Accompagnement à l'exécution des notebooks Jupyter en milieu éducatif

Christophe Casseau

LABORATOIRE BORDELAIS DE RECHERCHE EN INFORMATIQUE LaBRI

université de BORDEAUX

THÈSE

Présentée pour obtenir le grade de docteur délivré par

L'université de Bordeaux

École doctorale : Mathématiques et Informatique

Spécialité : Informatique

présentée et soutenue publiquement par

Christophe Casseau

le 20 juin 2024

Accompagnement à l'Exécution des Notebooks Jupyter en Milieu Éducatif

Directeurs de thèse : **Jean-Rémy Falleri**, **Xavier Blanc**

Jury

Jean-Rémy Falleri,	Professeur	Directeur
Xavier Blanc,	Professeur	Directeur
Mireille Blay-Fornarino,	Professeure	Rapporteur
Benoit Combemale,	Professeur	Rapporteur
Guillaume Blin,	Professeur	Examinateur
Franck Silvestre,	Maître de conférences	Examinateur

Invité

Thomas Degueule,	Chargé de recherche	co-encadrant

Université de Bordeaux, LaBRI
351, cours de la Libération F-33405 Talence cedex
France

Remerciements

L'accomplissement de cette thèse a représenté bien plus qu'une simple étape académique. Cette période intense et exigeante de ma vie a été marquée par des défis constants, des découvertes enrichissantes, des moments de joie mais également par des instants de doute et de remise en question. J'ai souvent été confronté à des obstacles qui ont mis à l'épreuve ma détermination et ma persévérance. À travers les succès et les échecs, j'ai appris à accepter l'incertitude, à embrasser les imprévus et à réaliser que la science est autant une quête de soi qu'une recherche du savoir. Mais durant ces 5 années de thèse, j'ai eu la chance d'être entouré de personnes que j'ai énormement appréciées et sans qui rien de tout cela n'aurait pu être possible. Je tiens à remercier tout particulièrement Jean-Rémy Falleri, Xavier Blanc et Thomas Degueule qui m'ont accueilli au sein de l'équipe PROGRESS. Trois personnes qui m'ont appris que la science n'est pas qu'une quête de vérité, mais aussi une œuvre collective, où l'échange, le soutien, et l'inspiration mutuelle jouent un rôle aussi crucial que la rigueur et le travail.

Je tiens également à remercier les membres du jury pour avoir accepté d'évaluer mon manuscrit et pour l'intérêt qu'ils ont porté à mes travaux de recherches : Mireille Blay-Fornarino et Benoit Combemale ainsi que les autres membres du jury, Guillaume Blin et Franck Silvestre.

Merci aux collègues Corentin Latappy, Romain Robbes, Joachim Bruneau-Queyreix, Laurent Reveillere, Floréal Morandat, Carole Blanc et Lionel Clement pour leur soutien moral et/ou technique. Merci également à tous les autres collègues du LaBRI avec qui j'ai pu échanger et qui ont consacré du temps pour répondre à toutes les questions que j'ai pu leur poser.

Enfin, je tiens à remercier ma famille pour leurs encouragements inestimables tout au long de ces années, mention spéciale à Jean-Pierre qui a distillé sans le savoir cette envie un peu folle. Un grand merci à ma femme pour avoir accepté les nombreux sacrifices que nous avons dû faire pour mener à bien cette thèse, ainsi que pour son soutien sans faille et sa compréhension durant ces années exigeantes. Je remercie également mes enfants pour leur patience et leur soutien. Je tiens à leur exprimer toute ma fierté en les voyant terminer leurs études et réaliser leurs propres objectifs. Votre détermination et vos réussites sont une source d'inspiration et de joie pour moi, et je suis profondément admiratif de vos accomplissements.

Je conclus ces remerciements par quelques mots destinés à mes deux directeurs de thèse, Xavier Blanc et Jean-Rémy Falleri, ainsi qu'à mon encadrant préféré, Thomas Degueule. À vous trois, Xavier, Jean-Rémy et Thomas je dédie ces lignes avec une gratitude immense. Chacun de vous a joué un rôle essentiel et unique dans la réalisation de cette thèse. Merci d'avoir été là, d'avoir cru en moi et d'avoir fait de ces cinq années de thèse des années de partage.

Cher Xavier,

Il est difficile de trouver les mots justes pour exprimer toute ma gratitude envers toi. Ton soutien a été l'étape essentielle qui a permis à cette thèse d'exister. Le jour où je me suis présenté à ton bureau pour t'exposer ce projet un peu fou, j'étais loin de mesurer pleinement l'ampleur de la tâche à accomplir. Cependant, tu as accepté de t'engager avec moi en tant que directeur de thèse, et je t'en suis profondément reconnaissant. Les longues discussions que nous avons eues ensemble ont été des moments importants de ce parcours. Ta vision décalée et ta créativité m'ont souvent ouvert les yeux sur de nouvelles perspectives, enrichissant ainsi ma réflexion et mon approche du travail.

Cher Jean-Rémy,

Ta joie et ton dynamisme m'ont aidé à rebondir dans les moments difficiles et pour cela je te dis un grand merci. Mais au delà des ces qualités humaines, j'ai énormément apprécié ta rigueur scientifique qui a assuré la solidité et la crédibilité de mon travail. Cette exigence intellectuelle constante que tu cultives, m'a permis de développer une approche méthodique et rigoureuse dans mes recherches et a été essentielle pour transformer ce projet en une réalisation académique de qualité. En fin de compte, même si j'ai souvent été perplexe face au désormais célèbre *c'est pas pire* concluant certaines de tes remarques, j'étais bien conscient que cela exprimait ta volonté de me pousser à toujours améliorer la qualité de mon travail.

Cher Thomas,

Ta bienveillance, ta capacité à percevoir les besoins des autres et à offrir ton aide ont fait de toi le troisième pilier de cette thèse. Ton écoute attentive et ta disponibilité ont été des sources constantes de soutien et d'encouragement. Merci pour ton encadrement de grande qualité et je te suis très reconnaissant d'avoir accepté de partager tes connaissances, que je trouve immenses, tout au long de ce doctorat.

Un grand merci à tous, avec toute ma reconnaissance,
Christophe Casseau

Résumé

Les notebooks sont devenus des outils incontournables dans le domaine de l'analyse de données. Initiés dans les années 1980 avec des logiciels tels que Mathematica et inspirés par le concept de la programmation littéraire de Knuth, leur popularité se concrétise grâce au projet Jupyter en 2014. Ils ont transformé la manière dont les scientifiques communiquent leurs idées en combinant du code exécutable parmi une grande variété de langages de programmation, des visualisations et des explications textuelles dans un même document interactif. Ils ont également largement investi le monde éducatif par exemple avec le programme CANDYCE lancé par l'état français en 2021. Ce programme encourage l'utilisation de l'environnement Jupyter dans l'enseignement des sciences du numérique et ce à tous les niveaux, du primaire à l'enseignement supérieur en proposant des notebooks éducatifs qui sont au cœur de cette thèse.

Dans ce contexte éducatif, malgré leurs avantages indéniables, les notebooks présentent également des défis importants, notamment en matière de reproductibilité et de modèle d'exécution. En effet, les notebooks éducatifs embarquent une activité pédagogique contenant des instructions textuelles guidant les étudiants à travers les différentes tâches à réaliser. Ensuite, l'enseignant cherche à reproduire les résultats des étudiants en suivant un ordre le plus souvent linéaire. La reproductibilité des résultats constitue une promesse des notebooks, mais plusieurs études ont révélé des difficultés à atteindre cet objectif, nécessitant le développement d'approches pour accompagner les utilisateurs dans la création de notebooks reproductibles. De plus, le modèle d'exécution flexible des notebooks donne la possibilité aux étudiants d'exécuter les cellules de code dans un ordre différent de celui prévu par l'enseignant pouvant occasionner des erreurs et/ou des résultats trompeurs.

Dans cette thèse, nous nous penchons sur ces deux défis que sont la reproductibilité des résultats et l'exécution des notebooks éducatifs. Notre objectif est de proposer deux approches indépendantes du langage de programmation afin d'accompagner les étudiants i) vers la reproductibilité des résultats dans un modèle d'exécution linéaire du haut vers le bas et ii) à l'exécution d'un notebook contenant un scénario c'est à dire des instructions liées à son exécution. Pour répondre à ces deux défis nous avons développé des outils directement intégrés à l'environnement JupyterLab : NORM et MOON. Ces outils ont permis de mettre en évidence à travers des expérimentations menées avec des étudiants de C.P.G.E et de première année universitaire une nette amélioration concernant les deux défis sans entraver l'apprentissage des étudiants.

Mots clés

Jupyter, notebook, éducatif, reproductibilité des résultats, reproductibilité du scénario

Abstract

Notebooks have become essential tools in the field of data science. Initiated in the 1980s with software such as Mathematica and inspired by Knuth's concept of literate programming, their popularity was solidified with the Jupyter project in 2014. They have transformed how scientists communicate their ideas by combining executable code from a wide variety of programming languages, visualizations, and textual explanations in a single interactive document. They have also gain in popularity in the educational world, for example, with the CANDYCE program launched by the French government in 2021. This program encourages the use of the Jupyter environment in teaching digital sciences at all levels, from primary to higher education, by offering educational notebooks that are at the heart of this thesis.

In this educational context, despite their undeniable advantages, notebooks also present significant challenges, particularly in terms of reproducibility and execution model. Indeed, educational notebooks embed a pedagogical activity containing textual instructions guiding students through the different tasks to be completed. Then, the teacher attempts to reproduce the students' results by following a predominantly linear order. The reproducibility of results is a promise of notebooks, but several studies have revealed difficulties in achieving this goal, necessitating the development of approaches to support users in creating reproducible notebooks. Additionally, the flexible execution model of notebooks allows students to execute code cells in a different order than intended by the instructor, potentially leading to errors and/or misleading results.

In this thesis, we address these two challenges : the reproducibility of results and the execution of educational notebooks. Our goal is to propose two language-agnostic approaches to assist students i) towards result reproducibility in a top-down linear execution model and ii) in the execution of a notebook containing a scenario, i.e., instructions related to its execution. To tackle these challenges, we have developed tools directly integrated into the JupyterLab environment : NORM and MOON. Through experiments conducted with students from C.P.G.E. and first-year university, these tools have demonstrated a significant improvement in both challenges without hindering student learning.

Keywords

Jupyter, notebook, educational, reproducibility of results, reproducibility of scenario

Unité de recherche

LaBRI - Équipe Progress, UMR CNRS 5800
351, cours de la Libération
F-33405 Talence Cedex
France

Table des matières

Liste des figures

Chapitre 1

Introduction

« *Je ne savais pas encore que l'incompréhension va toujours plus loin que tout le savoir, plus loin que le génie, et que c'est toujours elle qui a le dernier mot.* »

Pseudo (1976)
Romain Gary

Sommaire

1.1 Contexte

Les notebooks ont émergé comme des outils puissants dans le monde de l'analyse des données et de l'enseignement. L'idée de créer des documents interactifs combinant du code exécutable, des visualisations et des explications textuelles remonte aux premières versions de logiciels comme Mathematica dans les années 1980 inspirés par le *literate programming* de Knuth [8]. Ces premières tentatives ont jeté les bases de ce qui allait devenir un changement de paradigme pour les supports de cours des enseignants. Plusieurs universités ont commencé à produire des notebooks pour illustrer leurs propres cours de mathématiques [6] ou de physique (mécanique, traitement du signal, physique quantique, relativité générale, etc). Très rapidement Mathematica accroît son succès en enrichissant sa base de données avec des cartes du monde en couleur pour les géographes et des éléments de la classification périodique pour les chimistes.

Cependant, le véritable tournant de ce que l'on appelle aujourd'hui les notebooks interactifs a eu lieu avec le lancement du projet IPython en 2001, introduisant une interface interactive en ligne de commande pour Python [13]. En 2011, c'est l'IPython Notebook qui voit le jour, offrant une plateforme web où le code, les résultats, et la narration pouvaient coexister harmonieusement [14]. Puis c'est en 2014 que le projet Jupyter [16] étend la compatibilité des notebooks à plusieurs langages de programmation, dont Julia, Python et R, d'où le nom "Jupyter".

Depuis lors, les notebooks Jupyter se sont répandus dans le monde de la programmation, de l'analyse des données et celui de l'éducation. Ils facilitent la création de rapports reproductibles ainsi que le partage de documents interactifs. Des plateformes spécialisées dans les notebooks, telles que JupyterHub, Binder et Colab, offrent aux établissements d'enseignement une solution complète pour héberger, gérer et distribuer des notebooks à grande échelle. Ces plateformes permettent aux étudiants d'accéder facilement aux notebooks depuis n'importe quel appareil connecté à Internet, sans avoir besoin d'installer un logiciel supplémentaire. En particulier, Noteable [1], développé par EDINA à l'Université d'Édimbourg, est une plateforme leader dans ce domaine. Elle permet aux enseignants de suivre facilement les progrès des étudiants, de fournir un retour d'information en temps réel et de favoriser la collaboration entre pairs.

Le nombre de notebooks Jupyter ne fait que croître depuis que le projet Jupyter a reçu le prix ACM software System Award en 2017 [2]. On estime aujourd'hui à plus de 10 millions le nombre de notebooks sur les dépôts publics GitHub [3] (fig 1.1). Les notebooks offrent des fonctionnalités de plus en plus avancées, des widgets interactifs aux extensions spécialisées, rendant la programmation et l'analyse de données plus accessibles que jamais. Bien que l'histoire des notebooks ait commencé modestement, elle a évolué pour devenir un élément essentiel du paysage technologique moderne, transformant la façon dont nous écrivons du code et dont nous envisageons un enseignement interactif à tous les niveaux du système éducatif.

Dans cette thèse, nous allons plus particulièrement nous intéresser aux notebooks du projet Jupyter dans un contexte éducatif. L'utilisation des notebooks dans ce contexte sera largement détaillée dans le chapitre 2.

1. https ://noteable.edina.ac.uk/
2. https ://blog.jupyter.org/jupyter-receives-the-acm-software-system-award-d433b0dfe3a2
3. https ://github.com/parente/nbestimate/blob/master/estimate.ipynb

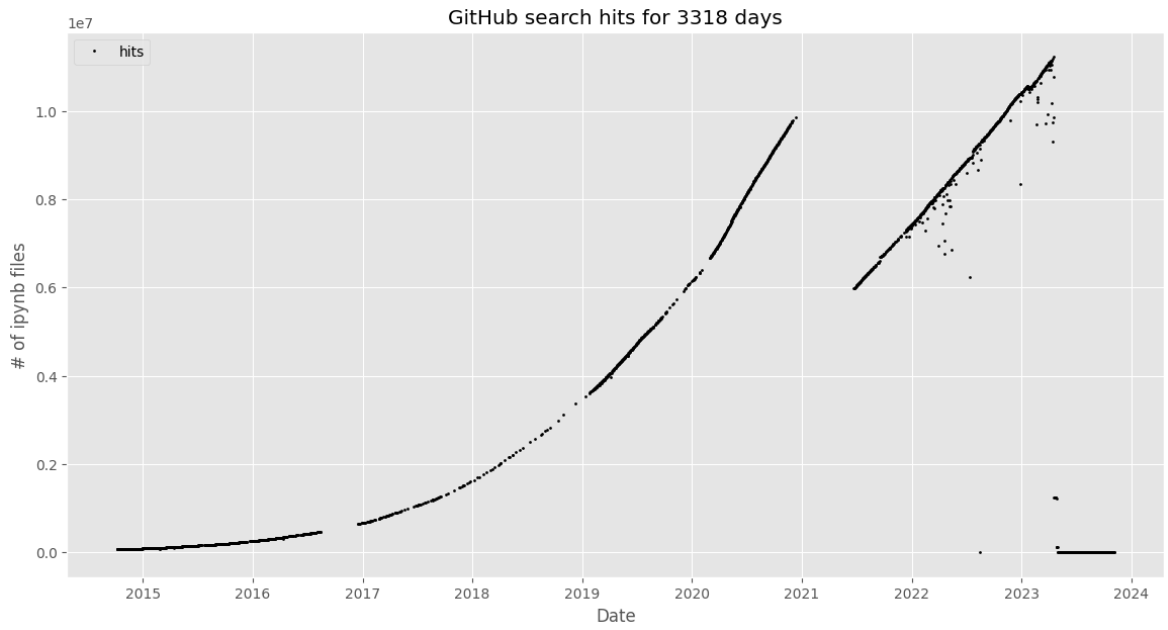


FIGURE 1.1 – Évolution du nombre de notebooks Jupyter sur les dépôts publics de GitHub. La baisse brutale constatée en 2021 est le résultat du retrait des *forks* dans le comptage.

1.2 Problématiques

Le notebook est un outil polyvalent censé faciliter la documentation et la visualisation des données liées au code. Les enseignants peuvent les utiliser pour créer des supports pédagogiques interactifs, permettant aux étudiants de s'engager activement dans l'apprentissage de la programmation. Les nombreux avantages du notebook ont suscité l'intérêt de plusieurs autres disciplines ayant des besoins de modélisation et de simulation, comme la biologie [22], la physique [19, 24], la chimie [28, 23, 10], les mathématiques [9] ou encore les sciences humaines [25]. Les notebooks peuvent être conçus et rédigés pour pousser les étudiants d'une position passive vers une position active, dans laquelle ils sont encouragés à expérimenter, explorer, évaluer et jouer avec les solutions possibles à un problème [20]. Les étudiants se retrouvent ainsi dans une approche centrée sur l'apprenant.

Outre ces indéniables avantages, les notebooks possèdent également de nombreux points faibles qui restent à améliorer et plusieurs defis subsistent pour une bonne utilisation des notebooks dans un contexte éducatif [7]. Des études récentes ont révélé plusieurs types de problèmes. Dans le contexte éducatif, nous allons nous intéresser en priorité à la reproductibilité des résultats [15, 27, 1] et au modèle d'exécution des notebooks [3, 17, 26].

Reproductibilité L'une des promesses des notebooks Jupyter est qu'ils devraient rendre les contenus des notebooks reproductibles, c'est-à-dire que les lecteurs devraient être en mesure de reconstruire et d'évaluer le cheminement de la pensée de l'auteur du notebook à la lecture de celui-ci. Or de nombreuses études [15, 27, 18] ont montré que la reproductibilité des notebooks peut-être un vrai défi et qu'il est nécessaire de développer des approches qui soutiennent les utilisateurs (auteurs comme lecteurs) pour obtenir des notebooks reproductibles.

Modèle d'exécution Un autre défi dans le contexte éducatif est le modèle d'exécution du notebook. La grande souplesse de ce modèle permettant d'exécuter les cellules de

code dans n'importe quel ordre peut rapidement devenir un problème lorsque les étudiants ne suivent pas l'ordre d'exécution prévu par l'enseignant. Pour contrer cet effet indésirable, les enseignants écrivent généralement des instructions détaillées sur la façon dont les élèves sont censés utiliser les notebooks. Malheureusement ces instructions ne sont pas toujours bien suivies par les étudiants. La compréhension d'un texte dépend de nombreux facteurs, comme les caractéristiques du lecteur, le contenu et la conception du texte, et les instructions de lecture [11]. Certaines études démontrent même clairement que les scores de compréhension en lecture sont plus faibles pour les textes numériques que pour les textes imprimés [5, 12, 21, 4]. Ainsi, même avec la volonté de bien faire, les étudiants peuvent se retrouver avec un état du notebook non prévu par l'enseignant, ce qui entraîne des erreurs ou des résultats trompeurs qui entravent leur apprentissage [3].

1.3 Objectifs et Contributions

Les problématiques récurrentes dans le contexte éducatif, telles que la gestion de l'ordre d'exécution des cellules de code et la reproductibilité, engendrent des incertitudes aussi bien pour les enseignants que pour les étudiants quant à l'état réel d'un notebook. Par exemple, ces incertitudes peuvent poser des défis notables lors de l'évaluation, amenant les enseignants à interpréter des résultats parfois inattendus au moment de la correction. Ou bien, du côté des étudiants, cela peut susciter des frustrations, notamment liées à des erreurs imprévues ou à des incohérences par rapport au scénario de l'activité.

Les travaux entrepris dans le cadre de cette thèse ont pour objectif de fournir une assistance ciblée à l'utilisation des notebooks dans le contexte spécifique de l'éducation. Les approches proposées et les outils les supportant doivent être indépendants du langage de programmation afin d'être utilisables dans un éventail de disciplines le plus large possible. En effet chaque discipline a ses propres exigences et préférences en matière de langage de programmation en fonction des tâches et des applications spécifiques qu'elle aborde, par exemple les biologistes ont plutôt tendance à utiliser R alors que les sciences de l'ingénieur utilisent Java ou Python. Les outils que nous avons développés ont été testés dans le contexte d'un cours d'informatique avec des étudiants en classe préparatoire aux grandes écoles et des étudiants universitaires de niveau débutant. Ces évaluations ont été réalisées en utilisant le langage Python. Nos travaux proposent deux axes d'amélioration avec deux outils intégrés directement dans l'interface de JupyterLab.

1. **Comment accompagner les étudiants vers des résultats reproductibles dans un notebook éducatif dont le scénario d'exécution est linéaire?** Nous avons développé un outil open source NORM [1], permettant d'intégrer un mécanisme de surveillance en temps réel dans JupyterLab afin d'offrir à l'aide d'un système de couleurs une information immédiate sur la reproductibilité du notebook. Il avertit l'étudiant dès que la sortie d'une cellule n'est plus reproductible, offrant ainsi la possibilité de résoudre le problème dès qu'il est détecté. Les résultats montrent que NORM améliore significativement la reproductibilité des notebooks sans sacrifier la productivité des étudiants.

2. **Comment accompagner les étudiants pour optimiser le suivi du scénario d'une activité contenue dans un notebook éducatif?**

   Nous proposons une approche utilisant un outil open source appelé MOON [2]. Cet outil permet aux enseignants de créer des scripts qui intègrent des éléments visuels directement dans l'interface du notebook. L'avantage est de faciliter la lecture de

l'activité et d'assurer une navigation plus cohérente avec les instructions contenues dans le notebook. Nous avons réalisé une expérience contrôlée impliquant 21 étudiants de première année universitaire qui montre que MOON aide les étudiants à mieux respecter le scénario prévu sans entraver leur capacité à progresser.

En combinant ces deux axes d'amélioration, notre approche cherche à enrichir l'expérience éducative avec des outils visuels intuitifs et des indicateurs en temps réel, contribuant ainsi à une utilisation plus efficace et efficiente des notebooks dans le domaine éducatif.

