code, exploring and visualizing data, and sharing ideas with others*. ». Il ne s'agit ni à proprement parler d'un logiciel unique, ni d'un programme, mais d'un ensemble interconnecté d'outils et de formats permettant de programmer de manière interactive. D'une certaine manière, il s'agit de l'extension de la programmation – une série d'instruction sous forme textuelle à un ordinateur – vers une interface logicielle favorisant l'interactivité[16]. Ils ont actuellement un statut intermédiaire entre *logiciel*, *standard*, *infrastructure* et *plateforme*.

Un exemple mis en avant par le projet Jupyter lui-même est le notebook produit par les équipes du grand détecteur d'onde gravitationnelles LIGO et VIRGO, récipiendaires du prix Nobel de physique en 2017, pour leur observations de la collision de trous noirs. Ils ont présenté leur démarche en ligne dans le cadre d'un notebook qui retrace les étapes de l'analyse (cf. figure 1)[17]. Le dépôt public Github – plateforme collaborative lié à la programmation open source - du projet contient un ensemble de données et un document notebook qui récapitule les étapes permettant de charger, analyser et visualiser les données des télescopes. Ce document, qui dans les fait est un fichier au format *.ipynb,* comprend des cellules de textes, comme le titre, ou le texte de début : « *Welcome! This IPython notebook (or associated python script LOSC_Event_tutorial.py) will go through some typical signal processing tasks on strain time-series data associated with the LIGO Event data*

[13] Il faut noter qu'outre le projet Jupyter, de nombreuses solutions de notebooks computationnels existent : Noteable, Google Colab, Kaggle, Microsoft Azur, etc. Plus encore, le format notebook regroupe en puissance d'autres types de supports similaires, par exemple des formats plus statiques comme Quarto.

[14] Pour avoir un ordre de grandeur, les statistiques de téléchargement sur le mois d'août 2023 sur pypi.org se comptent en millions. Pour IPython : 26,795,744 ; Jupyter notebook : 16,962,036 ; Jupyterlab : 10,496,539.

[15] Les commentateurs tâtonnent encore pour définir clairement ces entités hybrides, qui dans les faits peuvent recouvrir des dispositifs assez différents, certains uniquement exécutable dans un environnement sur l'ordinateur, d'autres en ligne. De nombreuses appellations ont été proposées pour désigner des solutions différentes « article exécutables », « computational notebooks », etc. Une proposition a été faite d'un « genre exécutable » (Zurbach, 2022). Si le point de départ et la filiation commune revendiquée est claire – la programmation lettrée (ou literate programming), ce qu'elle devient dans chacun des outils est moins évident. Pour saisir les enjeux de ces technologies spécifiques, il est nécessaire d'établir un certain nombre de distinctions même si elles deviennent vite inopérantes tant les outils évoluent en permanence.

[16] « If you think about it, these are the kinds of workflows that Jupyter has always been about, but Jupyter has evolved in the opposite direction. Instead of adding scripting to GUIs, Jupyter added GUIs to scripting. » https://blog.jupyter.org/plug-your-application-into-the-jupyter-world-805e48918801

[17] https://github.com/losc-tutorial/LOSC_Event_tutorial Il faut noter que ces notebooks sont pédagogiques, et à ce titre illustrent une autre propriété de ces supports qui est de permettre d'ouvrir une démarche de traitement de données à un public plus large.

releases from the LIGO Open Science Center (LOSC) ». Il contient ensuite des cellules de code, dans ce cas écrit en langage Python, qui permettent de faire les différentes tâches (charger les données, les transformer, créer un graphique). Sous les cellules de code, les résultats du traitement sont affichés après exécution. Consulté en ligne, ce document se présente comme une page statique contenant l'ensemble des éléments, code, texte et résultats. Cependant, une fois ouvert dans un environnement adéquat, chaque cellule peut être modifiée et exécutée, rejouant le traitement, et donnant ainsi la possibilité de refaire le traitement avec d'éventuelles modulations[18]. Cette ouverture peut se faire sur un ordinateur avec un logiciel du projet Jupyter (Jupyter notebook ou Jupyter Lab), ou à travers un service en ligne comme Jupyter Hub, Binder ou Google Colab. Une fois le document ouvert dans un tel environnement, l'utilisateur peut exécuter le contenu en faisant le lien entre le document et l'interpréteur du langage Python, et donc retraité les données ou apporter des modifications au document. Les notebooks computationnels correspondent donc à la fois à un format d'écriture d'opérations de programmation, à un ensemble de logiciels permettant de le rendre interactif, eux-mêmes pouvant être mis à disposition comme un service sur internet ou installé sur l'ordinateur de l'utilisateur, et plus généralement à une normes de programmation consistant à rendre exécutable et interactif un document contenant de la programmation. Ils relient des pratiques de programmation (le code de traitement des données) et la restitution de la démarche intellectuelle (la narration avec les résultats).

Data Gaps

NOTE that in general, LIGO strain time series data has gaps (filled with NaNs) when the detectors are not taking valid ("science quality") data. Analyzing these data requires the user to loop over "segments" of valid data stretches.

In this tutorial, for simplicity, we assume there are no data gaps - this will not work for all times! See the notes on segments for details.

First look at the data from H1 and L1

```
# both H1 and L1 will have the same time vector, so:
time = time_H1
# the time sample interval (uniformly sampled!)
dt = time[1] - time[0]

# Let's look at the data and print out some stuff:

print('time_H1: len, min, mean, max = ', \
    len(time_H1), time_H1.min(), time_H1.mean(), time_H1.max() )
print('strain_H1: len, min, mean, max = ', \
    len(strain_H1), strain_H1.min(),strain_H1.mean(),strain_H1.max())
print( 'strain_L1: len, min, mean, max = ', \
    len(strain_L1), strain_L1.min(),strain_L1.mean(),strain_L1.max())
```

Au-delà de leur propriétés, il est pertinent d'envisager le notebook computationnel comme un support intermédiaire : à la fois entre plusieurs usages, entre plusieurs types de support, et entre plusieurs groupes sociaux[19]. En termes de contenu, ces notebooks sont un format intermédiaire entre un script de programmation, les résultats produits et un document explicatif de la démarche. Dans l'exemple présenté, le notebook permet de retracer les traitements tout en les resituant dans la démarche intellectuelle. En termes d'usagers, le projet Jupyter est à l'intersection entre les mondes.

[18] https://blog.jupyter.org/congratulations-to-the-ligo-and-virgo-collaborations-from-project-jupyter-5923247be019

[19] En ce qui concerne les groupes sociaux, cela peut se traduire par un véritable rôle d'objet intermédiaire entre des communautés de pratiques (Vinck et Trompette, 2009). Ce n'est pas l'objet de cet article de revenir en détail sur les échanges entre les communautés autour des notebooks, des réflexions à ce sujet ont été initiées dans le cas du projet NOOS, notamment lors d'une présentation à Jupytercon 2023 (Schultz et al., 2023).

Les utilisateurs mais aussi les concepteurs des notebooks se retrouvent à la fois dans le monde académique, le monde du logiciel libre, la société civile et le monde économique. Ils sont ainsi de plus en plus utilisés dans les activités de data science qui se multiplient dans les entreprises. Dans le cas de l'exemple, ce support sert même d'intermédiaire entre des chercheurs et le grand public qui voudrait avoir une idée des traitements réalisés. Enfin, ils couvrent une diversité d'usages du traitement des données : ils peuvent à la fois servir de « bac à sable » pour explorer différentes approches d'analyse des données avec des lignes de programmation, de support pour des résultats intermédiaires, mais aussi être stabilisés comme des documents finalisés et publiables permettant la reproductibilité d'analyse avec l'objectif d'être de véritables « articles reproductibles ». Dans l'exemple, la vocation d'une reproductibilité pédagogique est mise en avant. Dans d'autres cas, les notebooks serviront de support à l'exploration des données.

La littérature qui se développe, notamment en science informatique[20], traduit ce caractère intermédiaire en termes d'une tension consubstantielle au format entre exploration et explication : faciliter d'une part le bricolage et l'expérimentation d'opérations désordonnées sans grand respect pour une démarche rigoureuse ; mais aussi pouvoir produire un rendu final qui raconte une histoire logique et permet sa reproduction étape par étape (Rule et al., 2018). Ils ne sauraient ainsi être capturés ni comme un simple « script » informatique, ni uniquement des « articles exécutables »[21] La galaxie d'outils progressivement stabilisée autour des notebooks computationnels Jupyter est l'extension de cette situation d'intermédiarité, allant de la construction d'applications autonomes à part entière, par exemple permettant d'interagir avec des données, à la production de documents finalisés comme des livres.

1.2 Resituer la préhistoire des notebooks dans la programmation scientifique

Pour saisir comment ces nouveaux objets intermédiaires sont apparus, il est nécessaire de revenir sur une histoire plus longue à l'interface entre le monde académique et le monde de la programmation.

Loin d'être une génération spontanée, la préhistoire des notebooks du projet Jupyter relie de fait de nombreux logiciels différents, initialement liés au domaine de la recherche en mathématique, mais aussi la formalisation progressive d'un besoin qui est à relier à un « style de pensée » proche de la recherche scientifique, qui est celui d'avoir à disposition une interface interactive pour explorer. Comme le remarque Gabriel Alcaras (2022), les mondes de l'open source sont attentifs à leur propre histoire, et les grandes étapes qui ont conduit à la possibilité des notebooks Jupyter ont déjà été racontées, souvent même par leurs fondateurs qui en ont fait une série d'étapes cruciales[22]. Dans une courte histoire des notebooks, William Horton (2020) se concentre sur trois principales dimensions à prendre en compte pour penser leur genèse[23] : l'expansion de la *programmation scientifique* ; la conceptualisation de la *programmation lettrée* et l'expansion des *logiciels open source.* Ce faisant, cela rend visible la centralité des valeurs de la recherche scientifique dans le développement logiciel et l'extension permise par les dynamiques des communautés open source[24].

---

[20] Celle-ci provient surtout d'une réflexion des sciences informatiques pour caractériser ces pratiques et identifier les bons usages, avec seulement une faible proportion des articles spécifiques au monde académique que ce soit pour la recherche ou l'enseignement (Beg et al., 2021 ; Wageman et al., 2022 ; Samuel et al., 2022).

[21] De fait, trois quart des notebooks hébergés sur Github échouent à l'être (Pimentel et al , 2021).

[22] https://www.oreilly.com/radar/the-state-of-jupyter/

[23] https://www.youtube.com/watch?v=kFhhCOeYcGw&ab_channel=EuroPythonConference

[24] « Major research collaborations and communities across physics, chemistry, biology, economics, Earth science, etc. leverage Jupyter as a foundational tool for their computational work, collaboration, education, and knowledge dissemination ». (Granger et Perez, 2021)

La *programmation scientifique* correspond à l'usage par les chercheurs de la programmation qui se distingue de l'ingénierie logicielle. Si l'ingénierie logicielle, entendue comme l'ensemble des savoirs et des pratiques pour concevoir des logiciels, occupe une place centrale, notamment en raison de sa capacité commerciale et des oppositions qui se sont constituées autour du logiciel libre, elle n'est qu'une partie des usages de la programmation. En effet, de nombreux chercheurs sont amenés à utiliser des langages de programmation pour automatiser des tâches, réaliser des calculs numériques, explorer des données ou encore développer des outils spécifiques pour leur activité. Ces opérations se caractérisent par le fait d'être dirigées vers la résolution de problèmes spécifiques, une portée restreinte, un usage réservé à quelques spécialistes, et un environnement fortement incertain. Elles nécessitent cependant une adaptation aux ordinateurs et à leur logique, donc de développer une forme de « *computationnal thinking* »[25].