1.4 Publications

- C. Casseau, J. Falleri, X. Blanc, and T. Degueule, "Immediate feedback for students to solve notebook reproducibility problems in the classroom," in *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2021, St Louis, MO, USA, October 10-13, 2021*, K. J. Harms, J. Cunha, S. Oney, and C. Kelleher, Eds. IEEE, 2021, pp. 1–5. [Online]. Available : https ://doi.org/10.1109/VL/HCC51201.2021.9576363
- C. Casseau, J.-R. Falleri, T. Degueule, and X. Blanc, "Moon : Assisting students in completing educational notebook scenarios," in *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2023, pp. 157–167.

1.5 Structure de la thèse

Afin de guider le lecteur à travers les différents aspects de cette recherche, il est essentiel de clarifier la structure de la thèse.

Suite à cette introduction, la thèse est structurée en quatre chapitres principaux, chacun traitant d'un aspect spécifique du sujet de recherche. Le premier chapitre présente la notion de notebook, puis explore l'utilisation des notebooks dans un contexte éducatif, en mettant en lumière les défis qui y sont associés. Le deuxième chapitre se concentre sur l'état de l'art concernant ces défis. Les deux derniers chapitres détaillent les approches adoptées pour surmonter ces défis, en décrivant la méthodologie utilisée pour la collecte et l'analyse des données, ainsi que les résultats obtenus.

Le chapitre final synthétise les principales conclusions de la thèse, discutant des implications théoriques et pratiques, et suggérant des pistes pour des recherches futures. Tout au long de la thèse, des références appropriées sont citées pour soutenir les arguments présentés.

Chapitre 2

Ce chapitre explore d'abord la popularité croissante des notebooks, en décrivant brièvement les applications Jupyter et leur architecture sous-jacente, notamment en examinant l'interface utilisateur d'un notebook et la structure JSON du fichier notebook.

Ensuite, il examine le rôle des notebooks en tant qu'outil éducatif, mettant en lumière leurs capacités interactives et exploratoires qui soutiennent l'apprentissage. Il clarifie également les notions de notebook éducatif et de cycle de vie associé.

Enfin, il soulève les défis rencontrés dans l'utilisation des notebooks à des fins éducatives et présente brièvement les deux contributions de cette thèse : NORM et MOON.

Chapitre 3

Ce chapitre consacré à l'état de l'art s'articule en deux parties distinctes. En réponse aux défis relevés dans le chapitre 2, nous commencerons par examiner la notion de reproductibilité appliquée aux notebooks. Nous ferons le point sur les définitions fondamentales existantes de la reproductibilité qui sous-tendent notre discussion, mettant en avant l'importance de la reproductibilité dans le domaine scientifique. Nous parlerons ensuite très brievement de son application dans le génie logiciel, avant d'analyser un problème spécifique aux notebooks : l'impact de l'ordre d'exécution des cellules de code sur la cohérence et la reproductibilité des notebooks. Nous conclurons en examinant les outils développés pour aider les utilisateurs de notebooks.

Dans la deuxième partie, nous aborderons l'utilisation de la polyvalence des notebooks dans un contexte éducatif, à la fois dans diverses disciplines et plus spécifiquement avec des étudiants en informatique.

Chapitre 4

Ce chapitre aborde le problème de la reproductibilité des résultats dans l'évaluation des notebooks des étudiants. Les enseignants ont souvent du mal à reproduire les résultats présentés par les étudiants, car ces derniers exploitent la structure non linéaire des notebooks. Pour résoudre ce problème, nous présentons le plugin Jupyter NORM (Notebook Reproducibility Monitor), qui offre aux étudiants un retour visuel sur la reproductibilité de leurs notebooks. Ce plugin alerte les étudiants avec un système de couleurs lorsque la sortie d'une cellule n'est pas reproductible lorsque le notebook est exécuté linéairement du haut vers le bas. Nous montrons ensuite que cette approche améliore significativement la reproductibilité des notebooks sans nuire à la productivité des étudiants, notamment lors d'une séance d'informatique pour tous en classe préparatoire aux grandes écoles (PCSI).

Chapitre 5

Ce chapitre explore la reproductibilité du scénario d'une activité embarquée dans un notebook. Nous mettons en lumière les problèmes potentiels, tels que les erreurs d'exécution ou les résultats incorrects, lorsque les étudiants ne suivent pas l'ordre d'exécution prévu par l'enseignant, en nous appuyant sur un cas d'étude.

Ensuite, nous introduisons MOON, une approche innovante visant à renforcer la reproductibilité de l'activité dans les notebooks. Cette approche repose sur la création d'un langage permettant aux enseignants de formaliser l'utilisation attendue des notebooks sous forme de script, et de fournir aux étudiants des indications visuelles en temps réel pour les guider pendant leur interaction avec les notebooks.

Nous évaluons ensuite notre approche à travers une expérience contrôlée randomisée impliquant 21 étudiants, démontrant que MOON aide les étudiants à respecter plus efficacement le scénario prévu sans entraver leur progression.

1.6 Références

[1] Christophe CASSEAU et al. “Immediate Feedback for Students to Solve Notebook Reproducibility Problems in the Classroom”. In : *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2021, St Louis, MO, USA, October 10-13, 2021*. Sous la dir. de Kyle J. HARMS et al. IEEE, 2021, p. 1-5. DOI : 10.1109/VL/HCC51201.2021.9576363. URL : https://doi.org/10.1109/VL/HCC51201.2021.9576363.

[2] Christophe CASSEAU et al. “MOON: Assisting Students in Completing Educational Notebook Scenarios”. In : *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2023, p. 157-167.

[3] Souti CHATTOPADHYAY et al. “What’s Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities”. In : *CHI ’20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*. Sous la dir. de Regina BERNHAUPT et al. ACM, 2020, p. 1-12. DOI : 10.1145/3313831.3376729. URL : https://doi.org/10.1145/3313831.3376729.

[4] Pablo DELGADO et al. “Don’t throw away your printed books: A meta-analysis on the effects of reading media on reading comprehension”. In : *Educational Research Review* 25 (2018), p. 23-38. ISSN : 1747-938X. DOI : 10.1016/j.edurev.2018.09.003. URL : https://doi.org/10.1016/j.edurev.2018.09.003.

[5] Andrew DILLON. “Reading from paper versus screens: a critical review of the empirical literature”. In : *Ergonomics* 35.10 (1992), p. 1297-1326. URL : https://doi.org/10.1080/00140139208967394.

[6] ROY JEAN-PAUL. “LE CALCUL FORMEL Une introduction aux logiciels MAPLE et MATHEMATICA (2ème partie)”. In : ().

[7] Jeremiah W. JOHNSON. “Benefits and Pitfalls of Jupyter Notebooks in the Classroom”. In : *SIGITE 20: The 21st Annual Conference on Information Technology Education, Virtual Event, USA, October 7-9, 2020*. Sous la dir. de Deepak KHAZANCHI et al. ACM, 2020, p. 32-37. DOI : 10.1145/3368308.3415397. URL : https://doi.org/10.1145/3368308.3415397.

[8] Donald E. KNUTH. “Literate Programming”. In : *Comput. J.* 27.2 (mai 1984), p. 97-111. ISSN : 0010-4620. DOI : 10.1093/comjnl/27.2.97. URL : https://doi.org/10.1093/comjnl/27.2.97.

[9] Jacob KOEHLER et Soomi KIM. “Interactive Classrooms with Jupyter and Python”. In : *The Mathematics Teacher* 111 (jan. 2018), p. 304. DOI : 10.5951/mathteacher.111.4.0304.

[10] Deborah LAFUENTE et al. “A Gentle Introduction to Machine Learning for Chemists: An Undergraduate Workshop Using Python Notebooks for Visualization, Data Processing, Analysis, and Modeling”. In : *Journal of Chemical Education* 98.9 (2021), p. 2892-2898. DOI : 10.1021/acs.jchemed.1c00142. URL : https://doi.org/10.1021/acs.jchemed.1c00142.

[11] Danielle S. MCNAMARA et Joe MAGLIANO. “Chapter 9 Toward a Comprehensive Model of Comprehension”. In : *The Psychology of Learning and Motivation*. T. 51. Psychology of Learning and Motivation. Academic Press, 2009, p. 297-384. URL : https://www.sciencedirect.com/science/article/pii/S0079742109510092.

[12] Jan M. NOYES et Kate J. GARLAND. “Computer- vs. paper-based tasks: Are they equivalent?” In : *Ergonomics* 51.9 (2008). PMID: 18802819, p. 1352-1375. DOI : 10.1080/00140130802170387. URL : https://doi.org/10.1080/00140130802170387.

[13] Fernando PEREZ et Brian E. GRANGER. “IPython: A System for Interactive Scientific Computing”. In : *Computing in Science & Engineering* 9.3 (2007), p. 21-29. DOI : 10.1109/MCSE.2007.53.

[14] Fernando PÉREZ et Brian E. GRANGER. “AN OPEN SOURCE FRAMEWORK FOR INTERACTIVE, COLLABORATIVE AND REPRODUCIBLE SCIENTIFIC COMPUTING AND EDUCATION”. In : 2012. URL : https://api.semanticscholar.org/CorpusID:15409553.

[15] João Felipe PIMENTEL et al. “A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks”. In : *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, p. 507-517. DOI : 10.1109/MSR.2019.00077.

[16] Min RAGAN-KELLEY et al. “The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication.” In : *AGU Fall Meeting Abstracts*. T. 2014. 2014, H44D-07.

[17] Adam RULE et al. “Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding”. In : *Proc. ACM Hum.-Comput. Interact.* 2.CSCW (nov. 2018). DOI : 10.1145/3274419. URL : https://doi.org/10.1145/3274419.

[18] Adam RULE et al. *Ten Simple Rules for Reproducible Research in Jupyter Notebooks*. 2018. arXiv : 1810.08055 [cs.OH].

[19] Farag M. SALLABI et Sanja LAZAROVA-MOLNAR. “Teaching Modeling, Simulation, and Performance Evaluation Course Online with Jupyter Notebook: Course Development and Lessons Learned”. In : *IEEE Frontiers in Education Conference, FIE 2022, Uppsala, Sweden, October 8-11, 2022*. IEEE, 2022, p. 1-8. DOI : 10.1109/FIE56618.2022.9962690. URL : https://doi.org/10.1109/FIE56618.2022.9962690.

[20] Alessio De SANTO et al. “Promoting Computational Thinking Skills in Non-Computer-Science Students: Gamifying Computational Notebooks to Increase Student Engagement”. In : *IEEE Trans. Learn. Technol.* 15.3 (2022), p. 392-405. DOI : 10.1109/TLT.2022.3180588. URL : https://doi.org/10.1109/TLT.2022.3180588.

[21] Lauren M. SINGER et Patricia A. ALEXANDER. “Reading on Paper and Digitally: What the Past Decades of Empirical Research Reveal”. In : *Review of Educational Research* 87.6 (2017), p. 1007-1041. DOI : 10.3102/0034654317722961. URL : https://doi.org/10.3102/0034654317722961.

[22] Adam A. SMITH. “Teaching Computer Science to Biologists and Chemists, Using Jupyter Notebooks: Tutorial Presentation”. In : *J. Comput. Sci. Coll.* 32.1 (oct. 2016), p. 126-128. ISSN : 1937-4771.

[23] Marie van STAVEREN. “Integrating Python into a Physical Chemistry Lab”. In : *Journal of Chemical Education* 99.7 (2022), p. 2604-2609. DOI : 10.1021/acs.jchemed.2c00193. URL : https://doi.org/10.1021/acs.jchemed.2c00193.

[24] C SUTRINI, D PASSARO et F PALLOTTA. “The potential of using Jupyter Notebook in physics education: Experimentation for high school students”. In : *Il nuovo cimento C* 45.6 (2022), p. 1-4.

[25] Chiin-Rui TAN, Sharon Broude GEVA et Dirk COLBRY. “The Nascent Case for Adopting Jupyter Notebooks as a Pedagogical Tool for Interdisciplinary Humanities, Social Science, and Arts Education”. In : *Comput. Sci. Eng.* 23.2 (2021), p. 107-113. DOI : 10.1109/MCSE.2021.3062199. URL : https://doi.org/10.1109/MCSE.2021.3062199.

[26] Jiawei WANG, Li LI et Andreas ZELLER. “Better code, better sharing: on the need of analyzing jupyter notebooks”. In : *ICSE-NIER 2020: 42nd International Conference on Software Engineering, New Ideas and Emerging Results, Seoul, South Korea, 27 June - 19 July, 2020*. Sous la dir. de Gregg ROTHERMEL et Doo-Hwan BAE. ACM, 2020, p. 53-56. DOI : 10.1145/3377816.3381724. URL : https://doi.org/10.1145/3377816.3381724.

[27] Jiawei WANG et al. “Assessing and Restoring Reproducibility of Jupyter Notebooks”. In : *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2020, p. 138-149.

[28] Charles J. WEISS. “A Creative Commons Textbook for Teaching Scientific Computing to Chemistry Students with Python and Jupyter Notebooks”. In : *Journal of Chemical Education* 98.2 (2021), p. 489-494. DOI : 10.1021/acs.jchemed.0c01071. URL : https://doi.org/10.1021/acs.jchemed.0c01071.

Chapitre 2

Les notebooks : une approche innovante pour l'enseignement

« L'éducation est une arme puissante pour faire évoluer les mentalités et transcender les différences »

Discours 06/09/2007
Nelson Mandela

Sommaire

Dans ce chapitre nous aborderons dans un premier temps la notion de notebook pour essayer de comprendre le succès de ce type de matériel. Nous présenterons les notebooks numériques et en particulier le notebook des applications Jupyter. Nous décrirons succintement l'architecture de ces applications et nous analyserons le contenu d'un notebook en tant que vue d'un fichier JSON dans une interface web.

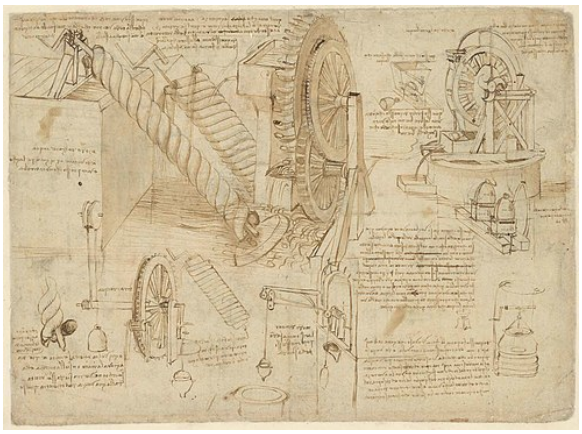
Dans un deuxième temps nous discuterons de l'intérêt d'un notebook en tant que matériel éducatif. Nous explorerons les éléments clés du notebook qui en font un outil capable de soutenir des activités d'apprentissage interactives. Ensuite, nous fournirons quelques définitions visant à mieux comprendre les concepts de notebook éducatif et de cycle de vie d'un notebook éducatif. Enfin, nous aborderons en particulier les défis aux notebooks éducatifs et offrirons un aperçu rapide des contributions NORM et MOON ayant fait l'objet de deux publications à VL/HCC [1] (Visual Languages and Human-Centric Computing) visant à proposer une assistance aux défits identifiés lors de l'utilisation des notebooks dans ce contexte.

2.1 La notion de notebook

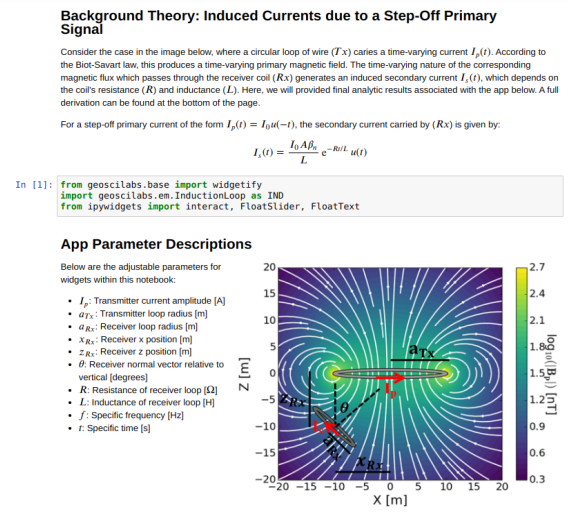
Les notebooks, dans leur conception la plus générale, sont des supports d'information utilisés pour organiser des idées, des données, ou des travaux de manière structurée. Que ce soit sous la forme d'un carnet papier traditionnel ou d'un format numérique interactif, ils servent à capturer et à partager des connaissances de manière efficace. Cette notion de notebook ou carnet de laboratoire est largement utilisée dans le domaine des sciences expérimentales comme l'atteste les célèbres carnets de laboratoire de Marie Curie, de Galilée ou encore de Léonard de Vinci (Figure 2.1a). Cependant, dans le domaine de la programmation informatique, la notion de notebook prend une toute autre dimension, émergeant comme un outil essentiel qui transcende les simples prises de notes comme l'illustre la Figure 2.1b d'un notebook de physique sur les courants induits [2].

Dans ce contexte, un notebook informatique se présente comme un environnement de développement interactif, offrant la possibilité unique de fusionner du code exécutable, des explications textuelles et des résultats visuels au sein d'un même document. Knuth avait commencé ce travail dans les années 80 avec ce qu'il avait appelé le *Literate Programming* [12]. En 2024, les notebooks numériques (Jupyter, Zeppelin, Microsoft Azure, Google Colab, CoCalc, ...) sont des outils puissants qui permettent aux utilisateurs de documenter et de partager leur travail de manière dynamique dans un document également capable d'intégrer divers médias tels que des images ou de la vidéo. Cette approche intégrée transforme le notebook en un véritable notebook de laboratoire numérique, capturant l'aspect technique avec le *comment* du code, et incluant également le *quoi* avec des explications en langage naturel.

1. https ://conf.researchr.org/home/vlhcc-2023
2. https ://notebooks.gesis.org/binder/jupyter/user/geoscixyz-em-apps-vvai1jpw/notebooks/index.ipynb



(a) Une page du Codex Atlanticus sur des systèmes d'irrigation, rédigé et dessiné par Léonard de Vinci entre 1480 et 1482 (Milan, bibliothèque Ambrosienne, no f26v).



(b) Notebook Jupyter sur la représentation de courants induits (2020)

FIGURE 2.1 – Ces deux illustrations montrent l'évolution des pratiques de documentation scientifique au fil des siècles, passant d'un format papier traditionnel (Figure 2.1a) à un format numérique interactif, 600 ans plus tard (Figure 2.1b).

2.2 Le projet Jupyter

2.2.1 Un bref historique

L'histoire des Jupyter Notebooks remonte à 2014, année de la création du projet Jupyter, mais elle trouve ses racines dans le projet IPython, initié en 2001 par Fernando Pérez [3] rejoint par Brian E. Granger en 2004 [18, 19]. IPython, qui signifie "Interactive Python", visait à créer un environnement interactif amélioré pour le langage de programmation Python. Cependant, avec le temps, la vision s'est élargie pour inclure d'autres langages de programmation, menant à la création de Jupyter en combinant les trois langages principaux : Julia, Python et R.

Jupyter est une plateforme open source qui a reçu le ACM Software System Award [4] en 2017 et qui permet aux utilisateurs de créer et de partager des documents interactifs mêlant le code, son exécution afin de produire des résultats numériques et/ou des visualisations graphiques et du texte explicatif. En 2018, dans un article de Nature [20], Jeffrey M Perkel cite Lorena Barba (ingénieur mécanique et aéronautique à l'Université George Washington à Washington DC) qui exprime le point de vue suivant : *For data scientists, Jupyter has emerged as a de facto standard.*

Vitrine des notebooks en 2018, le projet LIGO (Laser Interferometer Gravitational-Wave Observatory) de détection des ondes gravitationnelles d'origine cosmique présente certains de ses résultats sur des notebooks [5].

Au fil du temps, cette approche a bouleversé la manière dont les développeurs abordent la programmation, notamment dans la recherche scientifique, le domaine de la science des données et surtout celle de l'enseignement avec de nombreux projet intégrant la plateforme Jupyter dont le programme Candyce [6] (CArnets Numériques DYnamiques, in-

3. https ://fr.wikipedia.org/wiki/Fernando_P%C3%A9rez_(d%C3%A9veloppeur)
4. https ://fr.wikipedia.org/wiki/Prix_ACM_Software_System
5. https ://github.com/minrk/ligo-binder/blob/master/index.ipynb
6. https ://candyce.org/ProgrammeCandyce.html

teractifs et Collaboratifs pour l'Enseignement) ou le projet CAPYTALE [7] développé pour l'enseignement secondaire par l'académie de Paris. Cette approche interactive des notebooks Jupyter favorise l'apprentissage, la collaboration et améliore considérablement la reproductibilité des travaux.

Aujourd'hui les notebooks Jupyter sont largement utilisés pour la visualisation des données (Figure 2.2) [8]. Ils sont destinés à présenter des visualisations tels que des graphiques ou des tableaux pour faciliter la communication et contribue à la prise de décisions informées dans des contextes où la compréhension approfondie des données est essentielle. Ces notebooks mettent généralement l'accent sur l'analyse de données, ainsi que sur la présentation de résultats visuels clairs et informatifs pour les lecteurs.

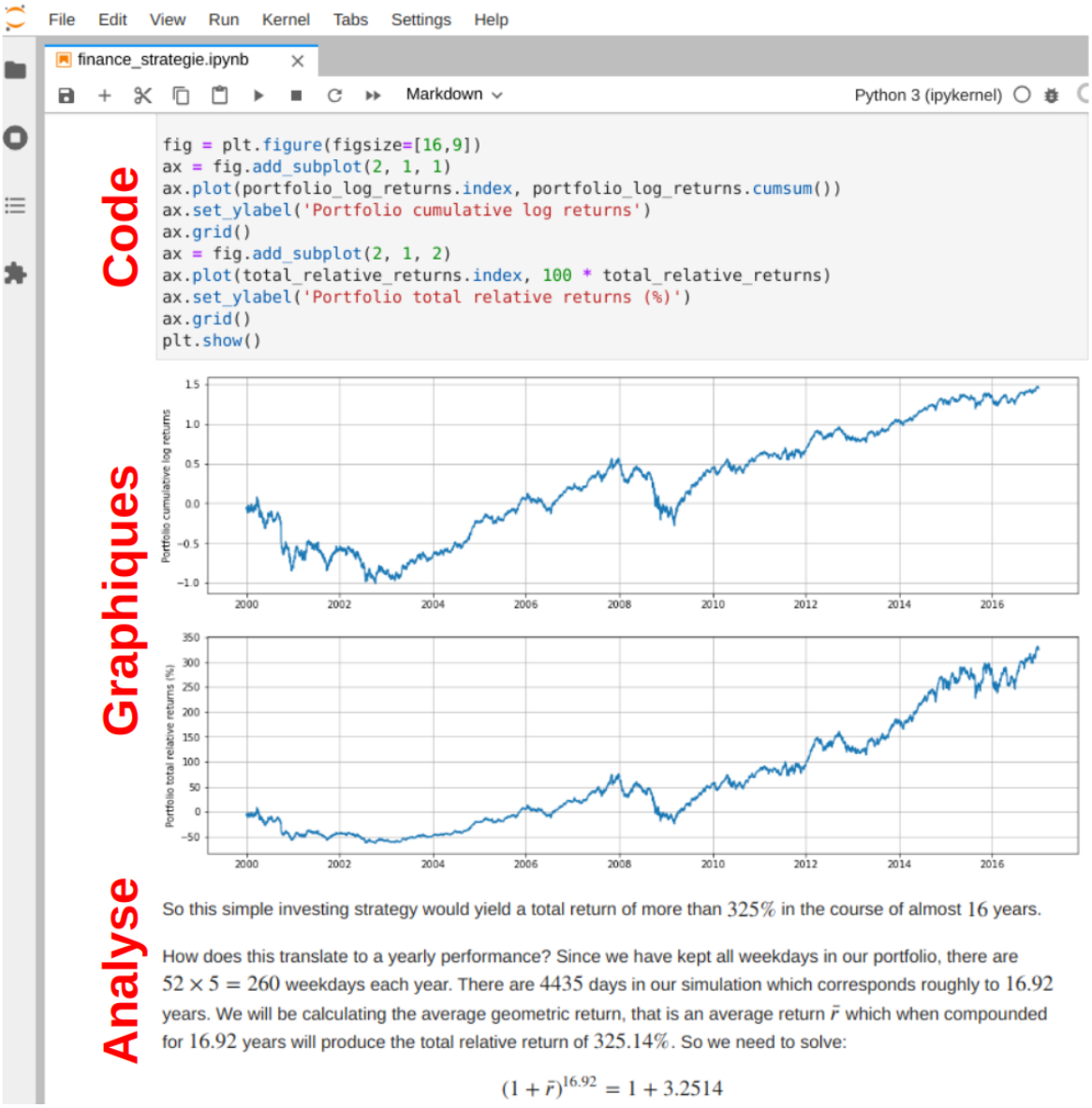


FIGURE 2.2 – Extrait d'un notebook consacré aux stratégies financières présentant des visualisations (sous forme graphique) accompagnées d'une analyse des résultats obtenus.

7. www.ac-paris.fr/capytale-un-service-web-pour-creer-et-partager-des-activites-pedagogiques-de-codage-121816
8. https ://github.com/LearnDataSci/articles

2.2.2 Applications principales du projet Jupyter

Le projet Jupyter offre une gamme d'applications conçues pour la création, l'édition, l'exécution et le partage de documents interactifs, appelés notebooks. Les principales applications sont : Jupyter QtConsole, Jupyter Console, JupyterLite, Jupyter Notebook, JupyterLab, JupyterHub et Binder. Ces deux dernières étant plus des services proposés pour le déploiement de serveurs Jupyter et le partage de notebooks à partir de dépôts git.

2.2.3 Architecture des applications Jupyter

L'architecture des applications Jupyter est basée sur un modèle client-serveur, où les clients interagissent avec des serveurs pour créer, modifier et exécuter des notebooks. Voici les principaux composants de l'architecture (Figure 2.3) :

- Jupyter Clients : C'est l'interface utilisateur à travers laquelle les utilisateurs interagissent avec les notebooks. Cela peut être un navigateur web (pour Jupyter Notebook, JupyterLab, Jupyter Lite), un terminal (pour Jupyter Console) ou une application de bureau (pour Jupyter Desktop).
- Jupyter Serveurs : Gère l'exécution du code, la gestion des documents et la communication avec les clients.
- Fichier notebook : C'est un fichier au format JSON avec l'extension ipynb. Ce fichier est une représentation sérialisée de la vue du notebook dans l'interface web. Il contient toutes les informations nécessaires pour reconstruire la vue d'un notebook dans son intégralité : le code, les résultats, le texte descriptif, ainsi que les divers médias et les métadonnées du notebook.
- Noyau : Environnement d'exécution pour le code dans un notebook. Chaque notebook est associé à un noyau spécifique, correspondant au langage de programmation utilisé (par exemple, Python, R, Julia).

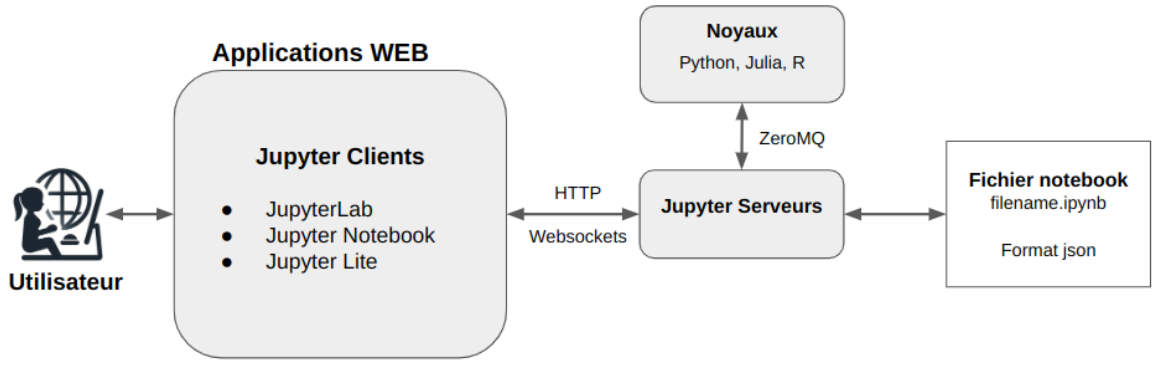


FIGURE 2.3 – La communication entre l'interface utilisateur et le noyau est effectuée à l'aide de WebSockets et ZeroMQ. WebSockets est un protocole qui permet une communication bidirectionnelle entre un client (comme un navigateur web) et un serveur via une connexion unique et persistante. Il permet au client et au serveur d'envoyer des messages à tout moment. ZeroMQ est une bibliothèque de messagerie asynchrone qui permet une communication entre processus à l'aide de sockets. Dans le contexte des Jupyter Notebooks, ZeroMQ est utilisé pour permettre la communication entre les différents composants du système Jupyter, tels que le serveur de notebook et les noyaux d'exécution.

2.2.4 Analyse d'un notebook

Ce que l'on appelle couramment un notebook correspond à l'interface utilisateur que vous voyez dans votre navigateur contenant les cellules de code et de texte (Figure 2.4). À l'ouverture d'un notebook, cette interface utilisateur représente visuellement le contenu et la structure du notebook. Cette vue est une représentation interprétée du modèle JSON sous-jacent. Chaque modification dans le notebook est enregistrée dans le fichier JSON. Le notebook s'insère dans une interface utilisateur plus large comme celle de JupyterLab ou Jupyter Notebook. Cette interface propose également à l'utilisateur des menus et des barres d'outils permettant de configurer l'environnement et d'interagir avec le contenu du notebook de manière visuelle et intuitive. Des services en ligne comme Try Jupyter [9] permettent de créer, d'éditer et d'exécuter des notebooks dans le cloud.

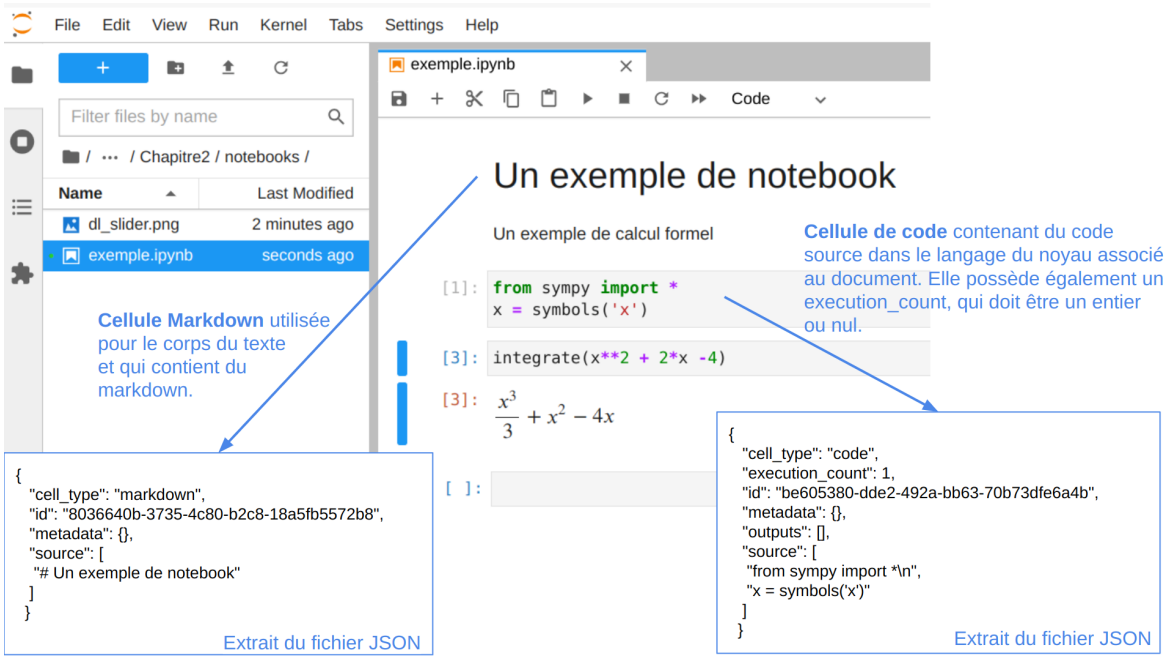


FIGURE 2.4 – Exemple d'un notebook dans l'environnement JupyterLab. Ce notebook contient 2 cellules de texte (Markdown) et 3 cellules de code. L'interface utilisateur comprend un menu permettant de gérer l'apparence, l'édition et l'exécution des cellules du notebook. Une barre d'icônes juste au dessus du notebook propose des raccourcis pour enregistrer ajouter, supprimer, exécuter ou modifier le type des cellules. Une barre latérale gauche offre du haut vers le bas : un accès à un explorateur de fichiers, aux différents noyaux actifs, à la table des matières du document notebook, et à un gestionnaire d'extensions.

Un notebook Jupyter c'est aussi un fichier JSON basé sur un dictionnaire avec quatre clés principales. La Figure 2.5 représente celui du notebook de la Figure 2.4 Voici une description succinte des clés d'un notebook :

cells (cellules) : Cette clé contient une liste de cellules qui composent le contenu du notebook. Chaque cellule dans un notebook Jupyter est également représentée comme un dictionnaire. Ce dictionnaire contient plusieurs clés qui décrivent différents aspects de la cellule, tels que l'id de la cellule, le type de cellule (code, Markdown, raw, etc.), le contenu de la cellule (le texte ou le code source), la sortie après exécution d'une cellule de code (stream, display_data, execute_result et error), les métadonnées spécifiques à la cellule et à sa sortie ou encore les pièces jointes à la cellule,

9. https ://jupyter.org/try

généralement des images intégrées qui peuvent être référencées dans le contenu Markdown d'une cellule.

metadata (métadonnées) : Les métadonnées sont des informations supplémentaires sur le notebook lui-même, telles que le titre, l'auteur, la date de création, et d'autres informations pertinentes. Ces métadonnées sont souvent utilisées pour organiser et décrire le notebook, ainsi que pour stocker des informations spécifiques aux outils ou aux extensions utilisés.

nbformat (format du notebook) : Cette clé indique la version principale du format de fichier utilisé pour stocker le notebook. Il est utilisé pour garantir la compatibilité entre différentes versions de Jupyter et pour assurer la lecture correcte du notebook par le moteur de notebook.

nbformat_minor (version mineure du format du notebook) : Cette clé spécifie la version mineure du format de fichier.

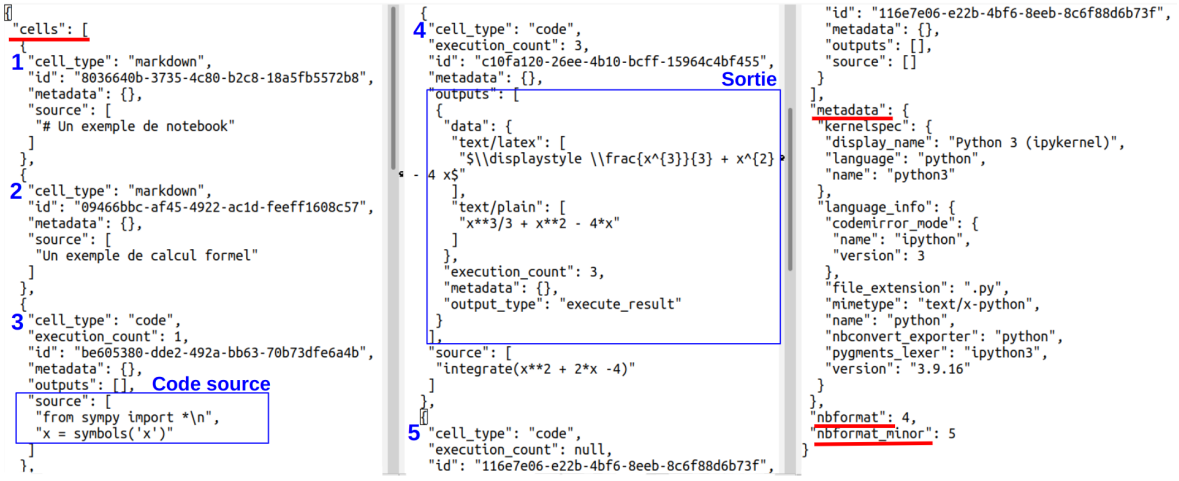


FIGURE 2.5 – En rouge les 4 clés principales du dictionnaire d'un notebook (cells, metadata, nbformat, nbformat_minor). Ce notebook contient 5 cellules numérotées en bleu. Chaque cellule est un dictionnaire contenant au minimum les clés : cell_type, id, metadata, source. Les cellules de code ont également une clé outputs et execution_count. La sortie (output) de la cellule 3 n'a pas de valeur associée contrairement à la sortie de la cellule 4 contenant le résultat de `integrate(x**2 + 2*x -4)`. La cellule 5 est une cellule de code vide, c'est la dernière cellule du notebook. la valeur de la clé execution_count est null puisque la cellule n'a jamais été exécutée

2.2.5 Interactivité et exécution du code

L'exécution d'une cellule de code dans un notebook Jupyter suit le concept de l'évaluation de code en mode REPL (Read-Eval-Print Loop) [28]. Elle est composée de 4 étapes :

1. Lecture (Read) : L'utilisateur entre son code dans une cellule de code du notebook. Cela peut être du code Python, R, Julia, ou d'autres langages pris en charge.
2. Évaluation (Eval) : Une fois que l'utilisateur a saisi le code, il peut exécuter la cellule. Le code est envoyé à l'interpréteur du langage spécifié (par exemple, l'interpréteur Python) pour être évalué. L'interpréteur exécute le code et produit un résultat.
3. Impression (Print) : Le résultat de l'évaluation est ensuite renvoyé au notebook, où il est affiché sous la cellule, généralement en tant que sortie. Cela peut être du texte, des graphiques, des tableaux, etc.

4. Boucle (Loop) : Ce processus se répète à chaque fois qu'une cellule est exécutée. L'utilisateur peut itérer rapidement sur des morceaux de code, expérimenter et voir les résultats instantanément.

L'évaluation de votre code dans un notebook est asynchrone, c'est à dire que vous pouvez continuer à utiliser les autres cellules de votre notebook pendant qu'il évalue encore la cellule que vous venez de d'exécuter.

2.3 Les notebooks comme matériel éducatif

2.3.1 Introduction

Les notebooks, principalement associés au domaine de la science des données, ont connu une diffusion importante dans le secteur de l'éducation ces dernières années. De nombreuses publications académiques récentes s'intéressent à l'introduction et l'utilisation des notebooks dans les environnements éducatifs [3, 10, 30, 16]. Par exemple, l'étude menée par Ruiz-Sarmiento et al. [24] à l'Université de Malaga (Espagne) se concentre sur la mise en oeuvre d'une collection de notebooks dans un cours de robotique. À travers différents types d'activités dont les objectifs pédagogiques s'appuient sur la taxonomie de Bloom [2, 9, 13], les résultats de cette étude montrent une amélioration significative de l'apprentissage des étudiants, soulignant ainsi le potentiel des notebooks en soutien des activités d'apprentissage. Dans un autre domaine, une étude sur l'utilisation des notebooks pour enseigner l'algorithmique conclue que c'est un outil précieux permettant d'améliorer les apprentissages et de favoriser des expériences d'apprentissage efficaces [27].

Au délà de l'intérêt, à tous points de vue, que lui porte la recherche académique, les notebooks ont également investi la vie de nombreux étudiants dans le monde. En France, l'état a lancé le programme CANDYCE [10] (CArnets Numériques DYnamiques, interactifs et Collaboratifs pour l'Enseignement) avec des acteurs comme l'Académie de Paris, l'Université de Lorraine, l'École Polytechnique, Centrale Supelec, le CNAM, l'ENSAM, etc. Ce programme encourage l'utilisation de l'environnement interactif Jupyter dans l'enseignement des sciences du numérique ainsi que dans d'autres disciplines et ce à tous les niveaux, du primaire à l'enseignement supérieur, en venant enrichir l'offre logicielle des différents ENT déployés dans les établissements. Cependant, l'engouement pour les notebooks n'est pas limité à la France. Des pays comme la Suisse, avec l'École Polytechnique Fédérale de Lausanne (EPFL [11]), ainsi que les États-Unis, avec des institutions telles que la NYU [12], Cornell [13], le MIT [14] et bien d'autres, partagent également un intérêt certain pour cet outil numérique comme support des activités d'apprentissage.

2.3.2 Composants clés (texte, code, visualisation)

Dans un contexte éducatif, le notebook offre de nombreux avantages. Par exemple la Figure 2.6 présente la première partie d'une activité d'apprentissage liée à une transformation bijective d'image sur le principe du photomaton. C'est à dire à partir d'une image de départ en obtenir 4 plus petites et ainsi de suite par itération.

10. https ://candyce.org/ProgrammeCandyce.html
11. https ://www.epfl.ch/education/educational-initiatives/jupyter-notebooks-for-education/
12. https ://guides.nyu.edu/data_management/jupyter-notebooks
13. https ://ilci.cornell.edu/fr/introduction-to-jupyterhub/
14. https ://openlearning.mit.edu/courses-programs/open-learning-library

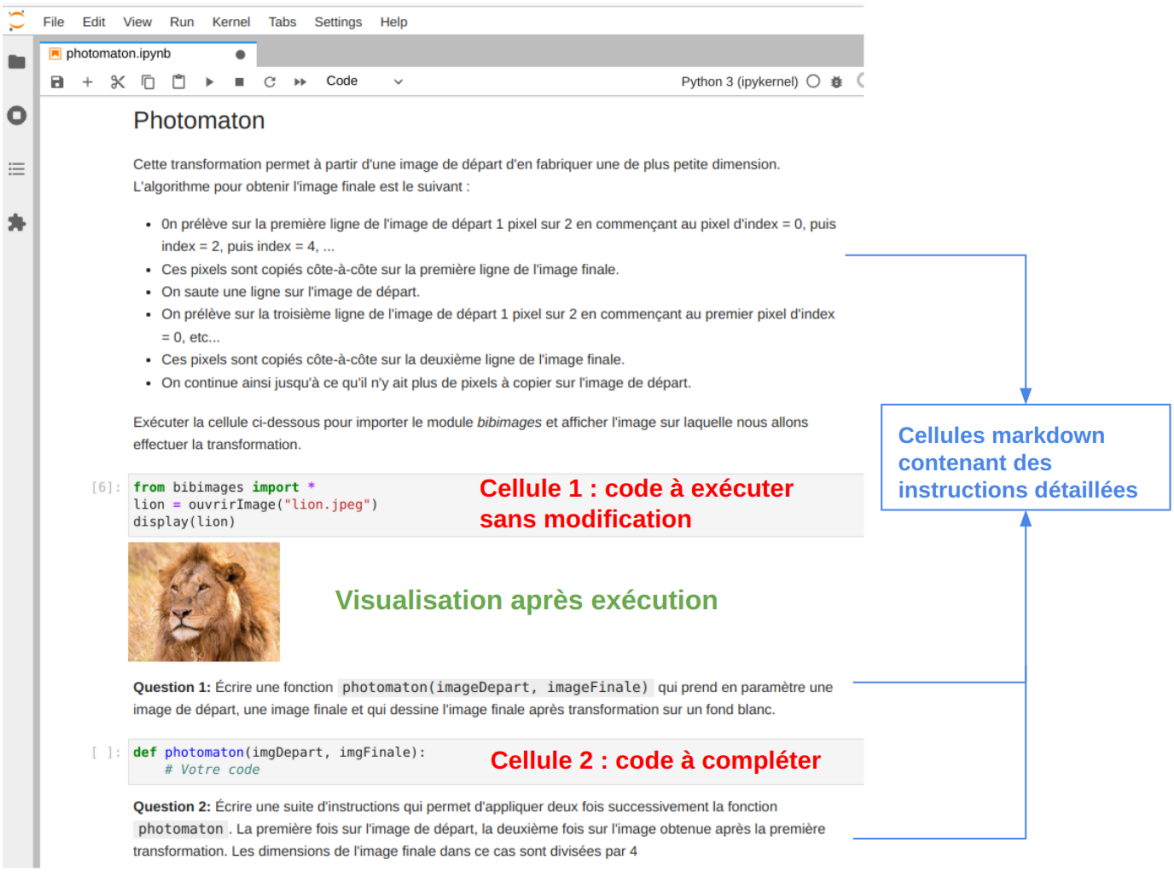


FIGURE 2.6 – Notebook contenant une activité d'apprentissage. Le notebook est un document où l'étudiant peut lire du texte expliquant un concept, écrire et exécuter du code pour mettre en pratique ce qu'il apprend, et observer directement les résultats de son code, souvent sous forme de visualisations qui peuvent être des graphiques ou des données sous forme de texte.