La programmation scientifique se distingue toutefois par son caractère situé : elle cherche à répondre à des questions spécifiques, exploratoires et itérative, avec beaucoup de code *ad hoc* écrit. Elle est largement idiosyncrasique. Cependant, les chercheurs doivent quand même apprendre à abstraire et étendre les solutions à d'autres cas, pour ne pas copier encore et encore le même code. Cela peut donc amener progressivement au développement de prototypes de logiciels qui structurent l'organisation d'équipes ou de collectifs (Spencer, 2015). Cette question du logiciel en recherche fait d'ailleurs l'objet de davantage d'attention notamment avec des financements dédiés (Heroux et al., 2021). A ce titre, la programmation faite par les chercheurs se distingue largement de l'ingénierie informatique engagée dans la création et le maintient des infrastructures numériques à destination d'un public large d'utilisateurs (Hannay et al., 2009), donc contraint de respecter des règles de qualités de plus en plus contraignantes (Granger et Perez, 2021). Pour autant, il existe en puissance une relation entre le script et le logiciel car dans certains cas les usages s'étendent à un collectif plus large, par exemple la spécialité. Le développement des outils de programmation scientifique en Python mais aussi en R s'inscrit dans cette logique d'équipement de la recherche par et pour les chercheurs[26].

L'activité de la programmation scientifique étant avant tout tournée vers la production de résultats pour la publication, les traitements comptent davantage que la qualité du code en lui-même : le produit n'est généralement pas le logiciel, et la programmation est généralement au service de la réflexion et des résultats[27]. La *programmation lettrée* s'inscrit dans la continuité de la programmation scientifique comme relevant d'une formalisation de la spécificité de cette démarche. Dans l'article fondateur de 1984, Donald Knuth, chercheur en informatique et mathématique, père aussi du langage de mise en forme de document TeX, propose de définir un nouveau mot d'ordre pour améliorer les pratiques de programmation : « *considering programs to be works of literature »*, et donc de changer d'attitude *« instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do* » (Knuth, 1984). Il implémente cette philosophie dans un premier langage dual (reposant sur TeX et Pascal). Si cette approche ne trouve pas de succès auprès des professionnels de

---

25 « In coding, we have to clarify to the point that a dumb machine can understand our thought process and what is it we are going to do. It means we have to organize the process in a very precise and unambiguous way [...] That process of logical systematic organized abstract computational thought, that I think is a useful valuable skill in many contexts in life. Being able to think through a problem in a systematic organized logical way, understand what parts of the problem are abstractable and generalizable, understand exactly how to accomplish a that will complete is a useful skill » https://www.youtube.com/watch?v=g8xQRI3E8r8&ab_channel=O%27Reilly

26 « in the Python world, scientists have collaboratively developed an open ecosystem of foundational tools that provide the key functionality needed for everyday computing work. » (Pérez et al., 2011).

27 Bien entendu, la limite est la spécialisation d'acteurs dans la production de logiciels scientifiques, qui conduit par exemple à la fondation de revues permettant de valoriser sous la forme de publication leurs logiciels (JOSS ou JOSR).

la programmation, elle va avoir des débouchés dans le domaine du calcul numérique[28]. Cependant la philosophie générale de la programmation lettrée va avoir une existence propre, et va trouver progressivement sa place dans de nombreux logiciels. Ainsi, les inspirations historiques citées sont le logiciel Mathematica, 1988, créé par Stephen Wolfram, avec une interface de type notebook créée par Theodore Gray, mais aussi le logiciel Maple créé en 1992 qui avait une « worksheet interface », ou encore le logiciel Sagemath, créé en 2005 par William Stein[29] : « *all such systems provide interactive exploratory environments that make language features and libraries immediately available to scientists who can use them to explore a problem domain […] In practice, however, scientists use rich environments with lots of functionality, and integrating multiple tools is an attractive aspect of commercial technical computing systems.* » (Pérez et al., 2011). La programmation lettrée sert de marqueur à une « autre » manière d'utiliser la programmation, entrelacée avec la réflexion et le problème à résoudre.

Il existe alors une continuité d'usages de la programmation dans les communautés scientifiques, certains qui aboutissent à la stabilisation de logiciels diffusant au-delà des premiers concernés. Cette ouverture au-delà des contextes locaux de pratiques est largement favorisée par les dynamiques du *logiciel libre* puis du modèle économique de l'*open source*. Comme le souligne les auteurs de la synthèse sur les information infrastructure studies, « *We cannot do the history of software without doing the history of their surrounding organizations* » (Bowker et al., 2010), et dans ce cas les logiques de l'open source sont centrales pour comprendre le développement des logiciels scientifiques en général, et des notebooks Jupyter en particulier. Le développement du logiciel libre initié largement dans les années 1980 avec d'abord le projet GNU de Richard Stallman continué dans la Free Software Foundation puis amplifié dans les années 1990 avec le développement de Linux, a surtout été raconté dans la perspective des mobilisations autour de valeurs (Kelty, 2008), et plus récemment sous celle du professionnel du code (Alcaras, 2022). Si l'interdépendance avec le milieu académique est forte, la sociologie des sciences ne s'y est pas beaucoup intéressée. Or, le monde du logiciel libre et celui de la science partagent des valeurs, qu'il s'agisse du communalisme, ou de formes de régulation comme la reconnaissance centrée sur la contribution des individus. Une comparaison organisationnelle fait apparaître de nombreux points communs sur les pratiques de production, comme « communautés productives » (Gläser et al., 2007).

L'évolution du logiciel libre vers une conception plus pragmatique du code ouvert au tournant des années 2000 (Kelty, 2008) rencontre les pratiques des chercheurs qui ont besoin de bricoler des outils pour leur problème puis des les partager. Comme l'écrit sur son blog Fernando Perez, il est facile de faire un lien entre science et logiciel libre : « *I realized that in many ways, the open source community practiced many of the ideals of science better than academia* » (blog de F. Perez, 11/12/2013). Ce faisant, il se crée une relation affinitaire entre science ouverte et logiciel libre, dans la mesure où ils participent à la reproductibilité des résultats en garantissant l'accès au code. L'évolution du format notebook montre l'importance des mécanismes de collaboration communautaires spécifiques à l'open source qui ont permis d'appuyer progressivement le logiciel sur une diversité d'usages, de praticiens venant de spécialités différentes, et favorisant aussi une diversité de formes de réappropriations permises par l'ouverture du code et des outils. Cela permet notamment de rendre compte de pourquoi les logiciels similaires plus spécifiques et propriétaires implémentant la logique des notebooks comme Mathematica n'ont pas connu le même destin que le

[28] « Heavy annotations also conflict with the idea in software engineering that well-structured code should "speak for itself" and be understandable with minimal documentation » (Kelly et al., 2018)

[29] « This definition of interactive computing is rooted in the modern scientific computing community. Tools such as IDL (1977), Maple (1982), Matlab (1984), and Mathematica (1988) offer this mode of interaction. It shouldn't be surprising that the two of us grew up as physicists using these interactive computing tools as a foundational part of our computational workflows. Indeed, we created IPython and Jupyter initially because we wanted the same type of interactive computing experience in the Python programming language. » (Granger et Perez, 2021)

projet Jupyter[30]. L'économiste Paul Romer, récipidendaire du prix d'économie de la banque de Suède en mémoire d'Alfred Nobel, et converti enthousiaste aux notebooks Jupyter[31], fait cette analyse dans un billet de blog en pointant le caractère central des dynamiques de l'open source, et de la proximité avec les valeurs de la science pour la création de la confiance sociale : « *Membership in an open source community is like membership in the community of science […] Because the communities of science and open source accept facts as the ultimate source of truth and use the same public system for resolving disagreements about the facts, they foster the same norms of trust grounded in individual integrity »*[32]. Il propose une explication générale, bien que simplificatrice, du succès de Jupyter : « *Jupyter is succeeding because the norms of the community that is developing it are aligned with the norms of its users*. »[33].

Les notebooks Jupyter sont donc la continuité d'équipement d'une approche lettrée de la programmation scientifique en contexte d'open source. Je propose de revenir sur les étapes de cette innovation pour insister sur le caractère processuel de la stabilisation et de la diffusion d'un logiciel initialement ancré dans les pratiques de recherche.

2. Le serpent et la science : l'origine académique des notebooks Python

Le projet Jupyter représente le désencastrement progressif d'un « *one use case* » initié par un chercheur en physique vers de nouveaux usages de plus en plus génériques. Cette évolution s'étend sur une vingtaine d'années entre 2000 et 2020, qui a vu arriver de nombreux bouleversement dans le paysage du logiciel et de l'open source[34]. Cela a été accompagné par une montée en abstraction, associant une conceptualisation des usages avec des outils permettant de couvrir un nombre croissant de situations. Trois principaux moments sont identifiables : la naissance d'un environnement interactif pour le python scientifique, IPython ; l'avènement du notebook comme nouveau format interactif pour programmer en Python ; le moment d'abstraction du projet Jupyter qui s'éloigne de ses origines scientifiques et pythonnesques.

2.1 La structuration d'une communauté de Python pour la recherche

L'histoire de IPython a été racontée à différents moments par ses créateurs, que ce soit lors de conférences ou de podcasts[35]. A l'origine se trouve la volonté de Fernando Perez, alors doctorant en physique des particules à l'université du Colorado, de développer un environnement interactif pour

---

30 « So here is my conjecture about the question the article poses. Mathematica failed, despite technical accomplishments, because the norms of its developers clashed so obviously with the norms of its intended users. Jupyter is succeeding because the norms of the community that is developing it are aligned with the norms of its users. » https://paulromer.net/jupyter-mathematica-and-the-future-of-the-research-paper/

31 « I stopped using Mathematica and gave up on notebooks, so it was only recently that I discovered how easy it is to use the Jupyter notebook to as a front end for Python libraries. It offers the best REPL I've ever used. It does a better job of delivering what Theodore Gray had in mind when he designed the Mathematica notebook. It lets me get quick feedback, via text or graphics, about what happens when I select a line of code and run it. »

32 https://paulromer.net/jupyter-mathematica-and-the-future-of-the-research-paper/

33 Il serait possible même d'aller un peu plus loin, et de voir suivant l'idée de Christopher Kelty pour le logiciel libre que le projet Jupyter est devenu un « public récursif » - « A recursive public is a public that is constituted by a shared concern for maintaining the means of association through which they come together as a public » (Kelty, 2008, p. 28) – constituant sa communauté autour de ses outils.

34 Cette histoire s'est déroulée en même temps qu'une transformation progressive du paysage du logiciel entre le début des années 2000 et le début des années 2020, qui a par exemple vu la création du logiciel de versionnement Git en 2005 au sein de la communauté Linux (Alcaras, 2022) permettant la création de Github comme plateforme de l'open source en 2008, et son rachat par Microsoft en 2018. Son développement arrive après les batailles de valeurs entre logiciel libre et code ouvert qui a laissé la place à un monde du développement logiciel où contributeurs bénévoles et industries de l'informatique se côtoient (Kelty, 2008).