Le texte contenu dans les cellules Markdown joue un rôle narratif en expliquant les concepts de manière détaillée et en fournissant des instructions claires. Il sert de fil conducteur, guidant les apprenants à travers le déroulement logique d'une activité [1]. En parallèle, le code informatique injecte une dimension interactive au texte [25]. Il est possible qu'une cellule contienne juste un code à exécuter, par exemple dans la cellule 1 (Figure 2.6) le code permet de vérifier que l'étudiant a bien le module et l'image dans son répertoire courant, mais très souvent l'étudiant devra écrire du code comme dans la cellule 2. Cela permet à l'apprenant d'expérimenter directement les concepts présentés, favorisant ainsi une compréhension pratique. Les états intermédiaires du code, reflétés dans les résultats affichés, représentent les jalons marquant les progrès et les découvertes tout au long du processus d'apprentissage. Par exemple l'image de lion affichée (Figure 2.6) indique à l'étudiant que tout est en ordre et qu'il peut démarrer son activité (premier jalon indispensable). Quand l'étudiant travaille sur son notebook, il est amené à écrire du code, à l'exécuter et très souvent à le modifier car la visualisation du résultat ne correspond pas à ce qu'il attend. Les visualisations agissent comme des illustrations (quelles soient sous forme d'images, de graphiques ou plus simplement de données textuelles) aux concepts visés par les objectifs d'apprentissage contenus dans le notebook [8, 10, 27]. Elles donnent également à l'enseignant un retour visuel sur la progression de l'étudiant et les éventuels problèmes qu'il peut rencontrer. Nous présentons (Figure 2.7) un notebook dans lequel un étudiant a implémenté la fonction `photomaton` (cellule 2). Les instructions de la cellule 3 permettent de tester cette fonction. La sortie de la cellule 3 est une visualisation

de ce qui est démandé pour l'algorithme du photomaton. Grâce à cette visualisation qui transforme le code de `photomaton` en une image représentative l'étudiant peut se rendre compte qu'il ne respecte pas toutes les consignes puisque le fond de l'image est noir alors que dans l'activité il est demandé un fond blanc. L'étudiant va donc devoir modifier son code. Si rien n'est fait de la part de l'étudiant la visualisation dans son contexte narratif renseigne l'enseignant sur les difficultés de l'étudiant.

L'ordre d'exécution des cellules de code peut également ajouter une dimension supplémentaire à l'exécution du notebook en ne proposant pas forcement un ordre linéaire du haut vers le bas. Par exemple si le test de la fonction `photomaton` est satisfaisant il pourrait être demandé à l'étudiant de modifier l'image de départ dans la cellule 1 (Figure 2.6) et de relancer le test. Une image avec des caractéristiques différentes pourrait mettre en évidence des problèmes d'implémentation.

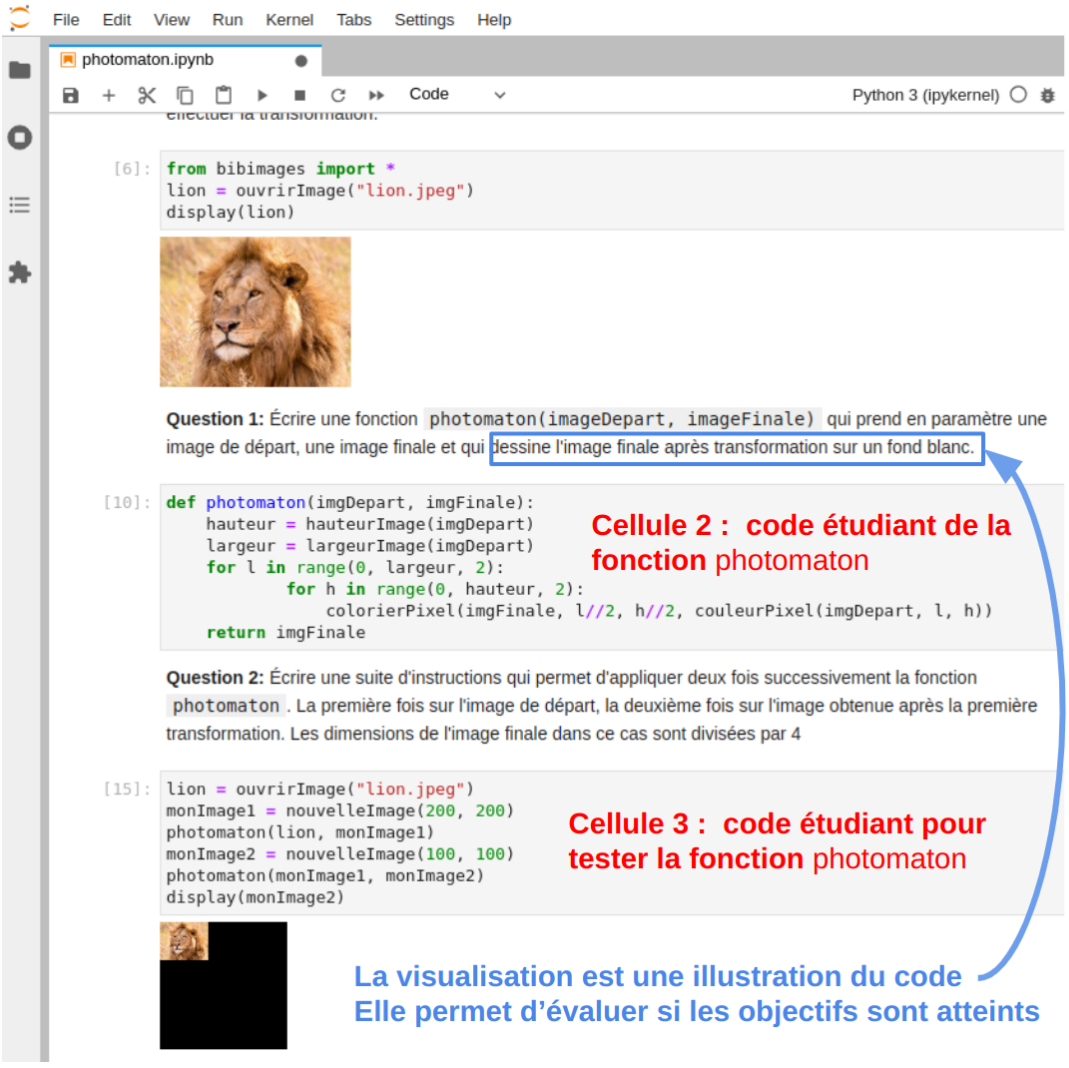


FIGURE 2.7 – Dans un document unique l'étudiant a accès aux consignes de l'activité, à l'implémentation de la fonction `photomaton` et à une visualisation de son implémentation sous forme graphique. De même l'enseignant peut visualiser rapidement l'avancer du travail de l'étudiant et intervenir en cas de difficultés.

En combinant ces composants clés dans un même document, les notebooks offrent une plateforme intégrée où les utilisateurs peuvent non seulement accéder à des explications textuelles, mais aussi interagir via du code et visualiser les résultats de manière

itérative. Cela favorise l'apprentissage actif, la résolution de problèmes et l'exploration des données, ce qui en fait un outil précieux pour l'enseignement.

2.3.3 Aspects des notebooks dans un contexte éducatif

Une caractéristique commune aux notebooks contenant des activités d'apprentissage de type exercices ou problèmes ouverts est d'encourager régulièrement les étudiants à rédiger, modifier et exécuter du code. Les deux principaux traits distinctifs de ces notebooks sont l'alternance entre des cellules de texte et des cellules de code avec des cellules de code qui la plupart du temps sont laissées intentionnellements vides (Figure 2.8) [15]. Nous présenterons dans l'état de l'art un exemple de programmation orientée objet (POO) spécialement adapté pour les notebooks. Dans cet exemple l'auteur fait le choix pédagogique de répartir les méthodes d'une classe Python sur plusieurs cellules. La fréquente inclusion de cellules de code vides offrant ainsi un espace significatif destiné à l'interaction des apprenants. À contrario, les notebooks destinés à du cours ont souvent leurs cellules de code déjà remplies afin de faciliter une première exécution servant à illustrer les propos du cours.

2.3.4 Terminologie

Comme cela a été souligné dans de nombreuses études [4, 5, 17, 26, 29], la technologie des notebooks Jupyter propose un cadre pour le développement de matériel pédagogique précieux pour atteindre avec succès les objectifs pédagogiques dans une large variété de paradigmes éducatifs.

La particularité des notebooks dans ce contexte éducatif est que dans la plupart des cas ils incarnent une activité d'apprentissage [7, 15], c'est à dire une situation planifiée par l'enseignant et proposée à l'étudiant pour l'aider à atteindre un objectif d'apprentissage. Les activités d'apprentissage peuvent prendre plusieurs formes : un cours, des exercices, une situation problème, une évaluation, un projet en groupe ou de l'apprentissage inversé. L'activité d'apprentissage comporte généralement une ou plusieurs tâches à accomplir qui peuvent se structurer en quatre grandes parties : mise en situation, expérimentation, objectivation, réinvestissement. Le déroulement des tâches à accomplir par l'étudiant est décrit par le scénario de l'activité [6, 21] d'apprentissage que nous définissons pour un notebook de la manière suivante :

Définition 1. *Dans un notebook, le scénario d'une activité d'apprentissage est défini par la structure narrative présente dans les cellules de texte du notebook. Cette configuration narrative établit le déroulement des étapes ou actions structurées visant à atteindre les acquis d'apprentissage de l'activité.*

Les notebooks embarquant des activités d'apprentissage seront appelés dans la suite de cette thèse des notebooks éducatifs. En s'appuyant sur la définition précédente, une définition très générale d'un notebook éducatif peut-être formulée de la manière suivante :

Définition 2. *Un notebook éducatif est un document numérique offrant à l'enseignant la possibilité de mettre en œuvre des activités d'apprentissage interactives. L'activité d'apprentissage embarquée dans ce notebook combine le scénario de cette activité avec des cellules de code, et des visualisations permettant d'atteindre des acquis d'apprentissage spécifiques.*

15. https ://ix.cs.uoregon.edu/ conery/eic/python/ipython/journey/index.html

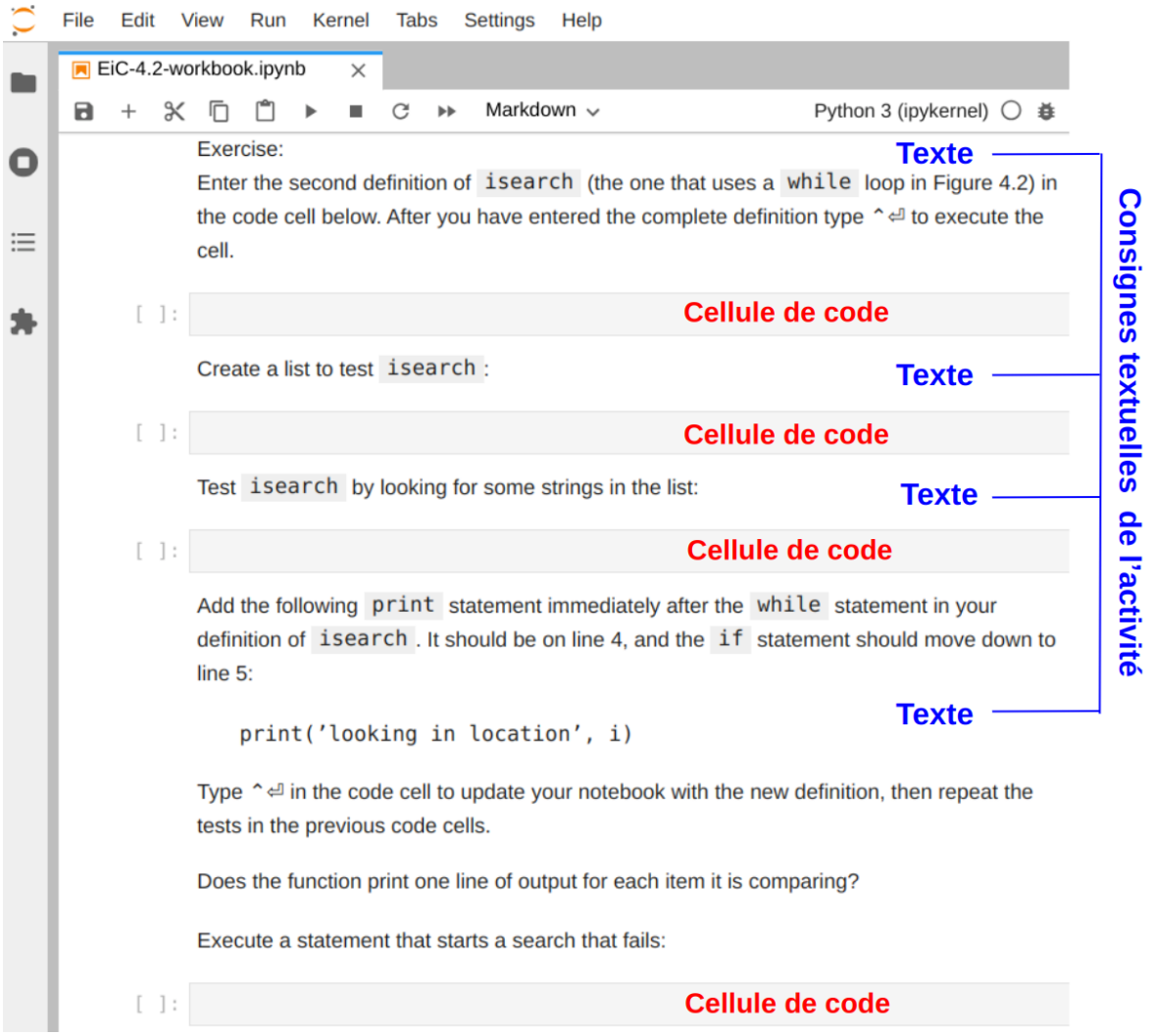


FIGURE 2.8 – Extrait d'un notebook contenant une activité d'apprentissage avec plusieurs exercices sur la notion de recherche séquentielle. Les cellules de texte donnent des consignes aux étudiants pour remplir les cellules de code.

À chaque interaction de l'étudiant avec le notebook, telles que l'édition, l'exécution ou l'ajout de contenu, l'état du notebook est modifié. Il existe essentiellement deux manières de concevoir cet état : l'état de la mémoire et l'état représenté par le fichier JSON associé au notebook. L'état de la mémoire est volatile. En revanche, l'état du notebook tel que représenté dans le fichier JSON est sauvegardé de manière permanente, contenant des métadonnées sur le notebook lui-même, ainsi que des détails sur chaque cellule de code ou de texte.

Dans le contexte de cette thèse, l'état du notebook se réfère directement au contenu du fichier JSON. Par conséquent, nous définissons l'état d'un notebook comme suit :

Définition 3. *L'état du notebook est une structure de données qui enregistre les informations spécifiques aux cellules du notebook dans un fichier JSON. Cette structure inclut des détails sur chaque cellule individuellement, tels que son type (code ou texte), son contenu, son état d'exécution, ainsi que d'autres métadonnées propres à chaque cellule. Cette représentation JSON capture le notebook dans un document fournissant une vue structurée de ses composants fondamentaux, facilitant ainsi la gestion et la manipulation du contenu du notebook.*

2.3.5 Cycle de vie d'un notebook éducatif

De sa conception jusqu'à un retour d'usage, un notebook éducatif passe par plusieurs phases distinctes. Cette succession de phases peut-être décrite comme le cycle de vie du notebook (Figure 2.9). Il est important de souligner que l'enseignant doit avoir une approche itérative, incrémentale et flexible de ce cycle dans le but d'une amélioration continue du notebook éducatif. Bien que les produits finaux diffèrent il y a une grande similitude entre le cycle de vie d'un logiciel et celui d'un notebook éducatif [23]. Les étapes du cycle de vie impliquées dans leur développement et leur utilisation présentent des similitudes dans la façon dont ils sont planifiés, créés, utilisés, évalués et mis à jour pour répondre aux besoins des utilisateurs finaux. Voici une description générale de ces étapes pour le notebook éducatif :

Conception : Cette phase initiale implique la planification et la conception du notebook. Les enseignants définissent les objectifs d'apprentissage, identifient les sujets à couvrir, et décident de la structure globale du contenu. Ils peuvent également sélectionner les médias à inclure, tels que des textes explicatifs, du code informatique et des visualisations.

Développement : Pendant cette étape, les enseignants et éducateurs créent effectivement le contenu du notebook. Cela peut inclure l'écriture de textes pédagogiques, la rédaction de code informatique, et la création d'éléments visuels.

Tests et révisions : Avant de déployer le notebook auprès des étudiants, il est essentiel de le tester. Cela implique de s'assurer que le contenu est fonctionnel, que le code informatique s'exécute correctement, et que les exercices interactifs sont pertinents. Les tests peuvent également inclure la collecte de retours d'enseignants ou d'autres collaborateurs.

Déploiement : Une fois que le notebook a été développé et testé, il est déployé dans le cadre éducatif. Les étudiants ont alors accès au contenu, qu'ils peuvent explorer de manière interactive pour renforcer leur compréhension.

Utilisation en classe : Les enseignants intègrent le notebook dans leurs sessions d'enseignement. Ils peuvent l'utiliser pour des démonstrations en classe, des exercices pratiques, ou des devoirs. Pendant cette phase, les enseignants peuvent également observer comment les étudiants interagissent avec le contenu.

Rendu : Durant cette phase les étudiants doivent nettoyer leur notebook pour s'assurer de la cohérence du scénario pédagogique et des résultats.

Retour d'usage : La dernière phase du cycle de vie d'un notebook éducatif s'attache à établir un bilan du scénario de l'activité d'apprentissage. L'objectif est double : d'une part évaluer l'efficacité du scénario en termes didactiques et pédagogiques, et d'autre part, identifier des points d'amélioration potentiels pour le scenario d'apprentissage afin d'être en phase avec les objectifs d'apprentissage.

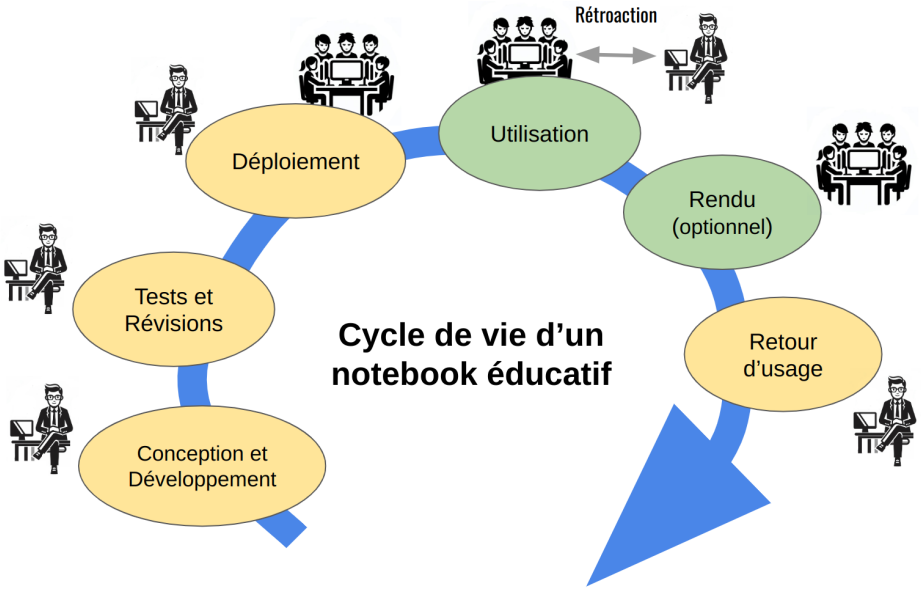


FIGURE 2.9 – Le cycle de vie du notebook présente les principales phases du processus d'évolution du notebook dans un contexte éducatif. Les phases précédentes au déploiement peuvent suivre un modèle incrémental, c'est à dire que le processus du cycle de vie avant déploiement se répète, et à chaque répétition le notebook est enrichi jusqu'à ce que tous les objectifs d'apprentissage soient atteints à travers l'activité proposée. Le retour d'usage permettra de s'assurer que l'activité contenu dans le notebook remplie bien ses objectifs. Dans le cas contraire l'enseignant pourra actualiser le contenu de son notebook pour un prochain déploiement.

2.4 Les défis à relever

Durant la phase de développement, l'enseignant va effectuer de nombreuses actions telles que l'ajout, l'édition, l'exécution, le déplacement et éventuellement la suppression de cellules de code ou de texte (Figure 2.10) qui vont lui permettre de développer le notebook à donner aux étudiants. Le notebook destiné aux étudiants est donc le notebook éducatif contenant l'activité d'apprentissage avec les instructions censées guider l'étudiant durant son travail. Une fois le notebook éducatif finalisé par l'enseignant, il est donc dans un état prêt à être déployé (Figure 2.9). Cet état correspond à l'état initial du notebook que les étudiants reçoivent et que nous désignons comme un notebook dans un état valide.

Définition 4. *Un état est dit valide lorsqu'il correspond à un état prévu par le scénario de l'activité embarquée dans le notebook.*

L'état initial du notebook correspond également à l'état initial du scénario. Les instructions contenues dans le notebook sont maintenant censées guider les étudiants de l'état initial vers une succession d'états valides du notebook c'est à dire conformes au scénario de l'activité. Il convient de souligner que cette progression n'est pas nécessairement linéaire. Durant cette progression et de part le modèle d'exécution du notebook qui permet d'exécuter les cellules de code dans n'importe quel ordre, les étudiants ont accès à une infinité d'états possibles. Parmi cette infinité il existe un sous ensemble d'états valides définissant des chemins qui peuvent être suivis pour aller du notebook initial au notebook final prêt à être soumis à l'enseignant (Figure 2.10). Ce chemin peut être composé exclusivement d'états valides (en vert) ou bien contenir des états non valides (en

rouge). Un chemin valide peut donc être composé uniquement d'états valides mais peut également contenir des états non valides si et seulement si il se termine avec un état valide. Attention un chemin valide n'implique en aucun cas un notebook exécutable et/ou reproductible. En l'absence de toute assistance, les premières difficultés pour l'étudiant sont de différencier un état valide d'un état non valide et de revenir dans un état valide si le notebook est dans un état non valide c'est à dire non prévu par le scénario de l'activité.

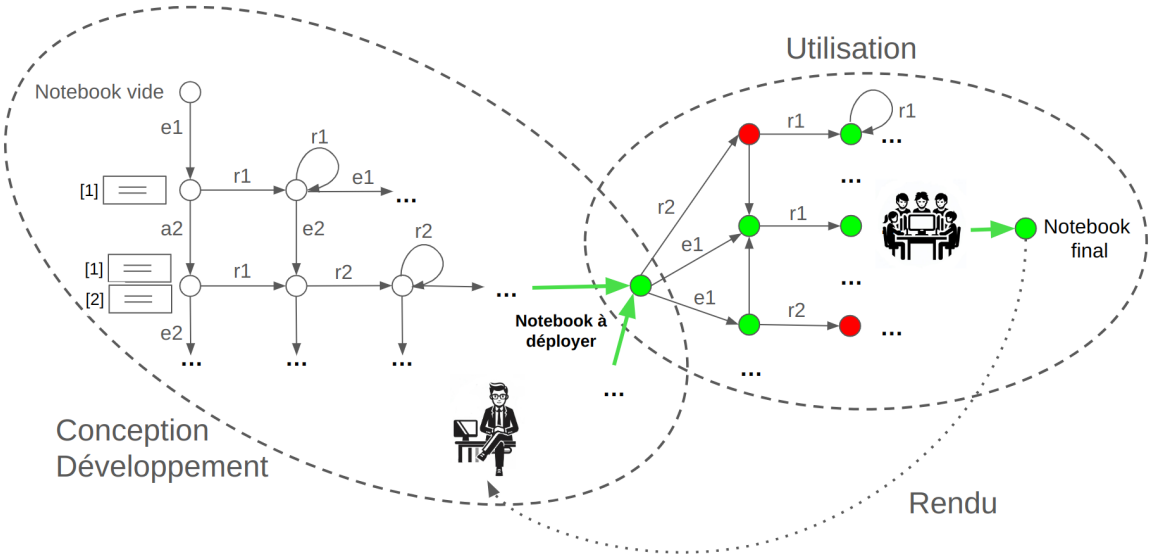
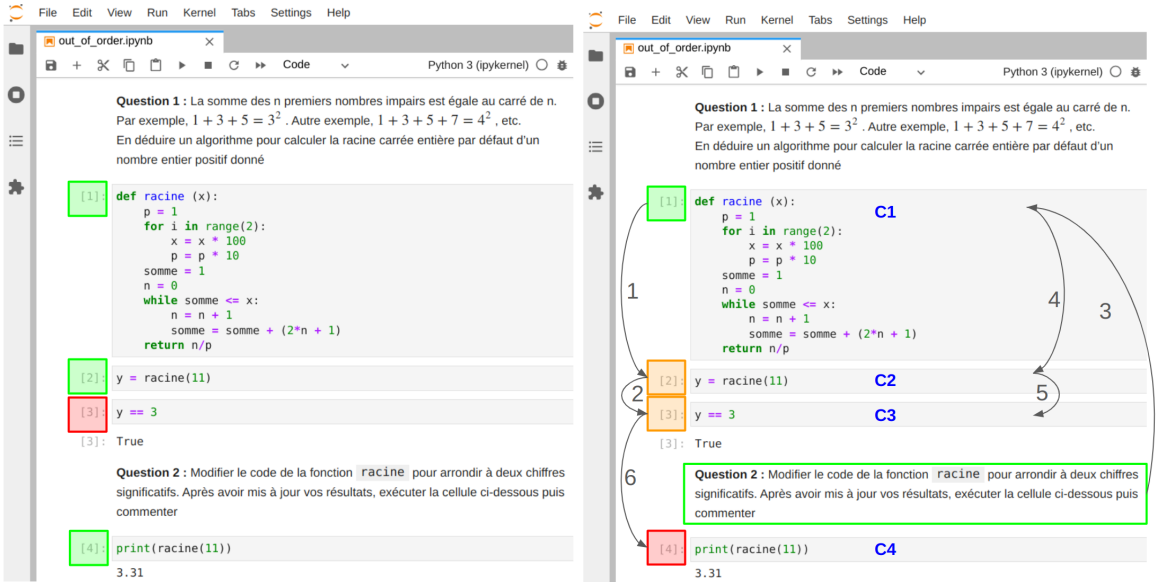


FIGURE 2.10 – Les sommets représentent les états d'un notebook. Le notebook à déployer correspond à l'état initial du notebook étudiant. Un sommet vert ● correspond à un état valide c'est à dire à un état prévu par le scénario de l'activité. Un sommet rouge ● indique un état non valide donc non prévu par le scénario de l'activité. L'ensemble des sommets verts donne l'ensemble des états valides autorisés par le scénario. Les arrêtes indiquent les différentes actions réalisées par l'utilisateur : ajouter (a), éditer (e), exécuter (r), supprimer ou déplacer une cellule de code ou de texte.

Une fois l'activité achevée une deuxième difficulté est la phase de rendu. En effet, dans le meilleur des mondes, l'étudiant doit s'assurer que les résultats de son notebook sont bien reproductibles. Si l'exécution du notebook n'est pas linéaire, en plus de corriger les eventuelles erreurs l'étudiant va devoir également retrouver l'ordre dans lequel il a exécuté les différentes cellules de son notebook. La phase de rendu peut donc engendrer un surcroit de travail parfois long et fastidieux. L'intérêt porté à ces deux problèmes récurrents pour les étudiants les a transformés en deux défis associés à une notion plus large qu'est la reproductibilité.

2.4.1 Reproductibilité des résultats

Pour ce premier défi nous nous intéressons à la reproductibilité des résultats dans le cas de l'exécution linéaire d'un notebook. Ce n'est pas le seul modèle d'exécution mais c'est sans doute un des plus courants et il a l'avantage d'être automatisé par le menu *Run All* de l'interface Jupyter qui exécute les cellules de code séquentiellement de haut en bas. Prenons un exemple avec une activité portant sur le calcul de la racine carrée d'un nombre entier (Figure 2.11a). La première question expose une propriété sur le calcul de la racine carrée entière à partir des nombres impairs et demande aux étudiants d'implémenter cet algorithme dans une première cellule de code, nommée `C1`, en créant une



(a) La couleur verte indique une sortie reproductible lors d'une exécution du haut vers le bas et en rouge une sortie qui n'est pas reproductible.

(b) Les flèches mettent en évidence l'ordre des actions à réaliser pour garder un notebook cohérent avec le scénario de l'activité. C'est à dire une succession d'états valides. Les couleurs indiquent la progression de l'étudiant dans l'activité. En vert la prochaine action, en orange ce qui a déjà été fait et en rouge ce qu'il n'est pas conseillé de faire dans l'immédiat.

FIGURE 2.11 – Notebook éducatif contenant un extrait d'une activité d'apprentissage sur l'algorithme de calcul de la racine carrée d'un nombre à partir d'une suite de nombres impairs

fonction appelée `racine`. Les deux cellules de code suivantes (`C2` `C3`) permettent aux étudiants de tester la fonction `racine`. Pour trouver une implémentation correcte de la fonction, les étudiants vont avoir une phase exploratoire durant laquelle ils devront sans doute modifier et/ou exécuter plusieurs fois les trois premières cellules (`C1` `C2` `C3`). La question suivante demande aux étudiants de modifier le code de la fonction `racine` pour obtenir cette fois une valeur approchée de la racine d'un nombre avec deux chiffres significatifs. Après avoir modifié le code dans la cellule (`C1`), les étudiants peuvent décider d'exécuter directement la dernière cellule (`C4`) sans mettre à jour les cellules intermédiaires (`C2` `C3`). Quand l'étudiant remet son notebook à l'enseignant, celui-ci s'attend à pourvoir reproduire les résultats obtenus avec l'exécution linéaire des cellules (`C1` `C2` `C3` `C4`). Or si les cellules `C2` et `C3` n'ont pas été mises à jour, l'exécution de la cellule `C3` par l'enseignant indiquera `False` ce qui en fait une cellule non reproductible. Nous cherchons donc à proposer une solution au défi suivant :

Défi n°1

Comment accompagner les étudiants vers des résultats reproductibles dans un notebook éducatif dont le scénario d'exécution est linéaire ?

2.4.2 Reproductibilité du scénario de l'activité d'un notebook éducatif

Le deuxième défi concerne le suivi du scénario contenu dans une activité afin d'éviter tant que possible les états non valides des notebooks éducatifs. Un notebook est censé offrir une certaine flexibilité permettant aux étudiants d'explorer les contenus mais également ajouter de nouvelles cellules, en réexécuter d'autres, et même en supprimer. Et ce faisant ils n'ont aucun moyen de savoir si les consignes du scénario ont été bien suivies.

Reprenons notre exemple (Figure 2.11) et considérons sous un nouvel angle l'état dans lequel l'étudiant a laissé son notebook juste avant le rendu (Figure 2.11b). Soit juste après avoir modifié le code dans la cellule (`C1`), l'étudiant décide d'exécuter directement la dernière cellule (`C4`) sans mettre à jour les cellules intermédiaires (`C2` `C3`). Dans ce cas le notebook est dans un état non valide car il ne respecte pas la consigne qui est de mettre à jour les résultats en exécutant les cellules intermédiaires (`C2` `C3`). Une fois le travail terminé, si les mises à jour prévues par le scénario de l'activité n'ont pas été faites, l'état non valide du notebook persiste et mène à une incohérence pour la valeur de `racine(11)` qui est différente dans la sortie de la cellule `C3` ($\sqrt{11} = 3$) et celle de la cellule `C4` ($\sqrt{11} = 3.31$). La présence répétée de telles incohérences dans un notebook contenant de nombreuses questions peut entraver l'apprentissage en générant des erreurs d'exécution, et/ou des résultats incorrects, ce qui peut nuire gravement à la qualité de l'apprentissage.

Pimentel et al. [22] ont montré qu'en l'absence d'instructions supplémentaires dans le notebook l'ordre d'exécution des cellules n'est pas toujours cohérent avec une exécution linéaire du haut vers le bas des cellules. Cela se traduit par toutes sortes d'erreurs à l'exécution des notebooks ou alors par des résultats non reproductibles quand les notebooks sont exécutables sans erreur. De plus l'exécution linéaire d'un notebook, bien que très courante, ne représente qu'une instance spécifique. Certaines études [11, 14] dont nous reparlerons dans l'état de l'art, proposent des solutions pour maintenir une certaine cohérence entre le contenu du notebook et les résultats affichés dans celui-ci. Ainsi, la seconde partie de cette thèse cherche à développer une approche pour accompagner les étudiants dans le respect du scénario de l'activité lors des phases exploratoires et s'intéresse à la faisabilité d'élargir les cas d'utilisation du notebook à des scénarios non linéaires.

Défi n°2

Comment accompagner les étudiants pour améliorer le suivi du scénario d'une activité contenue dans un notebook éducatif sans sacrifier les phases exploratoires ?

En plus de ces deux défis, une contrainte supplémentaire liée au contexte éducatif a été identifiée. Comme observé précédemment, les notebooks peuvent être employés dans différentes disciplines (physique, chimie, mathématique, informatique, histoire, géographie, ...) et dans différents contextes (algorithmique, IA, POO, traitement des données, ...) sans aucune garantie sur le langage de programmation utilisé. Pour essayer de satisfaire le plus large éventail possible de situations d'enseignement, nous nous sommes fixés l'objectif suivant :

Objectif commun aux défis 1 et 2

Garantir que les deux défis identifiés seront instanciés à l'aide d'approches indépendantes des langages de programmation utilisés dans les notebooks éducatifs.

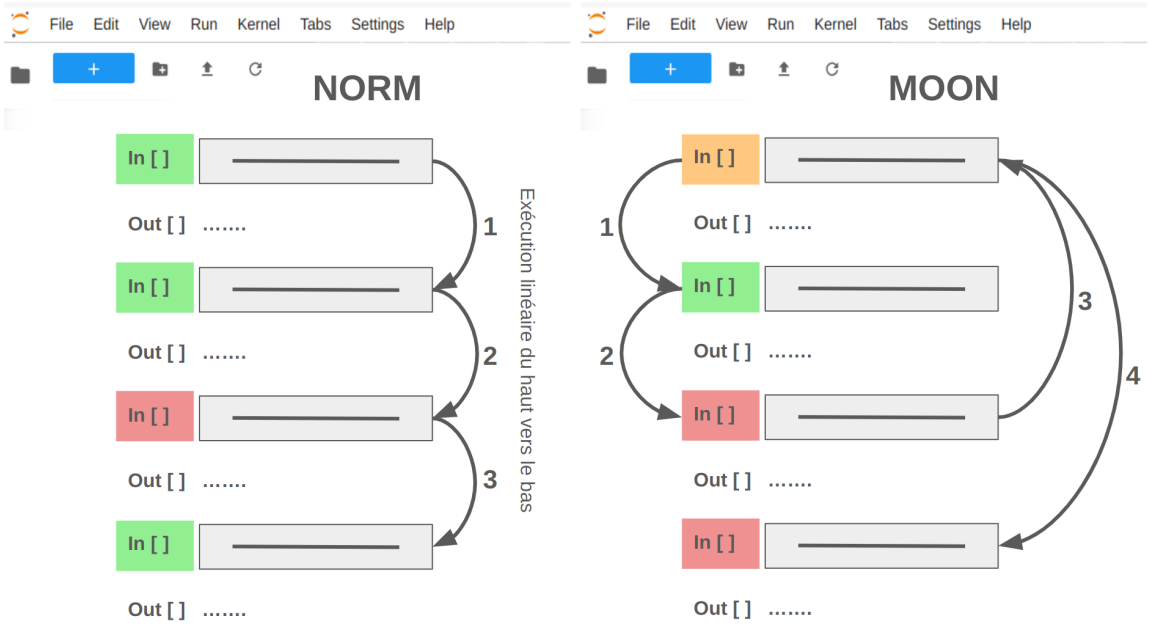


FIGURE 2.12 – Sur la partie gauche le retour visuel de NORM indique en vert les sorties des cellules de code reproductibles lors d'une exécution linéaire du haut vers le bas. En rouge les sorties qui ne sont pas reproductibles. Sur la partie droite MOON affiche en vert la ou les prochaines actions possibles prévues par le scénario de l'activité. En orange ce que l'étudiant a déjà validé et en rouge les actions à ne pas réaliser dans l'immédiat. Sur les deux figures les flèches indiquent un ordre d'exécution possible prévu par le scénario de l'acivité embarquée dans le notebook.

2.5 Contributions

Les approches proposées pour répondre aux défis enoncés dans la section précédente consiste à introduire dans la plateforme Jupyter des outils avec des fonctionnalités d'assistance pour les étudiants afin de faciliter l'utilisation du notebook éducatif. Dans le cadre de notre recherche nous avons développé :

NORM est un plugin JupyterLab très simple d'utilisation capable d'évaluer la reproductibilité des résultats d'un notebook éducatif lorsque l'étudiant doit exécuter les différentes cellules de code de manière linéaire, du haut vers le bas (Figure 2.12). Bien que Jupyter dispose d'un menu *Run All* qui permet cela, NORM présente l'avantage d'offrir une assistance visuelle aux étudiants en temps réel et à chaque exécution d'une cellule de code c'est à dire a chaque nouvel état du notebook. Il permet de distinguer avec des couleurs différentes les résultats reproductibles de ceux qui ne le sont pas permettant ainsi aux étudiants de ne pas cumuler des états cachés trop nombreux et difficiles à déméler par la suite. L'étudiant peut ainsi s'assurer que les résultats du notebook lors du rendu à l'enseignant seront reproductibles.

MOON est un plugin JupyterLab permettant à l'enseignant de scripter le scénario de l'activité d'apprentissage présentée dans le notebook éducatif. Le scénario ainsi embarqué dans le notebook peut-être activé par l'étudiant qui sera guidé à l'aide d'un code couleur tout au long de sa progression dans l'activité (Figure 2.12). MOON présente aussi l'avantage de laisser une grande liberté aux étudiants pendant les phases exploratoires de code. Si l'étudiant s'écarte du scénario, le plugin suggère, à chaque exécution d'une cellule de code, une voie à suivre pour rétablir un notebook cohérent avec le scénario. MOON garantit ainsi que les cellules de code du notebook

éducatif ont été exécutées dans un ordre respectant le scénario de l'activité écrit par l'enseignant.

2.6 Références

[1] Lorena A BARBA et al. "Teaching and learning with Jupyter". In : *https://jupyter4edu.github. io/jupyter-edu-book* (2019), p. 1-77.

[2] Benjamin S BLOOM et David R KRATHWOHL. *Taxonomy of educational objectives: The classification of educational goals. Book 1, Cognitive domain*. longman, 2020.

[3] Dhruva K CHAKRAVORTY et al. "Evaluating active learning approaches for teaching intermediate programming at an early undergraduate level". In : *The Journal of Computational Science Education* 10.1 (2019).

[4] Sutrini CLAUDIO, Davide PASSARO, Pallotta FILIPPO et al. "The potential of using jupyter notebook in physics education: Experimentation for high school students". In : *Il nuovo cimento C* (2022), p. 1-4.

[5] Alessio DE SANTO et al. "Promoting Computational Thinking Skills in Non-Computer-Science Students: Gamifying Computational Notebooks to Increase Student Engagement". In : *IEEE Transactions on Learning Technologies* 15.3 (2022), p. 392-405.

[6] Sandrine DECAMPS, Bruno DE LIÈVRE et Christian DEPOVER. "Entre scénario d'apprentissage et scénario d'encadrement. Quel impact sur les apprentissages réalisés en groupes de discussion asynchrone ?" In : *Distances Et Savoirs* 7 (juin 2009), p. 141-154. DOI : 10.3166/ds.7.141-154.

[7] Philippe DESSUS ALL MY PAPERS ARE AT PDESSUS.FR. "Qu'est-ce que l'enseignement ? Quelques conditions nécessaires et suffisantes de cette activité". In : *Revue française de pédagogie* (oct. 2008). DOI : 10.2307/41202407.

[8] Alex Daniel EDGCOMB et al. "Student performance improvement using interactive textbooks: A three-university cross-semester analysis". In : *2015 ASEE Annual Conference & Exposition*. 2015, p. 26-1423.

[9] William HUITT. "Bloom et al.'s taxonomy of the cognitive domain". In : *Educational psychology interactive* 22 (2011), p. 1-4.

[10] David H. Smith IV et al. "Towards Modeling Student Engagement with Interactive Computing Textbooks: An Empirical Study". In : *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*. Sous la dir. de Mark SHERRIFF et al. ACM, 2021, p. 914-920. DOI : 10.1145/3408877.3432361. URL : https://doi.org/10.1145/3408877.3432361.

[11] Mary Beth KERY et Brad A MYERS. "Interactions for untangling messy history in a computational notebook". In : *2018 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE. 2018, p. 147-155.

[12] Donald E. KNUTH. "Literate Programming". In : *Comput. J.* 27.2 (mai 1984), p. 97-111. ISSN : 0010-4620. DOI : 10.1093/comjnl/27.2.97. URL : https://doi.org/10.1093/comjnl/27.2.97.

[13] David R KRATHWOHL. "A revision of Bloom's taxonomy: An overview". In : *Theory into practice* 41.4 (2002), p. 212-218.

[14] Stephen MACKE et al. "Fine-Grained Lineage for Safer Notebook Interactions. CoRR abs/2012.06981 (2020)". In : *arXiv preprint arXiv:2012.06981* (2020).

[15] Khaldi MAHA et al. "The educational scenario architecture of a learning situation". In : *Global Journal of Engineering and Technology Advances* 3.1 (2020), p. 027-040.

[16] Nnamdi I NWULU, Uyikumhe DAMISA et Saheed Lekan GBADAMOSI. "Students Perception about the Use of Jupyter Notebook in Power Systems Education." In : *Int. J. Eng. Pedagog.* 11.1 (2021), p. 78-86.

[17] Keith J. O'HARA, Douglas S. BLANK et James B. MARSHALL. "Computational Notebooks for AI Education". In : *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2015, Hollywood, Florida, USA, May 18-20, 2015*. Sous la dir. d'Ingrid RUSSELL et William EBERLE. AAAI Press, 2015, p. 263-268. URL : http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS15/paper/view/10349.

[18] Fernando PEREZ, Brian E GRANGER et CPSL OBISPO. *An open source framework for interactive, collaborative and reproducible scientific computing and education.* 2013.

[19] Fernando PEREZ et Brian E. GRANGER. "IPython: A System for Interactive Scientific Computing". In : *Computing in Science & Engineering* 9.3 (2007), p. 21-29. DOI : 10.1109/MCSE.2007.53.

[20] Jeffrey M PERKEL. "Why Jupyter is data scientists' computational notebook of choice". In : *Nature* 563.7732 (2018), p. 145-147.

[21] Jean-Philippe PERNIN et Anne LEJEUNE. "Modèles pour la réutilisation de scénarios d'apprentissage". In : *TICE Méditerranée, Nice* (2004).

[22] João Felipe PIMENTEL et al. "A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks". In : *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, p. 507-517. DOI : 10.1109/MSR.2019.00077.

[23] W. W. ROYCE. "Managing the development of large software systems: concepts and techniques". In : *Proceedings of the 9th International Conference on Software Engineering*. ICSE '87. Monterey, California, USA : IEEE Computer Society Press, 1987, p. 328-338. ISBN : 0897912160.

[24] Jose-Raul RUIZ-SARMIENTO, Samuel-Felipe BALTANAS et Javier GONZALEZ-JIMENEZ. "Jupyter notebooks in undergraduate mobile robotics courses: Educational tool and case study". In : *Applied Sciences* 11.3 (2021), p. 917.

[25] Alessio De SANTO et al. "Promoting Computational Thinking Skills in Non-Computer-Science Students: Gamifying Computational Notebooks to Increase Student Engagement". In : *IEEE Trans. Learn. Technol.* 15.3 (2022), p. 392-405. DOI : 10.1109/TLT.2022.3180588. URL : https://doi.org/10.1109/TLT.2022.3180588.

[26] Chiin-Rui TAN, Sharon Broude GEVA et Dirk COLBRY. "The Nascent Case for Adopting Jupyter Notebooks as a Pedagogical Tool for Interdisciplinary Humanities, Social Science, and Arts Education". In : *Comput. Sci. Eng.* 23.2 (2021), p. 107-113. DOI : 10.1109/MCSE.2021.3062199. URL : https://doi.org/10.1109/MCSE.2021.3062199.

[27] Oguzhan TOPSAKAL. "Teaching Algorithms Design Approaches via Interactive Jupyter Notebooks". In : *European Journal of Technique (EJT)* 13.1 (), p. 1-6.

[28] L Thomas VAN BINSBERGEN et al. "A principled approach to REPL interpreters". In : *Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. 2020, p. 84-100.

[29] Charles J. WEISS. "A Creative Commons Textbook for Teaching Scientific Computing to Chemistry Students with Python and Jupyter Notebooks". In : *Journal of Chemical Education* 98.2 (2021), p. 489-494. DOI : 10.1021/acs.jchemed.0c01071. URL : https://doi.org/10.1021/acs.jchemed.0c01071.

[30] Alistair WILLIS, Patricia CHARLTON et Tony HIRST. "Developing Students' Written Communication Skills with Jupyter Notebooks". In : *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE '20. New York, NY, USA : Association for Computing Machinery, fév. 2020, p. 1089-1095. ISBN : 978-1-4503-6793-6. DOI : 10.1145/3328778.3366927. URL : https://doi.org/10.1145/3328778.3366927 (visité le 07/05/2021).

Chapitre 3

État de l'art

« *On ne se découvre qu'en se tournant vers ce que l'on n'est pas.* »

Paul Auster

Sommaire

Ce chapitre se décompose en deux parties. Pour faire suite aux défis identifiés dans le chapitre précédent, nous aborderons dans un premier temps la notion de reproductibilité appliquée aux notebooks. Nous débuterons par le rappel des définitions fondamentales qui ont constitué nore socle de départ et la nécessité de la reproductibilité dans le contexte scientifique. Nous présenterons ce que cela implique dans le domaine du génie logiciel, pour ensuite nous concentrer sur une problématique spécifique aux notebooks, à savoir l'impact de l'ordre d'exécution des cellules de code sur la reproductibilité d'un notebook. Nous terminerons avec les outils qui ont été développés pour assister dans ce cadre les utilisateurs de notebooks. Dans la deuxième partie, nous discuterons de l'utilisation des notebooks dans un contexte éducatif toutes disciplines confondues puis plus particulièrement avec des étudiants en informatique.