35 https://www.pythonpodcast.com/episode-10-brian-granger-and-fernando-perez-of-the-ipython-project

utiliser Python dans le cadre de sa thèse, « a *simple personal fix for a problem in my own workflow »*[36]. Cette philosophie de partir des usages et de constituer progressivement des outils est omniprésente dans la logique de programmation scientifique[37]. Après avoir utilisé de nombreux logiciels dans sa formation de physique, notamment Maple et Mathematica, qui l'inspireront par la suite, il a fait progressivement le choix de se tourner vers la programmation scientifique en langage Python en plein développement[38]. Pourquoi Python[39] ? Pour lui, il y avait une volonté de simplifier son activité dispersée sur un ensemble d'outils et logiciels avec une solution unique – peut-être une conséquence du style de pensée unificatrice propre à la physique[40]. Dans l'état des outils existants, il manquait « *a good interactive computing environment »* qui permettait de retrouver le style d'interaction dont il pouvait avoir l'habitude[41]. Cette notion d'environnement interactif de programmation est à la fois un héritage du parcours de physicien et un élément central pour comprendre les choix fait ensuite autour des notebooks. Il soutient sa thèse en 2002, et continue sur un postdoctorat au sein de l'université :*« The reason why I wanted to do it in Python is because I was using a proprietary tool for my work, but I wanted to do my scientific work with open tools. I think of science's mission as opening the black box of nature. It's a bit nonsensical to do that with tools that we are not allowed to open and understand, including proprietary software »*[42]

Inspiré de deux autres consoles interactives en Python déjà disponibles[43], rappelant l'activité permanente de circulation des idées et des codes dans le monde de l'open source, il développe en 2001 la console IPython qui vise à répondre à un objectif précis : faciliter l'interaction avec l'ordinateur du chercheur en physique pour tester et explorer des traitements. La première version, qui n'a pas été rendue publique, est codée en octobre 2001. Il met ce petit logiciel d'abord sur la page de l'université, continue de le développer et de le présenter à différentes conférences notamment celle de la NASA sur la programmation scientifique. Ce faisant, il initie une boucle de rétroaction positive, courante dans les expériences des communautés open source où la contribution initiale est valorisée par d'autres utilisateurs qui font des retours voire des contributions, et incitent en réaction à un plus grand engagement[44]. Le développement de IPython, mais plus généralement

---

36 Son parcours est un peu chaotique, se retrouvant renvoyé de sa thèse en cinquième année avant de la finir sous une autre supervision, il "productively procrastinate" sur le développement de Ipython, il soulignera plus tard que "open source saved me".

37 Il faut noter l'importance du créateur d'un projet informatique, surtout dans le domaine de l'open source. Notamment, le terme BDFL « Benevolent Dictator for Life » souvent utilisé pour désigner le pouvoir symbolique et réel du fondateur d'un projet sur son devenir souligne l'importance que joue la trajectoire et l'identité de la personne. Bien entendu, de nombreuses autres personnes contribuent au développement des projets.

38 L'entreprise Enthough qui développe la bibliothèque SciPy est créée en 2001 et la conférence SciPy est ainsi lancé sous la forme d'un workshop en 2002 https://conference.scipy.org/past.html,

39 Ce n'est pas le lieu de faire ici l'histoire du langage Python, on renverra à l'article du fondateur d'un langage antérieur, le langage ABC (Meertens, 2021). Citons cependant le créateur du langage Python qui porte en germe un peu de cette philosophie « Python is now also the language of amateurs, and I mean that in the best possible way » (von Rossum, 2021).

40 "I was using for the analysis of my data, the simulations themselves were run largely through computers and these were high-performance C codes, but I was using for the analysis of a lot of the data and the modeling of that data, I was using a, a vast collection of tools. I had PEARL scripts. I had bash and awk and sed scripts. I had C codes. I had codes in IDL. I had codes in Mathematica. I had codes in gnuplot, and I was finding myself, um, switching between many, many languages all the time." (blog de Fernando Perez, 2012)

41 "If you typed Python in the command line, you got a, an interactive shell, it was a very, very primitive and it didn't allow me to do the kinds of things that were very natural in interactive scientific workflows with tools like IDL or Mathematica that I used heavily or Matlab or Maple that other used which was simply to type a bit of code, see the results right there, open a plot, look at the files on, on the file system, et cetera." [Blog de Fernando Perez, 2012]

42 https://data.berkeley.edu/news/project-jupyter-celebrates-20-years-fernando-perez-reflects-how-it-started-open-sciences

43 LazyPython, développé par Nathan Gray à Caltech, et Interactive Python Prompt (IPP) de Janko Hauser à Kiel's Institute of Marine Research

44 «When I first posted the first release of IPython in scientific Python lists, others immediately jumped on it. Other

des outils de la programmation scientifique en Python, repose en effet sur l'existence d'une vaste communauté d'utilisateurs qui s'entraide en permettant l'amélioration collective des outils mais aussi nourrit une sociabilité partagée[45]. Celle-ci loin d'être purement imaginaire est incarnée lors d'événements comme les conférences SciPy (Scientific Python). Elle est centrale pour expliquer l'engagement et le temps passé, mais aussi ensuite la volonté de contribuer aux outils.

> *"So I think what I found was not only a technical challenge and a fun project to procrastinate a little bit on my physics research, for a while on, but also a community of people with the same set of ideals, and I think it's important to kind of see how these 2 play with each other because it wasn't only the computational questions and the software aspects and the usefulness of the tools for my physics research. It was also the fact that there was a community of like-minded people who were willing to collaborate and work together, and it was kind of in contrast to the ethos of, of competition that is so pervasive in science where people, uh, are often unwilling to even tell you what they're working on because they're afraid you're going to scoop them or who is going to get the paper first or who is going to be the lead author on the paper. Instead, these were people who basically would stay up late at night helping me with questions on a mailing list or on a chat room, uh, in order to help me install these libraries for no other reason than just helping me out."*

Trouvant progressivement sa place dans le paysage de la programmation scientifique, notamment avec une présentation lors de la deuxième conférence SciPy, IPython gagne en utilisateurs. Deux facteurs sont importants pour le devenir du projet : la diffusion dans les usages, avec les retours des utilisateurs, et la possibilité de continuer à développer et adapter le logiciel pour le faire gagner en pertinence par rapport à sa communauté d'adoption. Comme le remarque Kelty, « *modifiability is an imperative for building infrastructures that can last longer* » (Kelty, 2008, p. 12). Le logiciel reçoit un accueil enthousiaste par la communauté du python scientifique. Eric Jones, fondateur d'Enthought, entreprise soutenant largement le développement open source autour de Python, propose de l'héberger sur leur page[46]. Cette adoption attire des contributeurs, modifiant à la fois l'ampleur du projet et sa capacité d'intégrer les suggestions. En 2004, son ami de la période de thèse Brian Granger le rejoint dans le développement du logiciel, ainsi que Benjamin Ragan-Kelly, étudiant de Brian.

Pour accompagner cette diffusion qui en retour demande de faire évoluer le logiciel, un enjeu central est de pouvoir stabiliser une activité professionnelle. A ce titre, il faut noter la position de Fernando Perez dans un régime « technico-instrumental » comme définit par Terry Shinn (Shinn, 2000), entre la production de résultats de recherche et le développement d'instrument. Fernando Perez continue après sa thèse à l'Université du Colorado en tant que chercheur postdoctorant au département de mathématiques appliquées. Le support institutionnel de Mark D'Esposito, chef d'équipe du *Berkeley Brain Imaging Center* joue un rôle important au tournant entre la fin de sa thèse et la reconnaissance de sa contribution à l'instrumentation logicielle de la recherche, accompagnant le dépôt de demandes de financement au NIH nécessaire pour stabiliser une position, et amenant un virage vers les neurosciences. « *Very importantly, though, there were people at UC Berkeley early on who supported me when I was still a postdoctoral scholar in Colorado doing*

---

scientists, who came from other fields, said, "This is valuable, and we're interested in what you're doing." That feedback loop was critical to putting me back on track, along with the support of other mentors who helped me out. »

45 "I don't know about the rest of you... I came for the language, but I stayed for the community." Brett Cannon, Python Core Dev, Pycon US 2014 Opening remarks

46 « So here we have a situation where a student had on the other end of the line the CEO of a company which, granted, at the time was a very small company, but still helping me for hours on a chat room just to fix some bizarre compilation problems and, and that community effect was a big part of it. »

more traditional applied mathematics research with Python tools. I wasn't invested enough in the purely applied mathematics community to make a career just out of that. People at UC Berkeley that I connected to because of the Python community offered me a team and, eventually, a job. ». Le financement de projets déposés au NIH permet le financement d'une position académique en 2007.

Dans le développement de logiciels de recherche, et plus généralement dans l'open source, le code est très lié avec son développeur et circule avec lui. Fernando Perez rejoint l'Université de Berkeley en 2008 sur une position de chercheur et amène avec lui le projet IPython[47]. Cette évolution professionnelle le rapproche du logiciel scientifique à proprement parler, alors qu'initialement il était dans la continuité d'un usage de la programmation scientifique dans et pour la recherche :« *While last year we spent some effort introducing the language and to a certain extent justyfing its use in real-world scientific work, we felt that this time, the growth of the many python projects out there speaks for itself and that we should instead turn our attention to actual tools and projects useful for specific work. [...] for a number of years, many of us have been developing tools and justyfing Python as a viable alternative to tools such as Matlab or IDL, but we need to start moving away from that mode*. » (Perez, blog 3/09/2009). Tout en continuant à mener une recherche orientée sur des thématiques spécifiques, il oriente progressivement son activité sur le développement d'une instrumentation logicielle pour la recherche. Si la genèse d'un instrument logiciel peut se faire de manière circonstancielle, sa pérennité nécessite la création de carrières dédiées avec la transition progressive de chercheurs vers le développement logiciel, car comme le rappelle Matt Spencer : « *It is not just the money; careers are being invested.* » (Spencer, 2015), et la mise en place progressive d'une administration dédiée sur ce qui était au départ juste quelques lignes de code.

Au-delà de la situation individuelle permettant aux coordinateurs d'un projet de le maintenir, développer et étendre un logiciel nécessite de s'inscrire dans le temps long et de stabiliser des ressources. Comme le remarque Matt Spencer, les projets scientifiques avancent par saut en fonction des financements disponibles : « *The piecemeal growth that characterizes this software is a consequence of the manner in which science is funded.* » (Spencer, 2015). Des moments comme la participation en 2005 au Google Summer of Code permet de faire avancer certaines fonctions du logiciel. Ce sont surtout les acteurs privés de l'écosystème du Python scientifique qui seront les premiers financements. La stabilisation de la situation professionnelle des développeurs de IPython permet à la fois la lisibilité du projet mais aussi une recherche plus systématique de soutien. En 2010, Enthought finance le développement de différentes propriétés pour IPython, comme la console Qt ou le protocole réseau, qui ouvre la voie aux caractéristiques qui serviront de fondation au projet Jupyter. Cette période voit une réécriture (*refactorisation*) du code vers une architecture davantage pensée en réseau. Elle voit aussi l'arrivée de nouveaux contributeurs comme Thomas Kluyver qui assure la transition de Python 2 vers Python 3[48], soulignant l'importance de l'arrivée de contributeurs se consacrant à la programmation même s'ils ont une première carrière dans la recherche et pour la grande majorité une thèse. Le tableau des arrivées des principaux collaborateurs de IPython (avant 2015) permet non seulement de voir la progressivité de la constitution du projet, mais aussi son fort ancrage académique (et masculin) : en effet, la très grande majorité des contributeurs sont détenteurs d'un doctorat (Tableau 1).

---

47 https://data.berkeley.edu/news/project-jupyter-celebrates-20-years-fernando-perez-reflects-how-it-started-open-sciences Il fait ce choix avec une autre offre d'aller faire de l'ingénerie logicielle dans le secteur privé.

48 La syntaxe du langage de programmation évoluant elle-aussi.

Personne	Année d'arrivée	Doctorat	Discipline	Activité pro en 2020
Fernando Perez	2001	yes	physics	academic
Brian Granger	2004	yes	physics	academic
Benjamin Ragan-Kelley	2004	yes	engineering physics	academic
Thomas Kluyver	2010	yes	plant science	academic
Paul Ivanov	2010	yes	computational neuroscience	software engineer
Jason Grout	2011	yes	mathematics	academic/software engineer
Matthias Bussonier	2012	yes	physics	software engineer
Kyle Kelley	2012	no	computer science	software engineer
Damian Avila	2012	no	biochemistry	data scientist
Jess Hamrick	2013	yes	psychology	private research scientist
Jonathan Frederic	2013	no	physics	software engineer
Sylvain Corlay	2014	yes	mathematics	ceo/software engineering

Tableau 1. Principaux collaborateurs au projet IPython avant 2015

2.2 La réussite de l'implémentation d'un format notebook

En 2011, une solution de notebook est proposée par Perez, Granger et Ragan-Kelley[49]. L'arrivée de ce nouveau développement n'est pas une surprise : l'idée d'un notebook pour IPython était présent dès le début du projet. Les deux développeurs soulignent que des références à Mathematica et Maple, dotés d'interfaces, étaient présents dès la première version : « *that's a very natural environment to work in for scientists* ». Par contre, la tentative de créer un format notebook avait échoué régulièrement depuis une dizaine d'années. Plus précisément, cinq tentatives précédentes n'avaient pas abouties, la première en 2005 lors du *Google Summit of Code*[50]. Une des raisons avancées par les auteurs est que les technologies modernes du web n'étaient pas disponibles, comme les web sockets, ce qui limitait la possibilité de créer des concepts d'abstractions permettant de construire un tel format. Les caractéristiques techniques d'un web 2.0 plus mature et interactif étaient nécessaires, par exemple sur les avancées des technologies associées au langage *javascript*. Cependant, ces différents échecs ont permis d'identifier les mauvaises solutions, et les trajectoires parallèles d'autres dispositifs similaires dans des logiciels scientifiques, comme Sage, ont accompagné certains choix.

De nombreux échanges ont lieu sur les emprunts et les évitements qui permettent de stabiliser un format. Par exemple, la logique de programmation scientifique amène à penser les notebooks comme une interface ouverte permettant de contrôler l'ensemble du système informatique et ne pas être piégé uniquement dans un logiciel comme peuvent l'être les interfaces scientifiques existant par ailleurs[51]. Cela conduit en décembre 2011 à la première version publique des notebooks IPython

[49] Le code du premier client est ici : https://github.com/fperez/zmq-pykernel

[50] https://wiki.python.org/moin/SummerOfCode/2005

[51] « This is a key difference of our approach and the Sage notebook, so it' worth clarifying what I mean: the Sage notebook took the route of using the filesystem for notebook operations, so you can't meaningfully use 'ls' in it or move around the filesystem yourself with 'cd', because sage will always execute your code in hidden directories with each cell actually being a separate subdirectory. This is a perfectly valid approach and lets the notebook do many useful things, but it is also very different from the ipython model where we always keep the user very close to the filesystem and OS. For us, it's really important that you can access local scripts, use %run, see arbitrary files conveniently, as in data analysis and numerical simulation we make extensive use of the filesystem. So the sage

« *On December 21 2011, we released IPython 0.12 after an intense 4 1/2 months of development. Along with a number of new features and bug fixes, the main highlight of this release is our new browser-based interactive notebook: an environment that retains all the features of the familiar console-based IPython but provides a cell-based execution workflow and can contain not only code but any element a modern browser can display [...] For the IPython project this was a major milestone, as we had wanted for years to have such a system, and it has generated a fair amount of interest online.* » (Perez, blog, 1/08/2012).

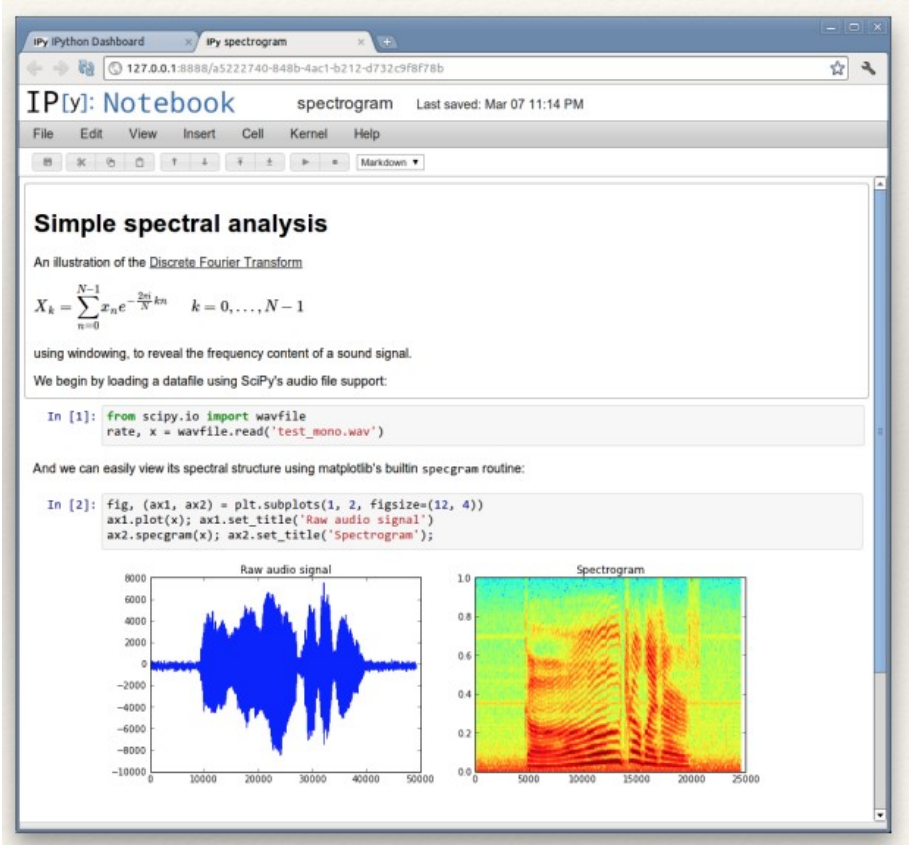

Figure 1. Exemple de notebook Ipython, issu d'une présentation de Fernando Perez - https://indico.ijclab.in2p3.fr/event/2987/contributions/6823/attachments/6364/7511/jupyter-bids-paris-oct-2015-fperez_1.pdf

L'arrivée des notebooks bénéficie de la visibilité importante de IPython. En 2012, Fernando Pérez reçoit le prix de la Free Software Foundation pour l'avancement du logiciel libre pour son travail sur Ipython largement diffusé dans la communauté scientifique. En retour, l'adoption de plus en plus large de IPython met à rude épreuve la capacité des porteurs du projet à maintenir un logiciel correspondant à tous les usages. Cela conduit en retour à investir dans la modularité et l'adaptabilité, qui se retrouvent dans de nombreux projets open source[52]. Cette évolution est accompagnée de nombreuses nouvelles collaborations, notamment avec le portage du projet sur

model wasn't really a good fit for us. Furthermore, we wanted a notebook that would provide the entire 'IPython experience', meaning that magics, aliases, syntax extensions and all other special IPython features worked the same in the notebook and terminal. The sage nb reimplemented some of these things in its own way [so] In some cases it's almost like ipython, in others the behavior is fairly different, which is fine for Sage but doesn't work for us. » (Perez, blog 1/08/2012)

52 Le cas d'EMACS, éditeur de texte console, dont le succès est de pouvoir être extensif et adaptable avec des modules, qui ensuite deviennent intégrés à l'éditeur. Cette croissance modulaire est ainsi déjà largement présente.

Github[53] permettant d'attirer des contributeurs dans une logique du logiciel libre. De nouveaux collaborateurs, comme Matthias Bussonnier, initialement contributeur bénévole pendant qu'il finissait sa thèse, rejoignent le projet sur des financements académiques.

Cette explosion de demandes et de contributions conduit en retour à devoir formaliser la logique du projet et les conditions de la collaboration : « *Since at this point we have too many feature requests from multiple fronts to be able to satisfy them all, we are trying to focus on ensuring that IPython can support indivdual projects building their own custom tools and extensions. We can't possibly merge every last idea from every front into IPython, but we can work to ensure it's a flexible and coherent enough foundation that others can build their own highly customized experiences on top* » (Perez, blog, 11/20/2012). Les enjeux amenés par le maintien du code face à la multiplication des contributions et des propositions de nouvelles fonctionnalités est un puissant moteur de formalisation du projet (Alcaras, 2022), favorisant en retour l'arrivée de nouveaux usages au sein du collectif. Cela se traduit par un déplacement plus fort vers les bonnes pratiques de l'ingénierie logicielle, la formalisation des fonctionnalités du logiciel, de même que par un engagement spécifique dans la réflexion sur les méthodes. Ainsi, Fernando Perez participe à l'obtention d'un financement auprès de la fondation Alfred P. Sloan et de la fondation Gordon and Betty Moore pour créer le Berkeley Institut For Data Science (BIDS) en 2013. Il devient chercheur associé au BIDS à partir de 2014, dirigeant le groupe *Software Tools and Environments* et staff scientist à la Computational Research Division de l'université, soulignant aussi la transition d'une activité de recherche de spécialité vers l'instrumentation. Ses communications et publications relevant soit du calcul numérique en collaboration, soit des solutions logicielles soulignent cette situation d'intermédiarité, avec une tendance à se déplacer vers l'instrumentation. Cette transition est aussi très visible dans les intitulés des projets de recherche déposés et des financements obtenus[54].

2.3 La séparation progressive avec la communauté Python

Une philosophie générale de la programmation d'abord aux services des chercheurs puis plus généralement de la science des données se stabilise progressivement avec le développement d'IPython. Un constat répété est que l'usage de la programmation par les chercheurs diffère largement de celle des développeurs informatiques. Dans une demande de financement en 2015, les porteurs du projet écrivent ainsi que « w*hile scientists have always used computers as a research tool, they use them differently than industrial software engineers: in science, the computer is a kind of "abstract microscope" that enables the scientist to peek into data and models that represent or summarize the real world*. » (demande de financement, 2015). Venant initialement de personnes impliquées dans la recherche, cela signifie surtout être adapté aux problématiques rencontrées dans le travail d'exploration et de consolidation de la preuve, aider l'humain à réfléchir, et de ne pas imposer a priori trop de contraintes venant de la formalisation informatique. La recherche d'interactivité se trouve au cœur des choix, permettant de mettre le plus possible « l'humain dans la boucle », en insistant d'une part sur l'interactivité, de l'autre sur la possibilité de construire un récit cohérent pour soi et pour les autres dans la perspective ouverte de la programmation lettrée. Plus encore, l'enjeu est moins la programmation pour elle-même qu'au service de l'analyse des données – emportant ainsi une dimension computationnelle. Se dessine les contours d'une computation lettrée (litterate computing) associant les logiques de la programmation lettrée et les enjeux de traitement des données. Les choix de programmation et d'interface sont fait dans cet objectif : « *To*

[53] La période 2005-2010 est un import du dépôt SVN de IPython.

[54] Si le premier financement de la NSF auquel il est co-PI en 2006 s'intitule « Fast Multiresolution Methods and Nonlinear Approximations for Multidimensional Problems », les suivants portent sur la programmation scientifique ou à l'interface avec le traitement numérique, avec la Fondation Simons en 2013 « Scientific neuroimaging: sharing and transparency », ou Alfred P. Sloan « An Open Source Framework for Interactive, Collaborative and Reproducible Scientific Computing and Education ».

answer this, let's flip the REPL (read-eval-print Loop) around and cast it from the perspective of the human user. The user has a counterpart to the computer's read-eval-print-loop: a "write-eval-think-loop" (WETL). » (Granger et Pérez, 2021). Cette idée initiée dans le contexte de recherche marque la philosophie du projet et contribue à son extension au dehors des frontières académiques.

Cette formalisation progressive d'une philosophie générale qui dépasse le logiciel contribue à le constituer en une organisation. En 2015, les auteurs déclarent « *we have quite ambitions plans for the future* », soulignant que l'enjeu est de pérenniser les financements et d'organiser la communauté. A ce stade, de nombreuses collaborations existent et ancrent IPython et les notebooks au-delà du monde académique[55]. Une étape décisive d'extension de ces outils est atteinte avec le « Big Split » annoncé le 16 avril 2015[56], correspondant à une explicitation de l'indépendance entre un langage de programmation et sa communauté de pratique avec la solution des notebooks développées depuis 2011.