3.1 Reproductibilité

3.1.1 Introduction

La reproductibilité [22, 14, 42] se réfère à la capacité de reproduire ou de recréer de manière cohérente et fiable les résultats, les expériences ou les conditions d'un processus ou d'une expérience. Cela signifie qu'un tiers, en suivant la même méthodologie ou les mêmes étapes, devrait être en mesure d'obtenir des résultats similaires ou identiques. La reproductibilité est un concept crucial dans divers domaines, tels que la recherche scientifique [19], le développement logiciel, l'ingénierie, et d'autres disciplines où la fiabilité des résultats est essentielle. Si les résultats d'une expérience ne peuvent pas être reproduits par d'autres chercheurs, la validité de ces résultats peut être remise en question. En 2016 dans un article de Nature, Monya Baker [5] explique que dans une enquête menée auprès de 1 576 chercheurs plus de 70% d'entre eux ont essayé en vain de reproduire les expériences d'un autre scientifique, et plus de la moitié ont échoué à reproduire leurs propres expériences. La reproductibilité est essentielle pour établir la crédibilité, la transparence et la confiance dans les résultats d'une expérience, d'un projet de recherche ou d'un processus.

Dans le contexte particulier du développement logiciel, la reproductibilité implique souvent la possibilité de recréer l'environnement exact dans lequel le logiciel a été testé ou exécuté, garantissant ainsi que le code fonctionne de manière cohérente dans différentes conditions. Comme exemple une étude [23] de 2018 propose une mise en oeuvre pratique et cinq recommandations pour aider un chercheur à s'engager sur la voie de l'analyse de données reproductible à l'aide de technologies de conteneurisation et de virtualisation. Les principaux éléments contribuant à la reproductibilité incluent la documentation complète des méthodes, l'utilisation de systèmes de gestion de version pour suivre les modifications, la spécification claire des environnements d'exécution, la disponibilité des données utilisées, et l'automatisation des tests. En 2021 Chris Lamb et al. [35] donne une définition du processus de construction reproductible d'un logiciel : *Le processus de construction d'un logiciel est dit reproductible si, après avoir désigné une version spécifique de son code source et l'ensemble de ses dépendances, chaque construction produit des artefacts identiques bit par bit, peu importe l'environnement dans lequel elle est réalisée.* Néanmoins, parvenir à une plus grande reproductibilité en ingénierie logicielle reste un grand défi pour la recherche, l'éducation et l'industrie [3, 13, 48].

3.1.2 Terminologies existantes

Plusieurs communautés de recherche se sont engagées à promouvoir la reproductibilité en science expérimentale. Par exemple la recherche en ingénierie logicielle comprend souvent des prototypes, des preuves de concept ou des outils de mesure, désignés comme artefacts de recherche et soumis à l'évaluation lors de conférences scientifiques. De nombreuses conférences en informatique organisent des évaluations d'artefacts dans le but d'assurer la reproductibilité. Une analyse de dix ans de conférences en ingénierie logicielle et en langages de programmation a révélé que les articles contenant des artefacts évalués n'ont pas nécessairement plus de visibilité, mais met en lumière des pistes pour améliorer le processus d'évaluation des artefacts [61]. Malheureusement une harmonisation transversale sur l'ensemble de ces communautés est longue et difficile et des terminologies conflictuelles persistent [6], se divisant principalement en deux groupes. Le premier groupe ne fait aucune distinction entre les termes "reproduire" et "répliquer", tandis que le second groupe, se basant sur une norme minimale pouvant être résumée par *mêmes données - mêmes méthodes = mêmes résultats*, se divise en deux camps utilisant tantôt le terme *reproduire*, tantôt le terme *répliquer* pour désigner la norme minimale. Une initiative récente de l'ACM [1] travaille sur la délivrance de badges pour le respect de certaines normes de partage de codes et de données dans les articles de recherche. Dans cette optique et en accord avec la NISO [2] dont l'objectif principal d'identifier, développer, maintenir et publier des normes techniques et des pratiques recommandées pour gérer l'information dans l'environnement numérique en constante évolution d'aujourd'hui, l'ACM définit la terminologie suivante :

Définition 5. *Répétabilité (Même équipe, même configuration expérimentale)*
La mesure peut être obtenue avec une précision déclarée par la même équipe en utilisant la même procédure de mesure, le même système de mesure, dans les mêmes conditions de fonctionnement, au même endroit lors de plusieurs essais. Pour les expériences computationnelles, cela signifie qu'un chercheur peut répéter de manière fiable ses propres calculs.

Définition 6. *Reproductibilité (Équipe différente, même configuration expérimentale)*
La mesure peut être obtenue avec une précision déclarée par une équipe différente en utilisant la même procédure de mesure, le même système de mesure, dans les mêmes conditions de fonctionnement, au même endroit ou à un endroit différent lors de plusieurs essais. Pour les expériences computationnelles, cela signifie qu'un groupe indépendant peut obtenir le même résultat en utilisant les artefacts fournis par l'auteur.

Définition 7. *Réplicabilité (Équipe différente, configuration expérimentale différente)*
La mesure peut être obtenue avec une précision déclarée par une équipe différente, un système de mesure différent, dans un endroit différent lors de plusieurs essais. Pour les expériences computationnelles, cela signifie qu'un groupe indépendant peut obtenir le même résultat en utilisant des artefacts qu'il développe complètement indépendamment.

Une terminologie connexe a été proposée par Goodman et al. [22] cherchant à éviter l'ambiguïté des termes reproductibilité, replicabilité et répétabilité. On remarquera une certaine similitude entre ces deux terminologies. L'article propose un nouveau lexique avec les définitions suivantes :

1. Association for Computing Machine - Artifact Review and Badging Version 1.1 - August 24, 2020. https://www.acm.org/publications/policies/artifact-review-and-badging-current#appendix
2. National Information Standards Organization https://www.niso.org/welcome-to-niso

Définition 8. *La reproductibilité des méthodes*
Elle vise à capturer le sens original de la reproductibilité, c'est-à-dire la capacité de mettre en œuvre, aussi exactement que possible, les procédures expérimentales et computationnelles, avec les mêmes données et outils, pour obtenir les mêmes résultats.

Définition 9. *La reproductibilité des résultats*
Elle fait référence à l'obtention des mêmes résultats à partir de la réalisation d'une étude indépendante dont les procédures sont aussi étroitement adaptées que possible à l'expérience originale.

Définition 10. *La reproductibilité inférentielle*
Elle fait référence à la formulation de conclusions qualitativement similaires à partir soit d'une nouvelle étude indépendante, soit d'une réanalyse de l'étude originale.

La reproductibilité inférentielle n'est pas identique à la reproductibilité des résultats ou à la reproductibilité des méthodes, car les scientifiques pourraient tirer les mêmes conclusions à partir de différents ensembles d'études et de données, ou pourraient tirer des conclusions différentes à partir des mêmes données originales, parfois même s'ils sont d'accord sur les résultats analytiques. En s'inspirant de ce lexique, d'autres études proposent des définitions adaptées à un domaine particulier comme par exemple les articles de Gundersen et al. [24] ou de Bouthillier et al. [8] pour des applications d'apprentissage profond dans le domaine de l'intelligence artificielle. L'article de Gundersen et al. introduit également un ensemble de métriques pour quantifier dans quelle mesure une méthode est reproductible.

Dans le cadre de cette thèse, nous nous sommes appuyés pour le défi n°1 sur la définition 6 de la reproductibilité des résultats c'est à dire que pour des expériences computationnelles un groupe indépendant (l'enseignant) peut obtenir les mêmes résultats en utilisant les artefacts (notebook et autres) fournis par l'auteur (l'étudiant). Cette définition correspond à la situation ou l'enseignant cherche à reproduire les résultats obtenus par l'étudiant dans son notebook lors d'un rendu.

3.1.3 Les principales causes d'un notebook non reproductible

De part sa nature le notebook encourage à suivre un paradigme de développement orienté *literate programming* et offre un moyen pratique de développer du code tout en documentant le processus d'écriture et d'exécution de manière claire et on l'espère reproductible. Certains auteurs de notebooks vont même jusqu'à justifier l'organisation de leur code en faisant référence à la structure spécifique du notebook lui-même. Ils préconisent par exemple que pour la programmation orienté objet il est préférable de découper le code en fragments de code courts entrecoupés d'explications (Figure 3.1). [3]

Cependant, une étude de 2019 menée par Pimentel et al. [43] indique que parmi 800 000 notebooks extraits de dépôts GitHub, seulement 25% d'entre eux, considérés comme valides sont effectivement exécutables. Un notebook est considéré comme valide quand il paraît possible de déterminer l'ordre d'exécution des cellules à partir du numéro d'exécution (*exécution count*) qui se trouve à gauche d'une cellule. En effet, chaque fois qu'une cellule de code est exécutée dans un notebook, elle se voit automatiquement attribuer un numéro croissant de manière strictement monotone. En théorie, cette information devrait permettre d'en déduire l'ordre d'exécution des différentes cellules en utilisant la relation d'ordre établie par ce mécanisme (Figure 3.1). Sans donner de définition précise de

3. https ://d2l.ai/chapter_linear-regression/oo-design.html

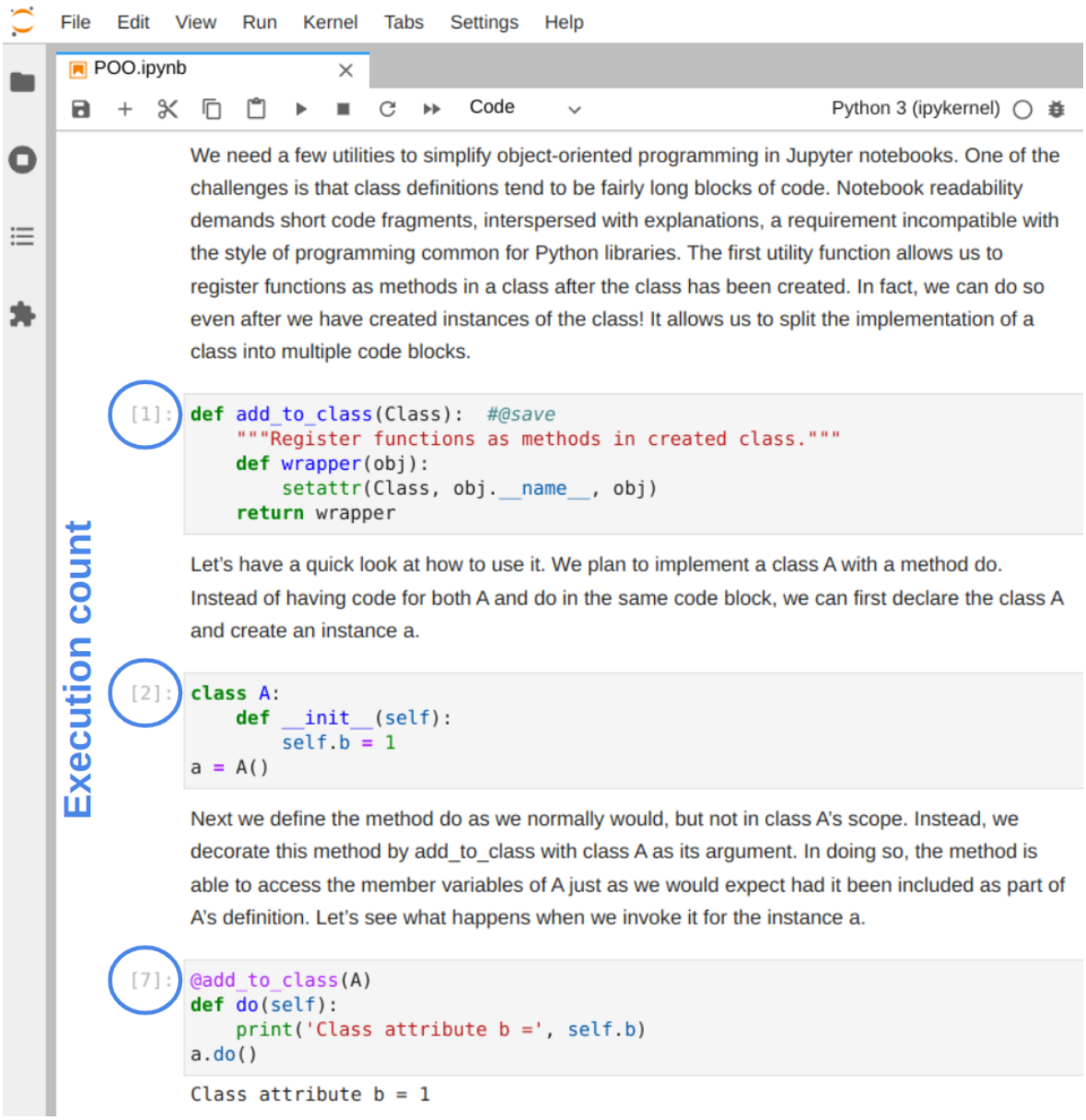


FIGURE 3.1 – Extrait d'un notebook sur la POO qui commence par définir une fonction utilitaire permettant de tisser de nouvelles méthodes au sein d'une classe dont une instance existe déjà. Cela permet de diviser l'implémentation d'une classe en plusieurs petits blocs de code.

la reproductibilité, cette étude indique également que seuls 4% de ces notebooks exécutables peuvent être reproduits avec les mêmes résultats. Les auteurs résument les causes courantes rendant les notebooks non exécutables et/ou non reproductibles. Parmi celles-ci il y a les dépendances manquantes, l'accessibilité des données externes aux notebooks ainsi que le modèle d'exécution intrinsèque au notebook permettant d'exécuter les cellules dans n'importe quel ordre.

Les deux premières causes peuvent être étudiées dans un cadre beaucoup plus large lié à la reproduction d'environnements logiciels. De nombreuses études se sont attelées à ces problèmes [13, 57]. Par exemple une équipe Inria propose un noyau Guix [16] spécifique aux notebooks Jupyter. Ce noyau permet d'avoir des notebooks autosuffisants en capturant toutes les informations nécessaires pour permettre un déploiement reproduc-

tible. Dans le même ordre d'idée, il existe *ReproZip*[4] [40] développé par une équipe NYU[5] afin d'aider les utilisateurs à capturer automatiquement les dépendances (y compris les données, les variables d'environnement, etc.) des notebooks, et à configurer automatiquement ces dépendances dans un autre environnement informatique.

Notre recherche laisse de coté ces deux premières causes et se focalise sur l'importance de l'ordre d'exécution des cellules qui constitue une autre cause fondamentale rendant les notebooks non reproductibles [43, 59, 27, 33, 51, 50, 49, 53]. Nous attirons l'attention sur le fait que l'ordre d'exécution des cellules d'un notebook peut le rendre exécutable et non reproductible. L'hypothèse centrale est que, idéalement, en respectant un ordre d'exécution approprié des cellules, il serait possible d'assurer la reproductibilité des notebooks dans un environnement ou toutes les autres dépendances seraient satisfaites.

3.1.4 Paradigme d'exécution des notebooks

Les notebooks Jupyter ont instauré un modèle d'exécution de code lié à la notion de cellules. Un notebook est un document composé d'un ensemble de cellules de code et de texte, où les cellules de code peuvent être exécutées dans n'importe quel ordre. Bien qu'ayant été largement adopté par de nombreuses communautés scientifiques en raison de ses nombreuses qualités, la nécessité de prêter attention à l'ordre d'exécution des cellules s'est rapidement manifestée. Par défaut, l'interface des notebooks Jupyter propose un menu *Run All* permettant d'exécuter les cellules de manière linéaire, du haut vers le bas. Cependant, dans de nombreux contextes, l'utilisation d'autres schémas d'exécution des cellules du notebook peut être pertinente, par exemple pour vérifier si des modifications apportées à une étape analytique antérieure ont un impact sur les calculs ultérieurs. Les notebooks destinés à l'enseignement contiennent également des schémas incitant les étudiants à des allers-retours entre les cellules de code. Plusieurs exemples de ce type sont disponibles sur le site de l'Université de l'Orégon[6] dans la section enseignement des sciences de l'informatique. Par exemple dans la section 7 de cet enseignement intitulé : *Définition de nouveaux objets : déclarations de classe* nous avons un notebook dont nous présentons un extrait sur la figure 3.2. Dans cette activité l'étudiant est amené à compléter de manière itérative le constructeur d'une classe qui se trouve dans la première cellule du notebook au fur et à mesure de sa progression. Les figures 3.1 et 3.2 présentent deux approches différentes de la POO dans un notebook, venant souligner dans la figure 3.1 un aspect pédagogique prioritaire par rapport à une conception plus classique de la notion de classe en POO (Figure 3.2). Rule et al. [49] estiment dans une étude de 2018 que sur un échantillon d'environ un million de notebooks collectés sur GitHub, 43.9% ont un ordre d'exécution non linéaire. Ce résultat met en évidence le fait que de nombreux notebooks sont utilisés pour des analyses itératives et rares sont ceux qui fournissent des explications détaillées du modèle d'exécution de leur notebook.

Lors d'une présentation à la Jupytercon de 2018, Joel Grus dans une présentation intitulée *I don't like notebooks*[7] met en garde les utilisateurs de notebooks sur l'ordre d'exécution des cellules. La figure 3.3 illustre un exemple de notebook exécutable mais pas reproductible. Ce qui est particulièrement étonnant sur la figure 3.3b, c'est que les numéros situés à gauche de chaque cellule de code semble suggérer que les cellules ont été exécutées dans un ordre conforme aux attentes. Il est important de noter que dans les

4. https ://docs.reprozip.org/en/1.x/jupyter.html
5. https ://engineering.nyu.edu/
6. http ://ix.cs.uoregon.edu/ conery/eic/python/ipython/index.html
7. https ://conferences.oreilly.com/jupyter/jup-ny/public/schedule/detail/68282.html

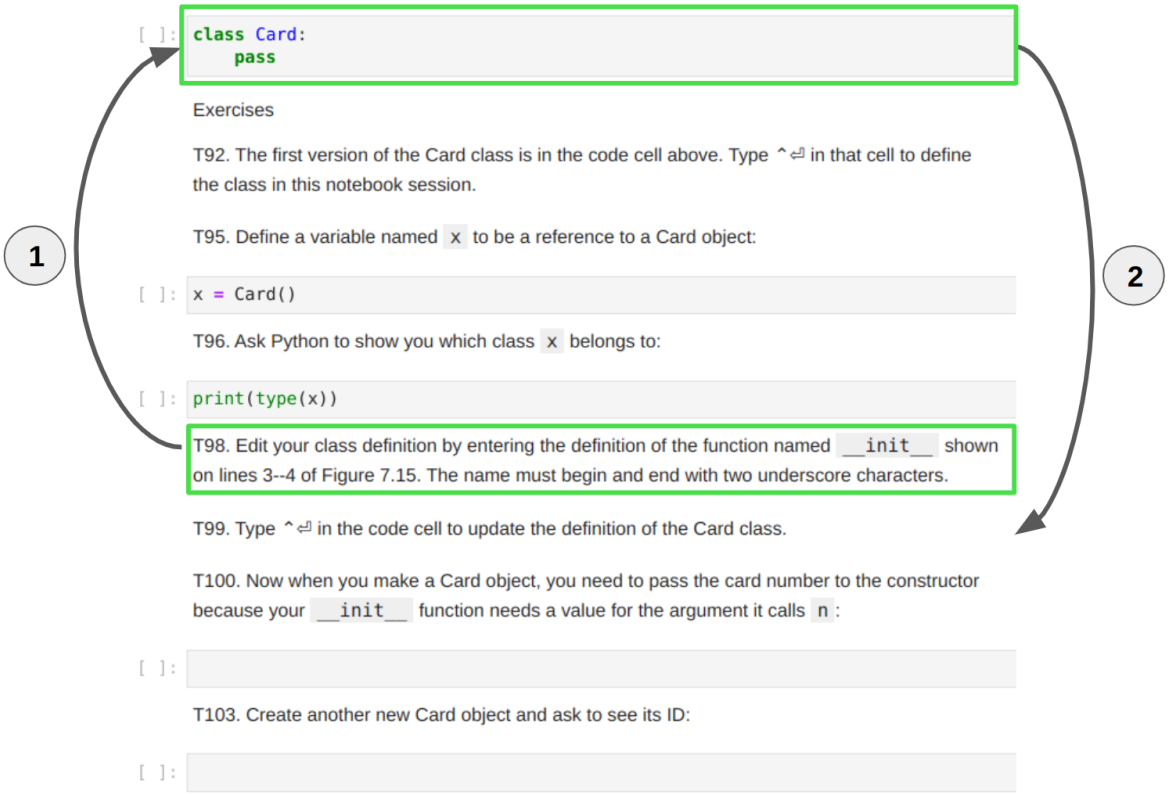
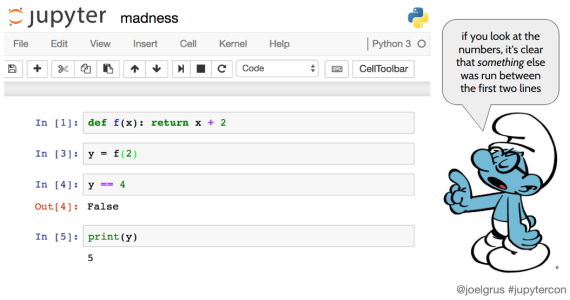


FIGURE 3.2 – Extrait d'un notebook sur la POO où l'étudiant doit effectuer des allers-retours dans la première cellule pour la compléter au fur et à mesure de sa progression dans l'exercice. En vert ce que l'étudiant doit faire dans l'immédiat.

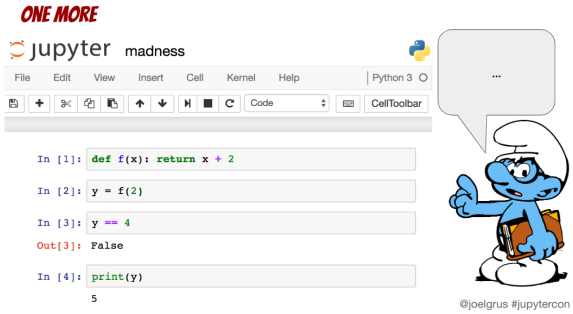
deux exemples présentés la sortie de la cellule contenant le code Python `y == 4` donne un résultat qui n'est pas correct puisque une éxécution séquentielle des cellules de haut vers le bas, comme attendu, devrait indiquer `True` au lieu de `False`. Ce qui devient problématique si on décide de partager le notebook.