Cette évolution est très visible dans l'activité de la liste de diffusion historique IPython-dev où les mails concernant les notebooks gagnent progressivement en visibilité, et l'apparition progressive du nouveau nom du projet, Jupyter (Figure 2). Ce nom, en plus d'insister sur la proximité entre les notebooks computationnels et les carnets d'observation de Galilée découvrant les lunes de Jupiter, qui se retrouve dans le logo, souligne les trois principaux langages de programmation scientifique utilisés par les chercheurs : Julia, Python et R, qui deviennent des noyaux (*kernels*) appelés dans un logiciel plus générique qui se concentre sur trois objectifs : l'interactivité, la narration et la collaboration. « *Those langages are not ennemies, the ennemies are closed science* ». Cette étape prend place en même temps que l'obtention d'un financement conséquent de six millions de dollars par le Helmsley Trust, le Gordon and Betty Moore Foundation et le Alfred P. Sloan Foundation pour un projet qui associe la programmation scientifique avec plus généralement la data science : "*Project Jupyter's mission is to create open source tools for interactive scientific computing and data science in research, education and industry, with an emphasis on usability, collaboration and reproducibility*" (grant, 2015).

[55] Dans la demande de financement écrite en 2015, les coordinateurs du projet égrainent ces collaborations dans et hors de champ académique : « we have ongoing collaborations with individuals and departments at Stanford, UW, NYU, MIT, Harvard, Bryn Mawr, U. Southampton, U. Sheffield and Simula Research Lab (Norway). In the area of open science, we coordinate efforts with the Center for Open Science (Brian Nosek and Jeff Spies) and Software Carpentry (Greg Wilson). In traditional journalism, we have relationships with staff at 538, BuzzFeed and the New York Times focused around data-driven journalism. In open source software, we collaborate closely with the core developers of all the major scientific computing and data science projects in Python (NumPy, SciPy, Pandas, Matplotlib, Scikit-Learn, etc.), Julia (core developers) and R (rOpenSci)."

[56] https://blog.jupyter.org/the-big-split-9d7b88a031a7

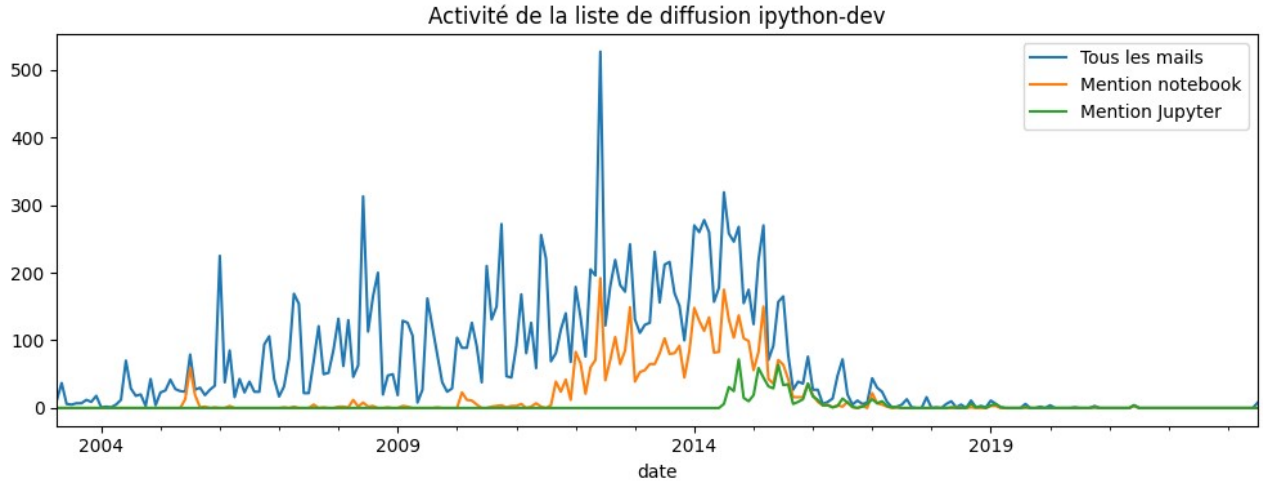

Figure 2. Evolution des mails sur la liste de diffusion Ipython-dev

3 Des notebooks académiques à une infrastructure de la data science

Le changement de nom du projet marque aussi un changement d'échelle à plusieurs niveaux de l'usage de notebooks computationnels, pour ses promoteurs mais aussi pour sa diffusion à l'échelle internationale hors du monde académique.

3.1 Le succès des notebooks en dehors du monde académique

Le passage de IPython vers Jupyter a été un événement remarqué[57] et, symboliquement, il marque la rupture avec son ancrage initial très fort dans la programmation scientifique Python, largement associée au monde scientifique. Si l'article princeps publié en 2016 débute par le constat que les chercheurs de toutes les disciplines sont amenés à écrire du code (Kluyver et al., 2006), les principaux utilisateurs actuellement se trouvent au-delà du monde académique. Les notebooks computationnels sont progressivement devenus une référence dans l'ensemble des activités étiquetées « data science », au croisement entre le monde de la recherche scientifique, le domaine de l'open source et le secteur privé de l'analyse des données porté par les grands acteurs du numérique. Si d'autres formats de notebooks computationnels existent, ceux du projet Jupyter sont de loin les plus visibles, en témoigne que la littérature académique traitent presque exclusivement d'eux. Ils ont reçu d'ailleurs le *2017 ACM Software System Award*[58], un prix annuel qui récompense des personnes ou une organisation « *pour le développement d'un système logiciel qui a eu une influence durable, reflétée dans les contributions aux concepts, dans l'acceptation commerciale, ou les deux* ».

Comment rendre compte de ce succès ? Il marque une intégration progressive de la philosophie des notebooks computationnels à l'intersection entre l'université, l'open source et le monde industriel. Comme le souligne Fernando Perez en 2021 « *Jupyter exists at the intersection of distributed open source development, university-centered research and education, and industry engagement. While the original team came mostly from the academic world, from the start we've recognized the value of engaging industry and other partners. This led, for example, to our BSD licensing choice, best articulated by the late John Hunter in 2004* ». L'arrivée du projet Jupyter est synonyme d'une

[57] https://www.lemondeinformatique.fr/actualites/lire-analyse-de-donnees-le-projet-ipython-evolue-vers-jupyter-62089.html

[58] https://en.wikipedia.org/wiki/Turing_Award

ouverture à un public beaucoup plus large de l'usage des notebooks. « *Jupyter aims to bring notebooks to a broader audience* ». Cette évolution a nécessité de nettoyer tous les reliquats spécifiques à Python. Cela a aussi été un passage de la programmation scientifique à la programmation interactive en général, qui n'est plus limitée au domaine académique : « *in the context of data science, you are not tring to build a software but build your understanding of the data and translate it to a narrative* ». Cette séparation été le moment d'une restructuration de l'organisation même du projet : la création d'un nouveau dépôt spécifique aux notebooks Jupyter a ainsi été créé en filtrant les éléments du dépôt précédent Ipython[59]. Le Big Split est bien évidemment plus graduel qu'une coupure nette : le passage de IPython à Jupyter est présentée comme « *a natural gradual realistion [...] everything we are doing here [...] there is nothing in here that is specific to Python ... what if we just abstract that over* ». Entre la période Python et la période générique, une partie des principaux collaborateurs reste[60].

D'un point de vue organisationnel, le projet a cependant beaucoup évolué. Comme le résume ses fondateurs lors du prix reçu en 2017, « *over the years, we evolved from the typical pattern of an ad-hoc assembly of interested people loosely coordinating on a mailing list to a much more structured project. We formalized our governance model and instituted a Steering Council*". IPython puis Jupyter s'ancre dans le fonctionnement open source avec une gouvernance communautaire. D'un point de vue juridique, le projet est porté par la fondation d'intérêt publique NumFocus (« Numerical Foundation for Open Code and Useable Science. ») qui vise à « *promote open practices in research, data, and scientific computing by serving as a fiscal sponsor for open source projects and organizing community-driven educational programs*. ». Mais ce qui est largement mis en avant, c'est la constitution d'un collectif qui utilise et développe ses outils – le public récursif de Christopher Kelty qui participe en permanence à constituer les infrastructures de l'activité collective de transformation des données.

L'extension du projet Jupyter au-delà des frontières du monde académique s'inscrit dans la directe continuation de collaborations déjà existantes entre des acteurs issus de mondes différents, qui évoquent une triple hélice université-gouvernement-secteur privé (Etzkowitz et Leydesdorff, 2000), avec comme jointure le monde de l'open source : « *Everything we describe here rests on a 20-year open collaboration where stakeholders from academia, industry, government, and more have participated as peers* » (Granger et Perez, 2021). La construction de cette communauté très spécifique se donne à voir non seulement par les sources de financement obtenues par le projet mais aussi par la composition des intervenants dans les grands rassemblements communautaires que sont les conférences Jupytercon. De nombreux acteurs majeurs de l'analyse des données, comme Google, Amazon ou Microsoft ont participé au développement du projet Jupyter et l'intégration de ces formats dans leur services. Ces acteurs sont présents dans les comités de gouvernance et participent directement à l'embauche du personnel[61]. Si l'origine est bien académique, sa trajectoire et son adoption massive le rapproche d'autres domaines. Les sources de financements rendent visible cette imbrication entre les trois hélices : le projet est financé via NumFocus par Microsoft, bénéficie de ressources de calcul de Rackspace, et de financements philanthropiques de la recherche

[59] Ainsi, le premier commit du dépôt notebook du projet Jupyter est le même que celui présent dans le dépôt de Ipython : commit ad18848364e5e4073f80b03c64b6fa4bc1dcfbda et commit commit 6f629fcc23ba63342548f61cc7307eeef4f55799

[60] Les 10 principaux contributeurs aux dépôts officiels (20/10/2023) comprennent des membres historiques et des contributeurs spécifiquement arrivés avec le projet Juyter : minrk (Benjamin Min):7098 ; Carreau (Matthias Bussonnier): 4857 ; takluyver (Thomas Kluyver):3396 ; jhamrick (Jess Hamrick): 2416 ; jdfreder (Jonathan Frederic) : 2275 ; ellisonbg (Brian Granger) : 1624 ; blink1073 (Steven Silvester): 1602 ; mathbunnyru (Ayaz Salikhov) : 1437 ; vidartf (Vidar Tonaas Fauske) : 1415 ; parente (Peter Parente) : 1312

[61] « These community building efforts, including the hiring of an Event Manager, is made possible through significant donations from our partners at Bloomberg and AWS. » https://blog.jupyter.org/jupyter-community-2021-update-84c5cd3c5e75

comme la Alfred P. Sloan Foundation (1,15M en 2014 pour la création de nbconvert, les widgets et un prototype de Jupyterhub) ou de la Simons Foundation (100k pour du partage de données en neuroimagerie), de Google (100k en 2014 pour le postdoctorat de Mathias Bussonnier sur l'intégration des notebooks dans Google Drive), Continuum (100k en 2015). De nombreux contacts sont pris avec les principaux acteurs du domaine qui sont présents aux conférences dédiées de la communauté : Google, Microsoft, Bloomberg, Continuum, Quantopian, West Health, IBM. Le financement important annoncé en juillet 2015 de 6M par trois fondations réunies permet de stabiliser le projet, par exemple par l'embauche d'un premier project manager en 2015.

Cette composition est importante, car ce sont les contributeurs et utilisateurs qui fixent la trajectoire de cet outil libre. Comme le soulignent les fondateurs, qui conceptualisent cette communauté comme une « communauté de pratique » : « *This community is not accidental: the core Jupyter team has invested significant effort into welcoming new contributors, helping users, planning and running community events (Jupyter CommunityWorkshops24, JupyterDays and JupyterCon25), and training and mentoring junior developers and designers* ». Cette communauté de pratique largement dématérialisée et répartie sur la planète devient visible lors des événements clés, comme les conférences Jupyter (Jupytercon)[62]. Ces moments réunissent des chercheurs titulaires et non titulaires dans de nombreuses spécialités (une des conférences magistrales a ainsi été donné par l'économiste Paul Romer), des ingénieurs informatiques, des data scientists, des entrepreneurs de compagnies dans le numérique, des représentants des grands groupes comme Amazon, des consultants et des contributeurs de l'open source[63]. Pour faire tenir tous ces intérêts différents, un travail important est mis en oeuvre autour de la gouvernance. Celle-ci est passée par différentes étapes, et a donné lieu à une refonte en 2023 avec l'accumulation de tensions[64] (dispersion des dépôts, etc.).

3.2 L'écosystème Jupyter comme une infrastructure de connaissances

Depuis les années 2000, la trajectoire a donc été une montée en généricité, permettant de réunir des pratiques par ailleurs séparées. Il est ainsi devenu un instrument générique qui dépasse les situations spécifiques d'usage, et à ce titre a intégré des secteurs économiques éloignés de son contexte de genèse.

Une des conséquences de l'adoption massive du logiciel notamment dans le secteur privé est la contrainte de stabilité : à la fois assurer la compatibilité avec le passé, mais aussi être réactif par rapport aux besoins de la communauté. Ce travail nécessite l'engagement pérenne de développeurs, surtout dans un monde dans lequel les briques technologies sont susceptibles d'évoluer rapidement[65]. Comme l'écrivent les membres du projet, « *when we first started working on IPython in the early 2000s, this workflow was mostly foreign to developers in the traditional software engineering world* »[66] . Par ailleurs, cela amène la conduite d'enquêtes auprès des utilisateurs pour faire remonter les besoins, et la tentative d'intégrer les demandes. Cela ne se fait pas sans difficulté. Ainsi, la collaboration interactive sur les notebooks est sollicitée pendant de nombreuses années

[62] Mis comme objectif dans la demande de financement obtenue en 2016, ces événements se sont tenues d'abord en 2017 et 2018 à New-York en collaboration avec O'Reilly Media, puis en ligne en 2020 à cause de l'épidémie de COVID-19 et en 2023 à Paris.

[63] Pour avoir les membres de la session de 2023 : https://cfp.jupytercon.com/2023/speaker/. Un travail systématique reste à faire pour caractériser ces conférences.

[64] Cet article n'a pas pour objectif de faire une analyse fine de l'évolution de la gouvernance de Jupyter. Celle-ci reste donc à faire.

[65] Un témoignage publié sur le blog de Jupyter de la difficulté de maintenir une extension non-officielle à travers les évolutions des infrastructures rend compte en creux du travail fourni par l'équipe pour assurer le fonctionnement des outils : https://blog.jupyter.org/the-continued-existence-of-the-emacs-ipython-notebook-54bd1c371d57

[66] https://www.oreilly.com/radar/the-state-of-jupyter/ 2017

mais rencontre des difficultés techniques à être mise en œuvre, et ne sera annoncée qu'en 2021[67]. Pour répondre aux nouveaux usages, de nouveaux outils sont aussi développés et stabilisés. En juillet 2016, l'équipe annonce ainsi le Jupyterlab développé en collaboration avec des partenaires privés[68], qui permet d'étendre l'interface d'un notebook à un ensemble d'outil plus vaste et se rapprocher d'un IDE au sens de l'ingénierie logicielle. La montée d'un usage en tant que service et les besoins liés à l'éducation conduisent au développement de JupyterHub qui permet de déployer le service sur un serveur afin d'offrir la possibilité d'exécuter les outils Jupyter en ligne. Autour de ces logiciels se développe toute une série d'outils de plus petite taille permettant de compléter les usages et de favoriser les transitions vers d'autres services.

> *"In reality, even today's "Jupyter Notebook" is a bit of a misnomer: the Notebook application includes not only support for Notebooks but also a file manager, a text editor, a terminal emulator, a monitor for running Jupyter processes, an IPython cluster manager and a pager to display help. And that is just what ships "out of the box", without counting the many third-party extensions for it. This rich toolset evolved organically, driven by the needs of our users and developers, even if we kept the increasingly ill-fitting "Notebook" name for the whole thing. "*[69]

En raison de leur caractère hybride entre format et logiciel, ils ont largement débordés les frontières du logiciel à proprement parler. La croissance d'usage et l'intégration du format notebook dans une diversité d'autres outils semble avoir facilité le passage d'un instrument délimité à une véritable infrastructure de connaissances[70]. Cela ne signifie pas que les enjeux propres aux notebooks d'un point de vue d'instrument logiciel disparaissent, mais que ceux-ci et les services associées perdent en partie leur singularité pour être davantage intégrés et maintenus dans le cadre d'autres activités au sein d'un « écosystème »[71] clairement revendiqué comme objectif[72]. Dans ces évolutions se retrouve une des tensions inhérentes aux mondes de l'open source, encore plus amplifiée par des communautés scientifiques : la plasticité des outils permettent de nombreuses modifications, changement, et au final chacun finit par utiliser des outils différents. Dans le cas du projet Jupyter, cette plasticité se traduit par la multitude d'extensions qui peuvent être installées, et dont le développement est prévu par les concepteurs.

De fait, Jupyter regroupe de nombreux sous-projets qui permettent de l'intégrer dans un écosystème plus large de logiciels open source. Ceux-ci sont systématiquement présentés comme élément du projet (Granger et Perez, 2021), et interrogent en retour la portée du découpage en terme de logiciel d'un ensemble d'outils connectés open source. La figure 3 représente les liens entre les sous-projets

---

67 « Right from the beginning, collaborative editing was on the agenda for Jupyter Notebooks. In 2012, core Jupyter contributor and creator/lead of JupyterHub, @minrk, wrote in the GitHub issue tracker: [..] This will finally make decent live collaboration feasible, which is our single most-requested and highest-priority new feature. ». En effet contrairement à une collaboration sur un document texte, il est nécessaire de créer une abstraction partagée des données.

68 « This effort is the fruit of an open collaboration between our industry partners at Bloomberg and Continuum and the Jupyter Team anchored at UC Berkeley/LBNL and CalPoly, funded by the Helmsley Trust, the Gordon and Betty Moore Foundation and the Alfred P. Sloan Foundation. We are extremely grateful for this support, and we hope to see in the future many more examples of similar partnerships between academia, philantrophic funders and industry »

69 https://blog.jupyter.org/jupyterlab-the-next-generation-of-the-jupyter-notebook-5c949dabea3

70 Edwards et al. (2013) définit les infrastructures de connaissances comme « robust networks of people, artefacts, and institutions that generate, share, and maintain specific knowledge about the human and natural worlds. »

71 Pour avoir une idée de cet écosystème et de ses interdépendances, des aspects les plus visibles (les services) à ceux les plus abstraits (les protocoles) : « Navigating the Jupyter Landscape | JupyterCon 2023 », Jeremy Tuloup, Johan Mabille, qui soulignent la difficulté d'avoir la vision générale https://www.youtube.com/watch?v=uWJ0-OPKTxI&ab_channel=JupyterCon

72 https://www.oreilly.com/radar/the-state-of-jupyter/

officiels regroupés dans l'organisation Jupyter sur Github. Le format en lui-même est inutile sans l'application Jupyter notebook, prolongée ensuite par le Jupyter Lab qui se rapproche d'un environnement davantage intégré de programmation. Le partage des notebooks est facilité par la pratique antérieure des dépôts sur des forges comme Github, qui implémente de son côté un rendu pour les notebooks afin de les consulter en statique. Pour permettre d'exécuter des notebooks, les contributeurs du projet participent à développer le service Binder.

Sans rentrer dans un travail d'épuisement de cet ensemble d'outils interdépendants, le point clé est de souligner que les usages s'appuient sur un travail permanent d'adaptation de l'idée initiale à d'autres logiciels. Ainsi les porteurs du projet peuvent écrire : « *Jupyter is more than merely software. More specifically, Jupyter also builds and consists of services, open standards and protocols, and community* » (Granger et Perez, 2021). Cette infrastructure vaut tant pour l'activité scientifique que plus généralement les pratiques de traitement des données. Ainsi, les notebooks intègrent rapidement l'environnement de biologie des systèmes Kbase en facilitant la connexion entre les différents modules[73]. Lors de la création du Journal of Digital History, ils s'imposent comme le format le plus fédérateur pour construire la trame computationnelle[74]. Le blog de Jupyter est une vitrine de présentation de l'intégration des outils dans différentes sous-communautés[75]. Les nombreuses extensions construites par les différentes communautés représentent des vecteurs de diffusion des outils Jupyter qui deviennent de véritables infrastructures[76]. Ils deviennent un support évident pour tous les formats d'apprentissages, de l'usage dans les cours aux démonstrateurs de nouvelles bibliothèques de programmation. Cette hybridation se retrouve à différents niveaux, notamment des logiciels tiers qui viennent eux-aussi intégrer les notebooks – comme Visual Studio Code, l'environnement de développement (IDE) le plus utilisé en programmation, dont l'extension spécifique pour les notebooks est plébiscitée. Les grands prestataires de service, comme Amazon ou Google, font des notebooks une des interfaces d'entrées de la possibilité de déployer des traitements d'apprentissages automatiques. Cela accroît alors le passage des notebooks d'un logiciel à un service, participant à ce titre à la plateformisation de la recherche déjà notée par d'autres travaux sur les supports numériques de communication académiques (Plantin et al., 2018).

---

73 « Another benefit of KBase's collaboration with Pérez is connecting systems biology researchers to Jupyter developers that are building open source tools and libraries that allow scientists to leverage commercial cloud computing resources to build on Jupyter notebooks published outside of KBase. » https://cs.lbl.gov/news-media/news/2021/project-jupyter-a-computer-code-that-transformed-science/

74 « After a long phase of studying different systems and editorial solutions, we finally decided to choose Jupyter notebooks as an editor for our journal. Becoming more and more popular, notebooks allow notable reproducibility of code and, once passed a small learning curve, an easy to use code and text editor. Their lack of structure for text cells can be compensated by the use of the markdown language, easy to learn and very flexible. » (Fickers et Clavert, 2021).

75 Ainsi, en 2019, un billet sur une bibliothèque de robotique illustre bien ce mécanisme de diffusion progressive par co-construction d'outils : « Project Jupyter is a huge hit in data science, but it has not yet found widespread adoption in robotics. Today, we are releasing the first version of jupyter-ros, a collection of Jupyter interactive widgets inspired by Qt and RViz, to bring their features to the Jupyter ecosystem. This may be the right time for Jupyter-based developer tools, as cloud robotics is taking off ».

76 Pour les sciences sociales, pour lesquelles le mouvement est plus récent, on peut noter par exemple la bibliothèque ipysigma permettant de faciliter la visualisation et l'exploration interactive de réseaux dans un notebooks, et qui dans les faits importe des outils développés ailleurs dans cet environnement pour une meilleure synercie (Plique, 2021).