À la même époque, Santana et al. [54] présentent une étude systématique des bugs et des défis auxquels sont confrontés les utilisateurs de notebooks Jupyter. Pour cela ils ont utilisés 14 740 commits de 105 projets open source sur GitHub liés à des notebooks ainsi que 30 416 publications sur Stack Overflow. Ce travail à permis d'élaborer une taxonomie des bugs associés aux notebooks Jupyter. Cette taxonomie des bugs introduit la catégorie *Implémentation* (StackOverflow - 22% | GitHub - 44.2%) englobant entre autres les erreurs de syntaxe et les erreurs sémantiques. Ces erreurs sont clairement associées par les auteurs à des problèmes de reproductibilité soit parce que le code ne s'exécute pas (erreurs syntaxiques) soit le code s'exécute correctement, mais son exécution génère une sortie différente de celle attendue (erreurs sémantiques). Les bugs les plus courants référencés dans cette catégorie sont ceux liés à la duplication de cellules de code et ceux liés à l'exécution des cellules de code dans un ordre qui n'est pas correct. Cette catégorie de bugs se positionne également comme la deuxième catégorie la plus fréquemment observée parmi les huit catégories examinées dans cette étude (Kernel, Conversion, Portability, Environment and settings, Connection, Processing, Cell Defect and Implementation), la première étant *Environment and setting* (StackOverflow - 43.2% | GitHub - 35.6%) faisant référence aux problèmes d'accès et/ou de version des bibliothèques ou encore à des problèmes d'installation de package en fonction des systèmes d'exploitation.

La possibilité d'exécuter les cellules dans n'importe quel ordre n'a pas que des défauts.



(a) Le code semble indiquer que la sortie `Out[4]` ne donne pas la bonne réponse qui devrait être : `True`. L'explication se trouve dans l'ordre d'exécution des cellules qui n'est pas celui que l'on pourrait déduire de la séquence de nombres à gauche des cellules.



(b) Les numéros à gauche des cellules indiquent sans ambiguïté une séquence d'exécution du haut vers le bas et pourtant en contradiction avec le résultat obtenu pour la sortie `Out[3]`

FIGURE 3.3 – Sur la gauche de chaque cellule, il y a un numéro qui augmente de manière strictement croissante à chaque exécution d'une cellule de code. Ce numéro est censé permettre d'en déduire l'ordre d'exécution des différentes cellules de code. Malheureusement les exemples des figures 3.3a et 3.3b montrent que cette information n'est pas fiable.

Cela s'avère particulièrement utile pour des communautés comme celle des sciences des données afin d'écrire et d'exécuter du code de manière incrémentale, ce qui facilite l'exploration des données et des algorithmes. Une étude de Chattopadhyay et al. [12] interviewant une vingtaine de *data scientists* souligne toutefois la difficulté même pour des utilisateurs avertis de garder l'historique des exécutions durant la phase exploratoire de leur travail entraînant des états de la mémoire non consistant avec l'exécution linéaire d'un notebook du haut vers le bas. S'engage alors un travail long et fastidieux de réorganisation du notebook [25, 29].

Toutes ces études ont montré la nécessité d'accompagner l'utilisateur au cours de son travail sur les notebooks afin de préserver la reproductibilité des résultats en assurant une cohérence entre l'ordre d'exécution des cellules et la documentation embarquée dans le notebook.

3.1.5 Les outils assistant la reproductibilité

Pour palier le problème de la reproductibilité des résultats liée à l'ordre d'exécution des cellules, la littérature propose principalement trois approches distinctes. La première consiste en une suite de bonnes pratiques à mettre en oeuvre pour élaborer, tester et partager les notebooks. La deuxième approche cherche à accompagner l'utilisateur tout au long de son travail sur le notebook, qui est l'approche adoptée dans cette thèse. La dernière approche a pour objectif d'évaluer la capacité du notebook à être reproduit une fois celui-ci achevé.

Les bonnes pratiques

Commençons par explorer les études traitant des bonnes pratiques liées aux notebooks Jupyter. De nombreux papiers, notes et articles de blog abordent ce sujet dans divers domaines tels que l'éducation [27, 39], le partage d'artéfacts dans la recherche scientifique [45, 51], et même le génie logiciel [46]. Dans un contexte général, rendre les notebooks reproductibles nécessite souvent un effort substantiel et un investissement en temps important, car cela implique de prendre en compte l'ensemble des facteurs ayant

un impact sur la reproductibilité des notebooks. Parmi toutes les bonnes pratiques énoncées dans la littérature, nous nous concentrons spécifiquement sur celles liées à l'ordre d'exécution des cellules de code. L'ensemble des études s'accordent pour dire que l'exécution des cellules de manière désordonnée, la suppression et la modification de cellules déjà exécutées contribuent fortement à engendrer des états de la mémoire susceptibles de gêner la reproductibilité. En l'absence d'outils permettant de s'assurer de la consistance des états successifs du notebook nous retiendrons 4 bonnes pratiques liées à l'exécution des celulles de code qui sont à encourager lors de la conception d'un notebook et pour son utilisation.

- **Exécuter linéairement les cellules de code du haut vers le bas et dans le cas contraire documenter sans ambiguïté la façon d'exécuter le notebook** [27, 45, 43, 58, 51]. Cette documentation, élaborée en parallèle avec le code, sert de guide de lecture pour son auteur d'origine permettant d'exécuter le notebook à l'avenir, mais elle est surtout utile lorsque le notebook est partagé. Les notebooks bien documentés peuvent être utilisés pour diffuser de manière transparente les résultats, permettant à d'autres utilisateurs de repoduire le notebook. En plus de commenter le processus d'exécution il est également recommandé de documenter les résultats.
- **Garder le notebook clair et concis** [45, 58, 51]. Tous les avantages de la programmation littéraire perdent de leur efficacité si les notebooks deviennent trop compliqués. Durant la conception du notebook, les phases exploratoires peuvent rapidement rendre le notebook difficile à lire, à comprendre et à exécuter sans état caché [25]. Il est donc important de réorganiser régulièrement son notebook et de faire attention aux cellules de code mal documentées et surtout non exécutées. Une accumulation de ces cellules critiques se trouvent souvent dans le bas du notebook [43].
- **Redémarrer le noyau et exécuter le notebook fréquemment** [27, 43, 51]. L'interactivité des notebooks les rend vulnérables à l'écrasement de code dans les cellules ou à la suppression accidentelle de celles-ci, Il est donc préférable de redémarrer régulièrement le noyau et d'exécuter toutes les cellules pour s'assurer de la consistence du notebook durant les phases exploratoires et de nettoyage.
- **S'engager dans la voie des bonnes pratiques du génie logiciel** [38, 27, 43, 58, 51] Le notebook contenant du code, les bonnes pratiques de développement et de génie logiciel s'y appliquent tout autant qu'à du code classique (conventions stylistiques, modularisation, tests et gestion des versions).

Assistance à la conception et/ou à l'utilisation du notebook

Les notebooks sont devenus des outils essentiels dans le domaine de l'analyse des données, et donc très utilisés par les professionnels de ce domaine communément appelés *data scientists*. De nombreuses recherches se sont concentrées sur la question de la reproductibilité des notebooks dans ce contexte.

Plusieurs études [52, 31] décrivent le processus d'obtention d'informations à partir des données comme ayant principalement 3 phases : i) préparation des données, ii) analyse des données et iii) communications des résultats. La phase d'analyse peut-être comparée à une phase exploratoire intense sous la forme d'un cycle itératif d'analyse, où le code est écrit, inspecté, modifié et débogué. Durant cette phase exploratoire il est relativement facile pour les développeurs de créer des états non cohérents dans le notebook engendrant des problèmes de reproductibilité. Il s'avère donc important d'apporter une assistance au développement de ce type de notebook.

Cette thèse s'intéresse principalement aux notebooks liés à l'enseignement. Les notebooks proposés aux étudiants se présentent souvent comme des feuilles d'exercices plus ou moins longues dans lesquelles les étudiants sont amenés à écrire du code, à le modifier et à le déboguer. Il y a donc une grande similarité avec la phase exploratoire à laquelle sont confrontés les data scientists et le risque de se retrouver avec des états du notebook non cohérent. Nos étudiants sont beaucoup moins préparés que des développeurs professionels et démêler des états non cohérents trop fréquents est un véritable défi pour eux qui peut engendrer frustration et découragement.

Nous allons donc commencer par passer en revue quelques plugins d'assistance dans le domaine de l'analyse des données.

Dataflow notebook En 2017, une étude de Koop et al. [33] présente le lien étroit entre reproductibilité et ordre d'exécution des cellules. Cette étude se penche sur la question de la reproductibilité d'un notebook lorsque les cellules ne sont pas exécutées de manière séquentielle, du haut vers le bas. Ils présentent dans leur étude une extension Jupyter appelée *dataflow notebook* qui introduit un nouvel identifiant unique et persistant pour chaque cellule de code du notebook, ainsi que pour les sorties correspondantes de ces cellules. Ils utilisent ces références pour permettre l'exécution récursive des cellules en amont, en enregistrant un graphe de dépendances et en recalculant ces dépendances lorsque nécessaire. Cette approche vise à assurer une meilleure cohérence des résultats.

Verdant est un outil développé par Myers et al. [28, 30] plus spécifiquement conçu pour le domaine de l'analyse des données. Verdant permet de capturer à chaque étape, la totalité des informations du notebook dans une structure arborescente. Le noeud racine est le notebook lui même et chaque cellule du notebook est un noeud enfant. Une version entière du notebook est capturée à chaque fois qu'une cellule de code est exécutée. Verdant permet ainsi de naviguer dans les différentes versions du notebook et il est ensuite capable de proposer les étapes à suivre pour essayer de reproduire une sortie sélectionnée par l'utilisateur (sous réserve que les données sous-jacentes utilisées n'aient pas changé). Cependant, un domaine où Verdant ne fournit pas de support direct est l'expérimentation avec des versions alternatives du code et la création de branches pour suivre ces versions. Un des points intéressant que nous avons retenu pour développer nos propres outils d'assistance est que Verdant fonctionne quelque soit le langage utilisé dans le notebook.

HISTREE [55] est une extension inspirée de Verdant et propose d'expérimenter de nouvelles branches avec d'anciennes versions du notebook. Cette extension affiche dans une barre latérale l'historique des modifications du notebook sous la forme d'un arbre dont les noeuds correspondent à une ou un ensemble d'actions dans le notebook, comme par exemple l'insertion, l'exécution, la suppression, le déplacement d'une cellule ou l'exécution de plusieurs cellules. HISTREE donne la possibilité à l'utilisateur de cliquer sur un des noeuds pour passer à une version différente du notebook. L'utilisateur a alors accès au notebook dans l'état ou il se trouvait pour le noeud cliqué. Celui-ci devient alors le noeud actuel permettant ainsi de développer si nécessaire une nouvelle branche dans l'arbre des historiques.

LOOPS [20] L'objectif de cette extension est de soutenir et de rendre la phase d'analyse compréhensible. Les modifications du contenu et de la structure du notebook sont regroupées en états puis affichés soit sous la forme d'une représentation compacte d'un historique qui donne un aperçu de l'évolution du notebook en termes de structure, de temps et de contenu, soit sous la forme d'une représentation détaillée qui

révèle comment le contenu des cellules a changé. L'ensemble des représentations peut-être affiché à l'aide d'un onglet dans la barre latérale de JupyterLab. Chaque fois que les utilisateurs exécutent une cellule dans le notebook et la mettent ainsi à jour, LOOPS enregistre les cellules du notebook, leur ordre, la cellule active et exécutée, la totalité des entrées et sorties pour ensuite proposer des artéfacts d'analyse des différentes versions du notebook.

NBSAFETY Stephen Macke et al. [37] ont développé NBSAFETY permettant de détecter d'éventuels problèmes liés entre autres à l'ordre d'exécution des cellules. Cet outils utilise la trace d'exécution et de l'analyse statique pour gérer automatiquement les dépendances à l'exécution des cellules et la cohérence de l'état global du notebook.

MARG Contrairement aux outils précédents, le plugin MARG [47] pour JupyterLab ne propose pas d'historique, mais offre la possibilité d'explorer narrativement les flux de travail non linéaires dans notebook. Ils sont représentés sous forme d'arbres ordonnés, où chaque nœud correspond à une cellule du notebook, affichant des informations supplémentaires telles que le type d'activité effectuée dans la cellule et le numéro d'exécution. Les noeuds possédants plusieurs fils sont des noeuds de divergence. L'exploration de cet arbre permet de mieux comprendre les séquences non linéaires présentes dans le notebook.

Analyse post-mortem du notebook

Une étude plus récente de 2020 menée par Zeller et al. [59] étudie la possibilité de répondre à la question de savoir si un notebook est reproductible. Dans le cadre de cette étude, le développement de l'outils Osiris [8] a permis l'analyse post-mortem d'un notebook incluant une exploration de l'ensemble des ordres d'exécution possibles des cellules garantissant des résultats identiques, à condition que les résultats du notebook soient reproductibles. Dans le cas d'un notebook non reproductible, l'étude présente deux stratégies, la première qui consiste à définir une tolérance faible à la reproductibilité. Cette stratégie renvoie l'ensemble des chemins d'exécution avec des résultats similaires, permettant ainsi une certaine souplesse pour la reproductibilité d'un notebook. La deuxième stratégie est celle dite du meilleur effort. Elle implique la possibilité de transformer le code contenu dans une cellule, tout en garantissant le maintien d'une sémantique équivalente pour atteindre l'objectif de tolérance *faible*. Ces deux stratégies visent à offrir des solutions flexibles et efficaces pour traiter la reproductibilité des notebooks, même dans des situations où la reproductibilité des résultats n'est pas directement possible. L'analyse post-mortem est peu adaptée aux notebooks éducatifs puisque (par exemple) on ne souhaite pas retrouver des ordres d'exécution fantasques ou réécrire le code des cellules.

3.2 Les notebooks dans le contexte éducatif

3.2.1 Polyvalence pédagogique des notebooks

En France, c'est actuellement l'un des outils les plus exploités par les enseignants en informatique du secondaire en raison de sa facilité pédagogique dans divers contextes (cours, exercices, évaluation). Le projet Capytale (Section 2.2.1), par exemple, est un service web pour les enseignants du secondaire avec plus 500 000 utilisateurs déclarés.

8. https ://github.com/Osiris-Jupyter/Osiris

L'utilisation de la visualisation dynamique assistée par ordinateur comme moyen d'enseignement remonte aux années 1980, avec des exemples précurseurs dans les domaines de l'informatique et de l'éducation en sciences de nombreux exemples en technologie, ingénierie, mathématiques ou encore en sciences humaines [9] [56, 2, 17, 60]. Ces visualisations interactives (Figure 3.4) demeurent extrêmement populaires dans l'éducation en informatique de nos jours, car elles offrent une méthode tangible et visuelle pour représenter des concepts par ailleurs abstraits [4, 15]. Une étude mené par Nwulu et al. [41] auprès d'élèves ingénieurs montre que même s'il reste des améliorations à apporter aux notebooks Jupyter, sa polyvalence semble être particulièrement appréciée.

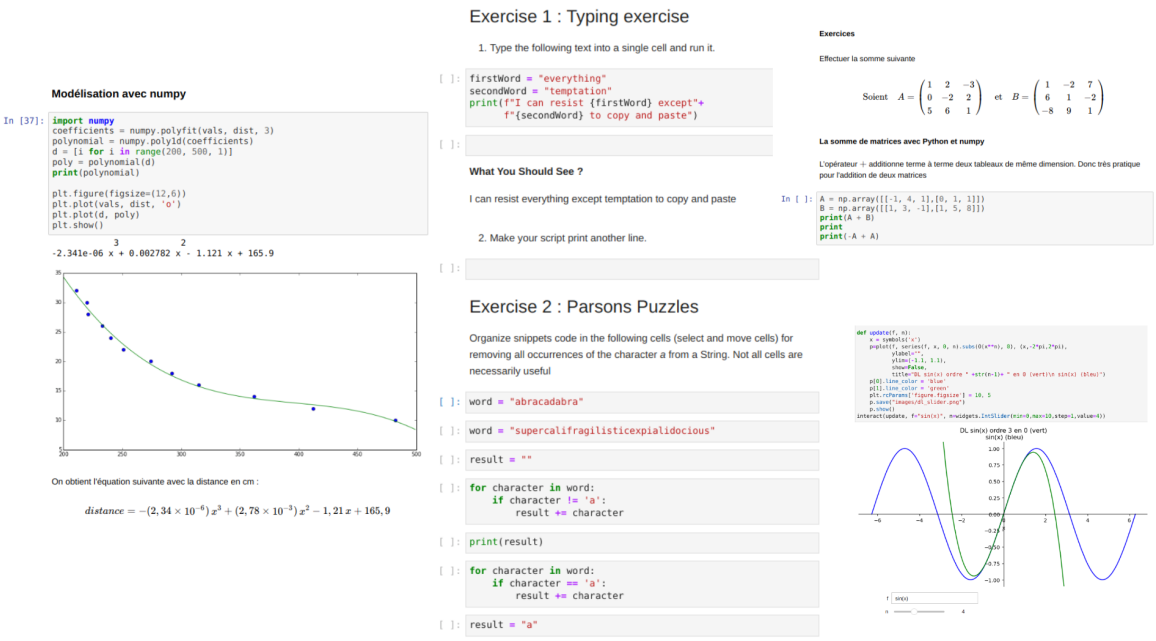


FIGURE 3.4 – Un ensemble d'activités pour la physique, l'informatique et les mathématiques.

La taxonomie de Bloom [34], est un modèle éducatif qui classifie les niveaux de compétence cognitive que les apprenants peuvent atteindre. Cette taxonomie, souvent représentée sous forme d'escalier, comprend six niveaux, du plus simple au plus complexe :

- Se souvenir (Remember) : La capacité de se rappeler des faits, des termes ou des concepts.
- Comprendre (Understand) : La capacité de comprendre le sens des informations et d'expliquer les idées avec ses propres mots.
- Appliquer (Apply) : L'aptitude à utiliser des connaissances ou des compétences dans des situations concrètes.
- Analyser (Analyze) : La capacité à décomposer des informations complexes en parties plus simples et à comprendre les relations entre ces parties.
- Évaluer (Evaluate) : L'aptitude à juger ou évaluer des informations en fonction de critères prédéfinis.
- Créer (Create) : La capacité à synthétiser des idées pour créer quelque chose de nouveau, original, ou à formuler des jugements basés sur des critères établis.

9. https ://programminghistorian.org/fr/lecons/introduction-aux-carnets-jupyter-notebooks

Cette taxonomie fournit une structure hiérarchique pour définir des objectifs d'apprentissage spécifiques et guider le développement de matériel pédagogique et d'activités d'évaluation. Les notebooks Jupyter s'avèrent être des outils polyvalents pour la mise en œuvre de la taxonomie de Bloom, offrant des avantages à chaque niveau.

Ils viennent soutenir des pédagogies actives permettant de mener les étudiants au sommet de l'escalier, telles que l'apprentissage fondé sur l'enquête (AFE) [1] et l'apprentissage par problèmes (APP) [11]. Les pédagogies actives placent les étudiants face à leurs questionnements, ou face à des problèmes réels avec des scénarios plus ou moins complexes à résoudre. Au lieu d'une transmission linéaire de connaissances, ces méthodes engagent les apprenants dans un processus actif d'exploration, d'analyse, d'évaluation et de résolution de problèmes [7]. Le notebook offre un environnement interactif, documenté et polyvalent qui s'aligne parfaitement avec les principes de ces approches pédagogiques. Il favorise l'expérimentation, la documentation claire, la collaboration et l'apprentissage pratique des étudiants. Pour évaluer le développement des logiciels issus de notre recherche sur l'assistance à l'utilisation de notebooks dans un contexte éducatif, nous avons employé deux notebooks axés sur des approches d'apprentissage par problèmes, ce qui implique des phases exploratoires intenses.

3.2.2 Utilisation des notebooks en informatique

Il a été mis en évidence que les documents interactifs de type notebooks ont un impact non négligeable sur la performance des étudiants en informatique. Dans une étude menée par Edgcomb et al. [21] avec 1 945 étudiants répartis sur 4 cours d'initiation à la programmation (C/C++) dans 3 universités différentes le remplacement des documents statiques (Figure 3.5) par un document interactif (zybooks [10]), avec la plupart des autres caractéristiques du cours restant inchangées, a entraîné d'importantes améliorations dans les notes aux examens, projets et la note globale, avec une signification statistique extrêmement forte. Une autre étude de Smith et al. [26] a elle mesurée l'engagement et l'impact sur l'apprentissage de l'utilisation des notebooks Jupyter dans un cours universitaire d'initiation à l'informatique avec 80 étudiants. Les notebook contenaient des tests statiques, des images et du code interactif avec des visualisations dynamiques et des images générées à partir de code exécutable. Les résultats de cette étude montre un engagement plus fort des étudiants dans les situations d'apprentissage avec l'utilisation des notebooks. Cependant il ne faut pas oublier que les notebooks ont été conçus dans le cadre plus spécifique de l'analyse de données, de la documentation et du partage de cette analyse documentée. L'adaptation des notebooks au domaine éducatif et en particulier dans l'enseignement de l'informatique pose encore de nombreux problèmes. Plusieurs études [26, 9, 10, 32] soulignent la difficulté pour les étudiants d'exécuter un notebook éducatif. Parmi ces difficultés, certaines ne sont pas uniquement liées à la discipline. Par exemple, les évolutions rapides des outils numériques ont amené des chercheurs en psychologie à s'interroger sur la qualité de la lecture numérique. Une méta-analyse menée par Delgado et al. [18] des articles parus entre 2000 et 2017 sur les impacts de la lecture numérique, impliquant 171 055 personnes indique un avantage certain pour une meilleur compréhension des textes avec la lecture papier. Plus surprenant encore, les nouvelles générations qui grandissent entourées de technologies numériques, ne sont pas épargnées par ces conclusions. Des études complémentaires [62, 36] fournissent des preuves que les gens adoptent un style de traitement plus superficiel dans les environnements numériques. Par exemple, la nécessité de faire défiler a été identifiée comme un obstacle

10. https ://www.zybooks.com/

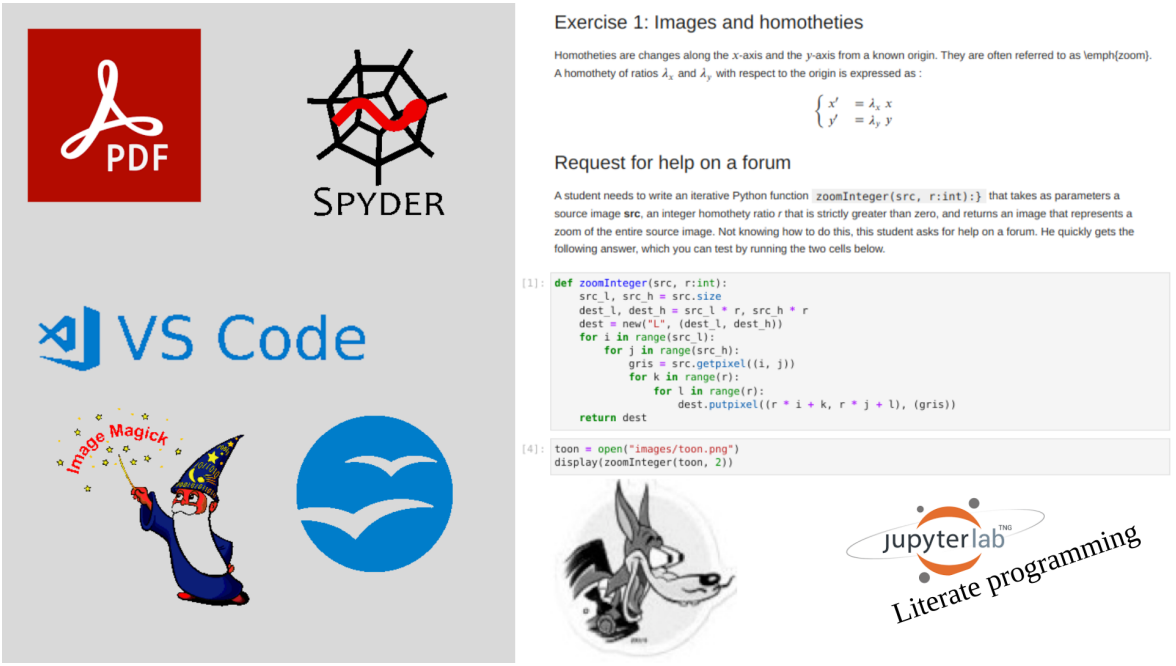


FIGURE 3.5 – Le notebook est un document dynamique et unique qui évite aux débutants d'avoir à gérer plusieurs applications produisant des documents statiques

potentiel à la compréhension lors de la lecture numérique [44]. Le défilement ajoute une charge cognitive à la tâche de lecture en rendant l'orientation spatiale vers le texte plus difficile pour les lecteurs que l'apprentissage à partir d'un texte imprimé.