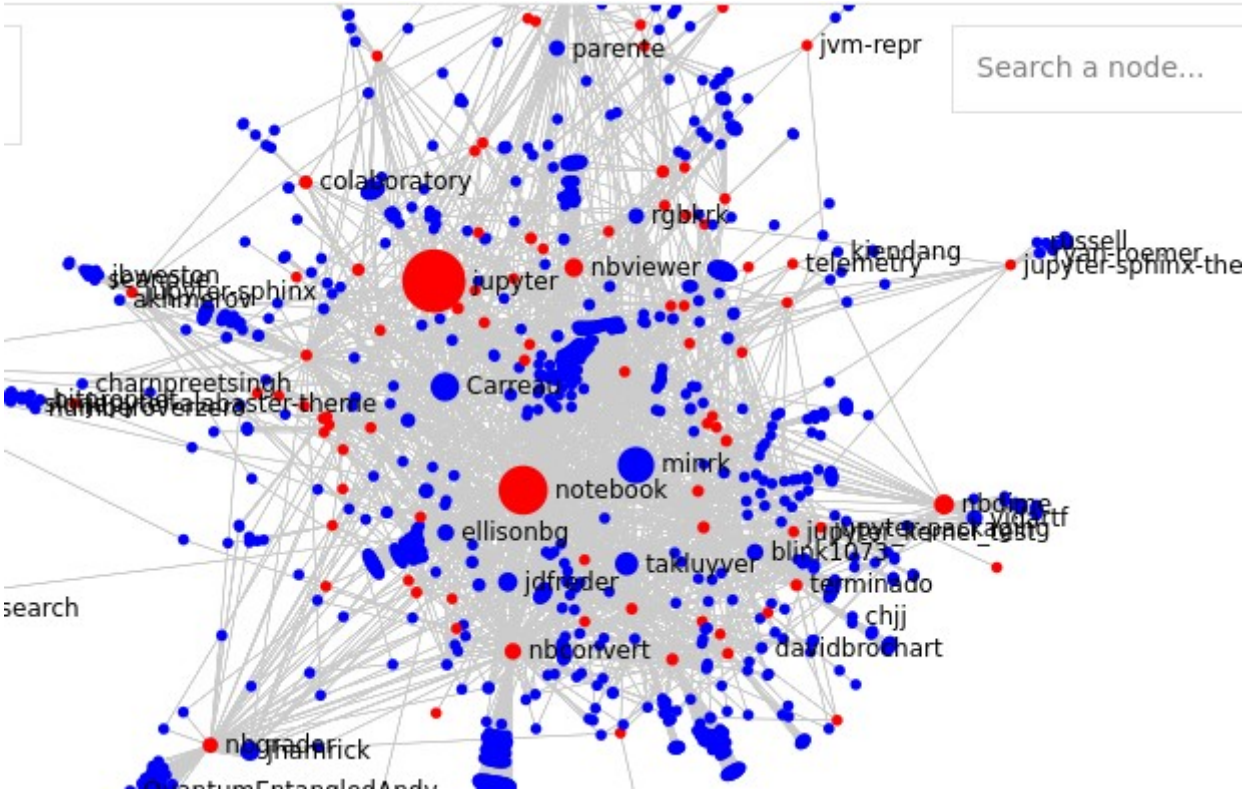

Figure 3. Analyse des contributions aux dépôts Github de l'organisation Jupyter. Les nœuds représentent les dépôts officiels (rouge) et les contributeurs (bleu), les liens représentent le nombre de contribution dans chaque dépôt.

4. Conclusion : flexibilités et résistances d'un format ouvert

La trajectoire du projet Jupyter esquissé dans cet article pose un ensemble de questions imbriquées. Pour la *sociologie de l'innovation*, elle pose la question générale des étapes de diffusion et d'appropriation d'une technologie, et des acteurs impliqués dans le cadre d'une co-construction permanente avec une communauté open source alliant contributeurs bénévoles et professionnels, rejoignant à ce titre des réflexions déjà initiées (Alcaras, 2022). Pour la *sociologie du numérique*, cette évolution apparaît comme un traceur du déploiement des métiers autour des données et de leurs infrastructures et normes, mais aussi de la transformation des pratiques de l'open source ou encore de la place croissante des navigateurs et des services. Enfin, pour *les études sur les sciences et les techniques*, cette trajectoire met en lumière la place du logiciel dans les pratiques de recherche, notamment autour de la programmation scientifique, et des conditions de standardisation des usages dans des outils génériques, mais renvoie aussi plus généralement aux promesses d'une relation entre science ouverte et open source (Mirowski, 2018).

Dans une perspective de diffusion des innovations techniques, Jupyter témoigne non seulement de l'importance de la co-construction des outils avec les communautés d'usagers, mais aussi du déplacement de ces communautés. Sans surprise à la lumière des études sur les techniques, cette trajectoire rend visible l'interdépendance entre un dispositif et son écosystème plus large : des navigateurs aux technologies *javascript* disponibles en passant par les autres logiciels utilisés majoritairement comme VS Code pour le développement logiciel, le déploiement des notebooks computationnels a accompagné des transformations plus larges sur les pratiques qui ont nécessité la contribution de nombreux acteurs, par exemple la visualisation des notebooks sur Github, et la

création de nombreux services accessoires permettant l'intégration. Cependant, cette trajectoire de diffusion pose la question des ontologies suivies : le passage d'un programme à un logiciel, puis à un projet et à une infrastructure met à l'épreuve la bonne focale d'analyse. A quel moment déclare-t-on un logiciel comme infrastructure ? Comment rendre compte de la co-évolution entre un format et son implémentation logicielle ? Son caractère évolutif est un enjeu à documenter, visible dans le cas de Jupyter avec des déplacements de communautés. Le logiciel comme un objet historique qui évolue peut être un problème pour fixer l'objet d'étude dans une enquête de sciences sociales, même si dans certains cas le code permet de revenir sur ces évolutions (Ermoshina, 2017).

Dans la perspective d'une sociologie du numérique et des logiciels, la trajectoire de Jupyter pose largement la question d'une transformation des pratiques existantes mais aussi la création de nouvelles communautés d'usagers. Ces nouvelles pratiques émergeant progressivement autour des notebooks participent à réunir des professionnels par ailleurs appartenant à des mondes différents, faisant se rencontrer des chercheurs en biologie avec des spécialistes de l'analyse de données en marketing. Cela amène à penser les pratiques communes qui dépassent les divisions habituelles, avec le risque d'imposer des pratiques hétéronomes à certains domaines notamment celles venant du monde l'informatique industrielle. La littérature en ingénierie logicielle a fait fleurir une série de réflexions sur les nouvelles pratiques amenant à proposer des règles de « bonne pratique » visant à standardiser les usages. Suivant les communautés concernées, le focus sera placé soit sur la robustesse informatique, soit sur la reproductibilité des résultats, ou encore la qualité de la narration. Un travail de démarcation notamment prend place entre les bonnes pratiques de développement logiciel déjà existantes, et les nouvelles pratiques introduites par les notebooks, initialement pensées dans le contexte exploratoire de la recherche scientifique : « *Jupyter users like to experiment in the notebook, and to use the notebook as an interactive communication tool. However, for more classical software development tasks such as the refactoring of a large codebase, they often switch to general-purpose IDEs*. » (Perez, blog, 2020). Ainsi, la critique des limites des notebooks devient un geste fréquemment répété : « *Tout puissant et utile qu'il soit, Jupyter a tout de même quelques limites qu'il faut prendre en compte*. »[77] Pour le cas des Notebooks Jupyter, la flexibilité des usages favorise la diversité des approches, et en retour une réflexion des utilisateurs sur les bonnes pratiques. Cela prend de nombreuses formes: des critiques adressées aux limites du logiciel, la proposition de règles à suivre, voire le développement d'outils dédiés visant à contraindre les bons usages. Il est bien évidemment possible de lister ces critiques. Elles concernent par exemple la non linéarité de l'exécution du code, la difficulté de gérer du versionnement, ou encore la dimension incomplète des notebooks diffusés dans l'espace public qui se retrouvent être non reproductibles (Pimentel et al, 2021). Les notebooks apparaissent dans un paysage où des bonnes pratiques de programmation scientifique sont déjà en place, et dans les faits viennent contrevenir à certaines d'entre elles (Wagemann et al., 2022). Cela appelle à une meilleure connaissance des pratiques logicielles en général, et des notebooks en particulier,notamment dans des collectifs liés à la production des connaissances[78].

Enfin, dans une perspective des études sur les sciences et les techniques, Jupyter rend visible la constitution des infrastructures permettant la manipulation des données qui modifient l'affordance même des autres outils numériques existants. Comme le montre Vertesi pour l'usage d'Excel et Powerpoint par la NASA, les possibilités ouvertes par des logiciels permettent des formes de coordination spécifique, et rendent ainsi possible d'accomplir de nouvelles activités mais aussi de nouveaux échanges, entre des communautés potentiellement peu connectées (Vertesi, 2019). Il attire

[77] https://www.lemondeinformatique.fr/actualites/lire-comment-jupyter-notebook-facilite-l-analyse-de-donnees-74605.html

[78] « Studying software in scientific and technical organizations requires adopting an approach to representation and documentation in which the electronic documents in question are not only the result of work, but the sites of work as well […] As the center of collective attention, they are the place where work is done » (Vertesi, 2019).

aussi largement l'attention sur la place du logiciel libre comme infrastructure de la science, qui mériterait un regain d'attention (Lin, 2005). En effet, une des perspectives ouvertes par le caractère ouvert de Jupyter est sa participation à la création de communs pour la science ouverte. Cet enjeu est explicitement au coeur des propos de ses créateurs, et des choix effectués dans la gouvernance du projet : « *all of this work we've talked about is driven by software, but software that is deeply embedded in the specifics of scientific research [...] I think our agenda should be one where we: a) take software more seriously as a core element of science, and therefore teach our students how to build it in ways that complement the work of our computer science colleagues; b) develop research efforts to truly explore what is unique about the intersection of software and science; and c) reward the careers of those who do this kind of work at all stages -- from students to engineers, researchers and faculty -- and therefore fund their work with a comprehensive vision that goes beyond "publish a paper and throw away the code*." (Perez, 2021). Il rejoint le constat de Kelty sur l'effet structurant d'UNIX : « *The fact that UNIX spread first to university computer-science departments, and not to businesses government, or nongovernmental organizations, meant that it also became part of the core pedagogical practices of a generation of programmers and computer scientists* [...] » (Kelty, 2008, p.132). Dans la mesure où cette infrastructure est travaillée par des investissements privés, Jupyter s'est initialement déplacée d'un instrument de recherche à un bien industriel public (Alcaras, 2021). Ce constat est important car ces logiciels sont aussi, modulo leur flexibilité (Vertesi, 2019), vecteurs de normes. Ils participent directement au régime technico-instrumental spécifique au sein des mondes de la science dont les acteurs se concentrent sur la stabilisation et la normalisation des instruments, participant en retour à créer des standards partagés entre des spécialités très différentes, qui se retrouvent à façonner des secteurs économiques[79] (Marcovich et Shinn, 2011). Plus encore, ils créent une double interface : d'une part entre les spécialités utilisant les mêmes outils, et d'autre part avec les communautés productrices qui peuvent être soit liées à une activité économique, soit aux mouvements de l'open source. Ce monde de l'open source, très connecté à l'ingénierie logicielle et régi par des dynamiques spécifiques, se trouve donc interdépendant de l'activité de recherche scientifique. A ce titre, le notebook computationnel du projet Jupyter est venu à occuper une position « d'objet frontière » travaillé en permanence par sa communauté composite une position d'objet frontière, conduisant à une diversité de pratiques qui restent à documenter (Quaranta et al., 2022).

L'objectif n'était pas d'épuiser l'histoire, mais d'identifier les principaux acteurs impliqués et de donner à voir la complexité des dynamiques sous-jacentes. Il est évidemment nécessaire de mieux documenter la période récente du projet Jupyter et la multiplication des usages des notebooks dans de nombreux secteurs pour avoir une vision d'ensemble. Plus encore, l'analyse de cet article présente des limites dans la capacité de délimiter les caractéristiques sociales des acteurs impliqués. Il est difficile de caractériser en général les utilisateurs notamment car les usages sont très divers : une étude de cas plus spécifique sur une communauté scientifique, par exemple les humanités numériques ou la biologie computationnelle permettrait d'avoir une meilleure connaissance du rôle de ces logiciels. Pourtant, il me semble que l'histoire des notebooks Jupyter ouvre la voie à une réflexion plus générale sur la programmation scientifique et l'usage des logiciels dans la recherche (Joppa et al., 2013), d'autant que la sociologie des sciences a encore insuffisamment porté son regard sur les logiciels, dont la trace est même difficile à suivre même dans les publications scientifiques (Li et al., 2017), ce qui pose la question de la reconnaissance de leurs contributeurs (Ledford, 2016). Par ailleurs, le visibilité donnée à la science ouverte impose de davantage déplier les interdépendances entre monde de la recherche et monde de l'open source.

[79] Ce n'est donc pas une surprise que les promoteurs d'un logiciel libre pour simuler les données fiscales en France viennent du monde académique (Shulz, 2021).

Bibliographie

Alcaras, Gabriel. 2022. "Des Logiciels Libres Au Contrôle Du Code. L'industrialisation de l'écriture Informatique."

Alexandre, Olivier. 2023. *La Tech. Quand La Silicon Valley Refait Le Monde*. Seuil.

Arvanitou, Elvira-Maria, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Jeffrey C Carver. 2021. "Software Engineering Practices for Scientific Software Development: A Systematic Mapping Study." *Journal of Systems and Software* 172(Wilson): 110848. https://linkinghub.elsevier.com/retrieve/pii/S0164121220302387.

Beg, Marijan et al. 2021. "Using Jupyter for Reproducible Scientific Workflows." *Computing in Science & Engineering* 23(2): 36–46. https://ieeexplore.ieee.org/document/9325550/.

Boaventura Bomfim, Daniela, and Emilien Schultz. 2022. "Les Essais Cliniques, Des Instruments de Recherche Comme Les Autres ? Étude d'un Essai Clinique de Médecine de Précision En France." *Zilsel*.

Bowker, Geoffrey C, and Susan Leigh Star. 1999. Classification and Its Consequences *Sorting Things Out. Classification and Its Consequences*. http://lexus.ischool.utexas.edu/Winget_Megan/2010/Fall/INF381/Readings/BowkerStar_SortingChaps9-10.pdf.

Bowker, Geoffrey, Karen Baker, Florence Millerand, and David Ribes. 2010. "Toward Information Infrastructure Studies: Ways of Knowing in a Networked Environment." In *International Handbook of Internet Research*, , 97–117. http://www.springer.com/?SGWID=0-102-0-0-0.

Broca, Sébastien. 2018. "Matière et Territoire Dans La Culture Du Logiciel Libre." *Geographie Economie Societe* 20(1): 15–32.

Cardon, Dominique. 2019. *Cultures Numériques*.

Collins, Harry. 2004. *Gravity's Shadow. The Search for Gravitational Waves*. Chicago Press.

Delfanti, Alessandro. 2016. "Beams of Particles and Papers: How Digital Preprint Archives Shape Authorship and Credit." *Social Studies of Science* 46(4): 629–45.

Desrochers, Nadine et al. 2018. "Authorship, Citations, Acknowledgments and Visibility in Social Media: Symbolic Capital in the Multifaceted Reward System of Science." *Social Science Information* 57(2): 223–48.

Ermoshina, Ksenia. 2017. 206 Reseaux *Le Code Peut-Il Réparer Les Routes?*

Etzkowitz, Henry, and Loet Leydesdorff. 2000. "The Dynamics of Innovation: From National Systems and 'Mode 2' to a Triple Helix of University–Industry–Government Relations." *Research Policy* 29: 109–23.

Fickers, Andreas, and Frédéric Clavert. 2021. "On Pyramids, Prisms, and Scalable Reading." *Journal of Digital History* 1(1): 1–13.

Froger-Lefebvre, Juliette, Quentin Lade, Estelle Vallier, and Catherine Bourgain. 2023. "E-

Prescription and Invisible Work in Genomics in France." *Frontiers in Sociology* 8(June).

Gläser, Jochen. 2007. "The Social Order of Open Source." http://www.igi-global.com/chapter/handbook-research-open-source-software/21220/.

Granger, Brian E., and Fernando Perez. 2021. "Jupyter: Thinking and Storytelling with Code and Data." *Computing in Science and Engineering* 23(2): 7–14.

Grotov, Konstantin et al. 2022. 1 Proceedings - 2022 Mining Software Repositories Conference, MSR 2022 *A Large-Scale Comparison of Python Code in Jupyter Notebooks and Scripts*. Association for Computing Machinery.

Hannay, Jo Erskine et al. 2009. "How Do Scientists Develop and Use Scientific Software?" *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, SECSE 2009*: 1–8.

Heroux, Michael et al. 2023. *Basic Research Needs in The Science of Scientific Software Development and Use: Investment in Software Is Investment in Science*. https://www.osti.gov/servlets/purl/1846009/.

Horton, William. 2020. *A Brief History of Jupyter Notebooks*.

Jaton, Florian. 2023. "Groundwork for AI: Enforcing a Benchmark for Neoantigen Prediction in Personalized Cancer Immunotherapy." *Social Studies of Science*.

Joppa, Lucas N. et al. 2013. "Troubling Trends in Scientific Software Use." *Science* 340(6134): 814–15. https://www.science.org/doi/10.1126/science.1231535.

Kanewala, Upulee, and James M. Bieman. 2014. "Testing Scientific Software: A Systematic Literature Review." *Information and Software Technology* 56(10): 1219–32. http://dx.doi.org/10.1016/j.infsof.2014.05.006.

Kelty, Christopher M. 2008. *Two Bits: The Cultural Significance of Free Software*. Duke University Press.

Knuth, D. E. 1984. "Literate Programming." *The Computer Journal* 27(2): 97–111. https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/27.2.97.

Lamy, Jérôme. 2022. "Le Renouveau de l'histoire Des Instruments Scientifiques." *Artefact* (17): 9–38. http://journals.openedition.org/artefact/13018.

Ledford, Heidi. 2016. "The Unsung Heroes of Scientific Software." *Nature* 535(7612): 342–44.

Leonelli, Sabina, and Niccolò Tempini. 2020. *Data Journeys in the Sciences*. SpringerOpen.

Li, Kai, Erjia Yan, and Yuanyuan Feng. 2017. "How Is R Cited in Research Outputs? Structure, Impacts, and Citation Standard." *Journal of Informetrics* 11(4): 989–1002. https://doi.org/10.1016/j.joi.2017.08.003.

Lin, Yuwei. 2005. "The Future of Sociology of FLOSS." *First Monday*. https://firstmonday.org/ojs/index.php/fm/article/view/1467.

Marcovich, Anne, and Terry Shinn. 2011. "From the Triple Helix to a Quadruple Helix? The Case

of Dip-Pen Nanolithography.” *Minerva* 49(2): 175–90. http://link.springer.com/10.1007/s11024-011-9169-z (September 28, 2014).

Marcovich, Anne, and Terry Shinn. 2021. “When Two Science Disciplines Meet: Evaluating Dynamics of Conjunction. The Encounter between Astrophysics and Artificial Intelligence.” *Social Science Information* 60(3): 372–77. http://journals.sagepub.com/doi/10.1177/05390184211025848.

Mariannig Le Béchec et al. 2022. *Pratiques et Usages Des Outils Numériques Dans Les Communautés Scientifiques En France. Pratiques et Usages Des Outils Numériques Dans Les Communautés Scientifiques En France*.

Méadel, Cécile, and Guillaume Sire. 2017. “Les Sciences Sociales Orientées Programmes.” *Reseaux* 206(6): 9–34.

Meertens, Lambert. 2022. “The Origins of Python.” *Inference*.

Mirowski, Philip. 2018. “The Future(s) of Open Science.” *Social Studies of Science* 48(2): 171–203.

Pérez, Fernando, Brian E. Granger, and John D. Hunter. 2011. “Python: An Ecosystem for Scientific Computing.” *Computing in Science and Engineering* 13(2): 13–21.

Perkel, Jeffrey M. 2021. “Ten Computer Codes That Transformed Science.” *Nature* 589(7842): 344–48. http://www.nature.com/articles/d41586-021-00075-2.

Perkel, Jeffrey M. 2018. “Why Jupyter Is Data Scientists’ Computational Notebook of Choice.” *Nature* 563(7729): 145–46.

Pimentel, João Felipe, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. 2021. “Understanding and Improving the Quality and Reproducibility of Jupyter Notebooks.” *Empirical Software Engineering* 26(4).

Plantin, Jean Christophe, Carl Lagoze, and Paul N. Edwards. 2018. “Re-Integrating Scholarly Infrastructure: The Ambiguous Role of Data Sharing Platforms.” *Big Data and Society* 5(1): 1–14.

Plantin, Jean Christophe, Carl Lagoze, Paul N. Edwards, and Christian Sandvig. 2018. “Infrastructure Studies Meet Platform Studies in the Age of Google and Facebook.” *New Media and Society* 20(1): 293–310.

Quaranta, Luigi, Fabio Calefato, and Filippo Lanubile. 2022. “Eliciting Best Practices for Collaboration with Computational Notebooks.” *Proceedings of the ACM on Human-Computer Interaction* 6(CSCW1): 1–41.

Rabinow, Paul. 1996. *Making PCR. A Story of Biotechnology*. Chicago Press.

Rule, Adam, Aurélien Tabard, and James D. Hollan. 2018. “Exploration and Explanation in Computational Notebooks.” *Conference on Human Factors in Computing Systems - Proceedings* 2018-April: 1–12.

Samuel, Sheeba, and Daniel Mietchen. 2022. “Computational Reproducibility of Jupyter Notebooks from Biomedical Publications.” : 1–31. http://arxiv.org/abs/2209.04308.

Sauret, Nicolas. 2022. “Intelligence Artificielle & Sciences Humaines et Sociales (SHS) : Opportunités, Défis et Perspectives.” *I2D - Information, données & documents* n° 1(1): 97–103.

Schindler, David, Felix Bensmann, Stefan Dietze, and Frank Krüger. 2022. “The Role of Software in Science: A Knowledge Graph-Based Analysis of Software Mentions in PubMed Central.” *PeerJ Computer Science* 8: 1–47.

Shinn, Terry. 2000. “Formes de Division Du Travail Scientifique et Convergence Intellectuelle. La Recherche Technico-Instrumentale.” *Revue française de sociologie* 41(3): 447–73.

Shulz, Sébastien. 2019. “Un Logiciel Libre Pour Lutter Contre l’opacitÉ Du SystÈme Sociofiscal: Sociologie d’une Mobilisation Hétérogène Aux Marges de l’État.” *Revue Francaise de Science Politique* 69(5): 845–68.

Simoulin, Vincent. 2012. *Sociologie d’un Grand Équipement Scientifique: Le Premier Synchrotron de Troisième Génération*. Lyon: ENS Edition.

Spencer, Matt. 2015. “Brittleness and Bureaucracy: Software as a Material for Science.” *Perspectives on Science* 23(4): 466–84.

Thomas, Rollin, and Shreyas Cholia. 2021. “Interactive Supercomputing with Jupyter.” *Computing in Science and Engineering* 23(2): 93–98.

Vertesi, Janet, and David Ribes. 2019. *DigitalSTS A Field Guide For Science & Technology Studies*.

Wagemann, Julia et al. 2022. “Five Guiding Principles to Make Jupyter Notebooks Fit for Earth Observation Data Education.” *Remote Sensing* 14(14).

Woolston, Chris. 2022. “Why Science Needs More Research Software Engineers.” *Nature*: 1–6.

Zurbach, Jonathan. 2022. “Le Genre « exécutable » : Un Écosystème de Contrats Au Service de La Science Ouverte.”