3.3 Synthèse

Bien que les utilisations des data scientists diffèrent considérablement de celles des étudiants débutants en informatique, la cohérence et la reproductibilité des notebooks restent des problèmes communs. Dans cet état de l'art il a été mis en évidence que de nombreuses erreurs ou résultats incorrects peuvent survenir à l'exécution du notebook dont une part non négligeable est dûe à l'ordre d'exécution des cellules de code. De nombreuses études ont également montré l'importance d'apporter à l'utilisateur une assistance à l'exécution des notebooks en fonctions des besoins de chacun.

Les candidats au défi n°1

Pour satisfaire le défi n°1 (2.4.1) nous avons examiné différents outils conçus pour aider à la reproductibilité. L'analyse post-mortem, comme celle fournie par le logiciel OSIRIS, s'avère peu adaptée aux notebooks éducatifs puisque l'enseignant ne souhaite pas retrouver des ordres d'exécution erratiques ou réécrire le code des cellules rédigé par les étudiants. Les outils d'assistance (NBSAFETY, VERDANT, HISTREE, etc) donnent des informations détaillées sur l'évolution des contenus d'un notebook permettant de faciliter le travail d'écriture, de réorganisation et de partage des notebooks dans le domaine de la science des données. Cependant, aucun de ces outils ne propose de mécanismes pour identifier les cellules non reproductibles, à l'exception de NBSAFETY, qui utilise de l'analyse statique. Il n'est n'est donc pas indépendant du langage de programmation. En définitive, nous n'avons pas trouvé d'outil pleinement adapté pour relever le défi n°1.

Le deuxième défi concerne le suivi du scénario contenu dans une activité sans sacrifier les phases exploratoires souvent nécessaires aux étudiants. On s'intéresse également dans ce défi à la faisabilité d'élargir les cas d'utilisation du notebook à des scénarios non linéaires.

Les candidats au défi n°2

La reproductibilité des résultats n'est pas prise en compte dans ce défi. Le logiciel OSIRIS est donc écarté de notre sélection. Les outils d'assistance cités dans le défi n°1 sont principalement axés sur la gestion des différentes versions d'un notebook pour aider les data-scientists à explorer des moments clés de l'évolution de leurs notebooks. L'utilisateur peut donc naviguer dans l'historique de ces versions. Cependant il n'existe pas d'outil capable de guider l'utilisateur en temps réel dans le scénario d'une activité embarqué dans un notebook. Il n'y a donc pas d'outils pour satisfaire pleinement le défi n°2 (2.4.2) dans un contexte éducatif.

Pour combler ces lacunes et relever les deux principaux défis de cette thèse, nous présentons deux approches pour l'environnement Jupyter : NORM et MOON. Le premier propose une assistance visuelle à la reproductibilité des résultats en temps réel. Le second se concentre sur la reproductibilité du scénario de l'activité embarquée dans le notebook, afin de guider les étudiants vers des états valides et l'espoir de réduire les problèmes d'états cachés (source courante d'erreurs d'exécution et de non reproductibilité) lors des phases exploratoires, qui semble être un problème récurrent pour les étudiants. Les deux défis seront bien abordés via des approches indépendantes du langage de programmation utilisé dans le notebook.

3.4 Références

[1] Richard C ALEXANDER et al. "Integrity, Confidentiality, and Equity: Using Inquiry-Based Labs to help students understand AI and Cybersecurity". In : *Journal of Cybersecurity Education, Research and Practice* 2024.1 (2023), p. 10.

[2] Ghada AMOUDI et Dina TBAISHAT. "Interactive notebooks for achieving learning outcomes in a graduate course: a pedagogical approach". In : *Education and Information Technologies* (2023), p. 1-36.

[3] Bente CD ANDA, Dag IK SJØBERG et Audris MOCKUS. "Variability and reproducibility in software engineering: A study of four companies that developed the same system". In : *IEEE Transactions on Software Engineering* 35.3 (2008), p. 407-429.

[4] Ronald BAECKER. "Sorting out sorting: A case study of software visualization for teaching computer science". In : *Software visualization: Programming as a multimedia experience* 1 (1998), p. 369-381.

[5] Monya BAKER. "1,500 scientists lift the lid on reproducibility". In : *Nature* 533.7604 (2016).

[6] Lorena A BARBA. "Terminologies for reproducible research". In : *arXiv:1802.03311* (2018).

[7] Lorena A BARBA et al. "Teaching and learning with Jupyter". In : *https://jupyter4edu.github. io/jupyter-edu-book* (2019), p. 1-77.

[8] Xavier BOUTHILLIER, César LAURENT et Pascal VINCENT. "Unreproducible research is reproducible". In : *International Conference on Machine Learning*. PMLR. 2019, p. 725-734.

[9] Christophe CASSEAU et al. "Immediate Feedback for Students to Solve Notebook Reproducibility Problems in the Classroom". In : *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2021, St Louis, MO, USA, October 10-13, 2021*. Sous la dir. de Kyle J. HARMS et al. IEEE, 2021, p. 1-5. DOI : 10.1109/VL/HCC51201.2021.9576363. URL : https://doi.org/10.1109/VL/HCC51201.2021.9576363.

[10] Christophe CASSEAU et al. "MOON: Assisting Students in Completing Educational Notebook Scenarios". In : *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2023, p. 157-167.

[11] Dhruva K CHAKRAVORTY et al. "Evaluating active learning approaches for teaching intermediate programming at an early undergraduate level". In : *The Journal of Computational Science Education* 10.1 (2019).

[12] Souti CHATTOPADHYAY et al. "What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities". In : *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*. Sous la dir. de Regina BERNHAUPT et al. ACM, 2020, p. 1-12. DOI : 10.1145/3313831.3376729. URL : https://doi.org/10.1145/3313831.3376729.

[13] Jürgen CITO et Harald C. GALL. "Using Docker Containers to Improve Reproducibility in Software Engineering Research". In : *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 2016, p. 906-907.

[14] Jon CLAERBOUT. "Active documents and reproducible results". In : *Stanford Exploration Project Report* 67 (1990), p. 139-144.

[15] Sutrini CLAUDIO, Davide PASSARO, Pallotta FILIPPO et al. "The potential of using jupyter notebook in physics education: Experimentation for high school students". In : *Il nuovo cimento C* (2022), p. 1-4.

[16] Ludovic COURTÈS. "Reproduire les environnements logiciels: un maillon incontournable de la recherche reproductible". In : *1024: Bulletin de la Société Informatique de France*. 18. 2021, p. 15-22.

[17] Alessio DE SANTO et al. "Promoting Computational Thinking Skills in Non-Computer-Science Students: Gamifying Computational Notebooks to Increase Student Engagement". In : *IEEE Transactions on Learning Technologies* 15.3 (2022), p. 392-405.

[18] Pablo DELGADO et al. "Don't throw away your printed books: A meta-analysis on the effects of reading media on reading comprehension". In : *Educational Research Review* 25 (2018), p. 23-38. ISSN : 1747-938X. DOI : 10.1016/j.edurev.2018.09.003. URL : https://doi.org/10.1016/j.edurev.2018.09.003.

[19] Loic DESQUILBET et al. *Vers une recherche reproductible*. Sous la dir. d'Unité régionale de formation À L'INFORMATION SCIENTIFIQUE ET TECHNIQUE DE BORDEAUX. Unité régionale de formation à l'information scientifique et technique de Bordeaux, mai 2019, p. 1-161. URL : https://hal.science/hal-02144142.

[20] Klaus ECKELT et al. "Loops: Leveraging Provenance and Visualization to Support Exploratory Data Analysis in Notebooks". In : *COMPUTER GRAPHICS forum*. T. 43. 3. 2024.

[21] Alex Daniel EDGCOMB et al. "Student performance improvement using interactive textbooks: A three-university cross-semester analysis". In : *2015 ASEE Annual Conference & Exposition*. 2015, p. 26-1423.

[22] Steven GOODMAN, Daniele FANELLI et John IOANNIDIS. "What does research reproducibility mean?" In : *Science Translational Medicine* 8 (juin 2016), 341ps12-341ps12. DOI : 10.1126/scitranslmed.aaf5027.

[23] Björn GRÜNING et al. "Practical Computational Reproducibility in the Life Sciences". In : *Cell Systems* 6.6 (2018), p. 631-635. ISSN : 2405-4712. DOI : https://doi.org/10.1016/j.cels.2018.03.014. URL : https://www.sciencedirect.com/science/article/pii/S2405471218301406.

[24] Odd Erik GUNDERSEN et Sigbjørn KJENSMO. "State of the art: Reproducibility in artificial intelligence". In : *Proceedings of the AAAI Conference on Artificial Intelligence*. T. 32. 1. 2018.

[25] Andrew HEAD et al. "Managing Messes in Computational Notebooks". In : *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*. Sous la dir. de Stephen A. BREWSTER et al. ACM, 2019, p. 270. DOI : 10.1145/3290605.3300500. URL : https://doi.org/10.1145/3290605.3300500.

[26] David H. Smith IV et al. "Towards Modeling Student Engagement with Interactive Computing Textbooks: An Empirical Study". In : *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*. Sous la dir. de Mark SHERRIFF et al. ACM, 2021, p. 914-920. DOI : 10.1145/3408877.3432361. URL : https://doi.org/10.1145/3408877.3432361.

[27] Jeremiah W. JOHNSON. "Benefits and Pitfalls of Jupyter Notebooks in the Classroom". In : *SIGITE 20: The 21st Annual Conference on Information Technology Education, Virtual Event, USA, October 7-9, 2020*. Sous la dir. de Deepak KHAZANCHI et al. ACM, 2020, p. 32-37. DOI : 10.1145/3368308.3415397. URL : https://doi.org/10.1145/3368308.3415397.

[28] Mary Beth KERY et Brad A MYERS. "Interactions for untangling messy history in a computational notebook". In : *2018 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE. 2018, p. 147-155.

[29] Mary Beth KERY et al. "The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool". In : *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. Sous la dir. de Regan L. MANDRYK et al. ACM, 2018, p. 174. DOI : 10.1145/3173574.3173748. URL : https://doi.org/10.1145/3173574.3173748.

[30] Mary Beth KERY et al. "Towards effective foraging by data scientists to find past analysis choices". In : *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, p. 1-13.

[31] Moaiad Ahmad KHDER, Samah Wael FUJO et Mohammad Adnan SAYFI. "A roadmap to data science: background, future, and trends". In : *International Journal of Intelligent Information and Database Systems* 14.3 (2021), p. 277-293.

[32] Jacob KOEHLER et Soomi KIM. "Interactive Classrooms with Jupyter and Python". In : *The Mathematics Teacher* 111 (jan. 2018), p. 304. DOI : 10.5951/mathteacher.111.4.0304.

[33] David KOOP et Jay PATEL. "Dataflow notebooks: encoding and tracking dependencies of cells". In : *9th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2017)*. 2017.

[34] David R KRATHWOHL. "A revision of Bloom's taxonomy: An overview". In : *Theory into practice* 41.4 (2002), p. 212-218.

[35] Chris LAMB et Stefano ZACCHIROLI. "Reproducible builds: Increasing the integrity of software supply chains". In : *IEEE Software* 39.2 (2021), p. 62-70.

[36] Tirza LAUTERMAN et Rakefet ACKERMAN. "Overcoming screen inferiority in learning and calibration". In : *Computers in Human Behavior* 35 (2014), p. 455-463.

[37] Stephen MACKE et al. "Fine-Grained Lineage for Safer Notebook Interactions. CoRR abs/2012.06981 (2020)". In : *arXiv preprint arXiv:2012.06981* (2020).

[38] Robert C MARTIN. *The clean coder: a code of conduct for professional programmers*. Pearson Education, 2011.

[39] José MONTERO AMENEDO. "GOOD PRACTICES, MISSED OPPORTUNITIES AND THE USE OF JUPYTER NOTEBOOKS FOR INQUIRY-BASED LEARNING". In : (2023).

[40] "N. Y. University. (2017) Making Jupyter Notebooks Reproducible with ReproZip. [Online]. Available: https://docs.reprozip.org/en/1.0.x/jupyter.html". In :

[41] Nnamdi I NWULU, Uyikumhe DAMISA et Saheed Lekan GBADAMOSI. "Students Perception about the Use of Jupyter Notebook in Power Systems Education." In : *Int. J. Eng. Pedagog.* 11.1 (2021), p. 78-86.

[42] Prasad PATIL, Roger PENG et Jeffrey LEEK. *A statistical definition for reproducibility and replicability*. Juil. 2016. DOI : 10.1101/066803.

[43] João Felipe PIMENTEL et al. "A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks". In : *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, p. 507-517. DOI : 10.1109/MSR.2019.00077.

[44] Mary POMMERICH. "Developing computerized versions of paper-and-pencil tests: Mode effects for passage-based tests". In : *The Journal of Technology, Learning and Assessment* 2.6 (2004).

[45] Luigi QUARANTA, Fabio CALEFATO et Filippo LANUBILE. "Eliciting Best Practices for Collaboration with Computational Notebooks". In : *Proc. ACM Hum. Comput. Interact.* 6.CSCW1 (2022), 87:1-87:41. DOI : 10.1145/3512934. URL : https://doi.org/10.1145/3512934.

[46] Luigi QUARANTA, Fabio CALEFATO et Filippo LANUBILE. "Pynblint: a static analyzer for Python Jupyter notebooks". In : *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*. 2022, p. 48-49.

[47] Dhivyabharathi RAMASAMY et al. "Visualising data science workflows to support third-party notebook comprehension: an empirical study". In : *Empirical Software Engineering* 28.3 (2023), p. 58.

[48] Gema RODRIGUEZ-PÉREZ, Gregorio ROBLES et Jesús M GONZÁLEZ-BARAHONA. "Reproducibility and credibility in empirical software engineering: A case study based on a systematic literature review of the use of the szz algorithm". In : *Information and Software Technology* 99 (2018), p. 164-176.

[49] Adam RULE, Aurélien TABARD et James D. HOLLAN. "Exploration and Explanation in Computational Notebooks". In : *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. Sous la dir. de Regan L. MANDRYK et al. ACM, 2018, p. 32. DOI : 10.1145/3173574.3173606. URL : https://doi.org/10.1145/3173574.3173606.

[50] Adam RULE et al. "Aiding Collaborative Reuse of Computational Notebooks with Annotated Cell Folding". In : *Proc. ACM Hum.-Comput. Interact.* 2.CSCW (nov. 2018). DOI : 10.1145/3274419. URL : https://doi.org/10.1145/3274419.

[51] Adam RULE et al. *Ten Simple Rules for Reproducible Research in Jupyter Notebooks*. 2018. arXiv : 1810.08055 [cs.OH].

[52] Jeffrey S SALTZ et Ivan SHAMSHURIN. "Exploring the process of doing data science via an ethnographic study of a media advertising company". In : *2015 IEEE international conference on big data (Big Data)*. IEEE. 2015, p. 2098-2105.

[53] Sheeba SAMUEL et Birgitta KÖNIG-RIES. "ProvBook: Provenance-based Semantic Enrichment of Interactive Notebooks for Reproducibility." In : *ISWC*. 2018.

[54] Taijara Loiola de SANTANA et al. "Bug Analysis in Jupyter Notebook Projects: An Empirical Study". In : *arXiv preprint arXiv:2210.06893* (2022).

[55] Laurens STUDTMANN, Selin AYDIN et Horst LICHTER. "HISTREE: A Tree-Based Experiment History Tracking Tool for Jupyter Notebooks". In : ().

[56] Oguzhan TOPSAKAL. "Teaching Algorithms Design Approaches via Interactive Jupyter Notebooks". In : *European Journal of Technique (EJT)* 13.1 (), p. 1-6.

[57] Michele TUFANO et al. "There and back again: Can you compile that snapshot?" In : *Journal of Software: Evolution and Process* 29.4 (2017), e1838.

[58] Julia WAGEMANN et al. "Five guiding principles to make jupyter notebooks fit for earth observation data education". In : *Remote Sensing* 14.14 (2022), p. 3359.

[59] Jiawei WANG et al. "Assessing and Restoring Reproducibility of Jupyter Notebooks". In : *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2020, p. 138-149.

[60] Charles J. WEISS. "A Creative Commons Textbook for Teaching Scientific Computing to Chemistry Students with Python and Jupyter Notebooks". In : *Journal of Chemical Education* 98.2 (2021), p. 489-494. DOI : 10.1021/acs.jchemed.0c01071. URL : https://doi.org/10.1021/acs.jchemed.0c01071.

[61] Stefan WINTER et al. "A retrospective study of one decade of artifact evaluations". In : *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2022. , Singapore, Singapore, Association for Computing Machinery, 2022, p. 145-156. ISBN : 9781450394130. DOI : 10.1145/3540250.3549172. URL : https://doi.org/10.1145/3540250.3549172.

[62] Maryanne WOLF, Mirit BARZILLAI et John DUNNE. "The importance of deep reading". In : *Challenging the whole child: reflections on best practices in learning, teaching, and leadership* 130 (2009), p. 21.

Chapitre 4

NORM : Améliorer la Reproductibilité des Notebooks

« Reproduire un résultat scientifique : plus facile à dire qu'à faire »

Alexandre Hocquet

Sommaire

Dans ce chapitre nous aborderons le problème de reproductibilité des résultats lorsque les étudiants restituent leur notebook à l'enseignant pour être évalués. Nous nous intéressons uniquement aux notebooks qui s'exécutent linéairement de haut en bas.

Nous commencerons par décrire la notion de reproductibilité du point de vue de l'étudiant, qui exploite la nature non linéaire des notebooks pour écrire et exécuter les cellules de code dans un ordre arbitraire. En revanche, les enseignants ne sont pas conscients de cet ordre d'exécution implicite et s'attendent à reproduire les résultats en exécutant les cellules de manière linéaire, de haut en bas. Ces deux modes d'utilisation entrent en conflit, ce qui rend difficile l'évaluation du travail des étudiants par les enseignants.

Nous présenterons ensuite une nouvelle approche mise en œuvre sous la forme d'un plugin Jupyter appelé NORM (Notebook Reproducibility Monitor). Ce plugin fournit un retour visuel immédiat aux étudiants sur la reproductibilité de leurs notebooks directement dans Jupyter. Il avertit l'étudiant dès que la sortie d'une cellule n'est plus reproductible lorsqu'elle est exécutée linéairement de haut en bas, offrant ainsi la possibilité de résoudre le problème dès qu'il est détecté.

Puis nous discuterons des avantages de notre approche durant une séance *d'informatique pour tous* (nom de la discipline enseignée) en Classes Préparatoires aux Grandes Écoles (CPGE) et nous montrerons que notre plugin améliore significativement la reproductibilité des notebooks sans sacrifier la productivité des étudiants.

4.1 Introduction

En France, les notebooks Jupyter sont désormais proposés par le Ministère de l'Éducation Nationale (à travers le projet CANDYCE [1]) comme environnement d'apprentissage virtuel et interactif pour les enseignants et les étudiants. Plus précisément, le contexte et les défis de ce chapitre découlent de notre expérience avec les cours d'informatique des étudiants en (CPGE). Les CPGE scientifiques sont des cursus post-secondaires hautement sélectifs et intensifs, composés de deux années d'études. Leur objectif principal est de former les étudiants à l'admission dans les écoles d'ingénieurs scientifiques les plus prestigieuses. Les algorithmes, la science des données, la modélisation et la simulation sont au cœur du cours d'informatique pour ces étudiants, et les notebooks Jupyter sont un environnement idéal pour enseigner ces sujets [11].

Notamment, les notebooks éducatifs permettent aux étudiants de travailler sur des activités d'apprentissage de programmation en Python, le langage de choix dans les CPGE. Pour répondre aux questions de l'activité, les étudiants écrivent du code source dans des cellules de code. L'exécution d'une cellule de code produit une sortie optionnelle (le résultat de l'évaluation du code dans la cellule) affichée dans une zone dédiée en dessous de la cellule. Les étudiants utilisent ensuite ces sorties pour vérifier que leur code produit le résultat attendu. Durant la phase exploratoire de l'activité, les étudiants sont libres d'insérer, de modifier, de supprimer, d'exécuter et de réexécuter les cellules de code dans n'importe quel ordre. Une fois satisfaits des résultats, ils remettent le notebook à leur enseignant.

Pour évaluer leur travail, les enseignants vérifient d'abord que les sorties stockées dans les notebooks des étudiants contiennent les résultats corrects. Ensuite, n'ayant aucun moyen de connaître l'ordre d'exécution suivi par les étudiants, les enseignants réexécutent les cellules linéairement, de haut en bas, pour s'assurer d'obtenir les mêmes résultats [4]. Lorsque c'est le cas, l'enseignant peut alors se concentrer sur la qualité du

1. https ://candyce.org/ProgrammeCandyce.html

code pour noter le devoir.

Idéalement, parce que les cellules de code sont identiques, les sorties obtenues par les étudiants juste avant de remettre leur notebook et celles obtenues par l'enseignant lors de la réexécution du notebook devraient être les mêmes [4]. Dans un tel cas, le notebook est dit *reproductible*, un aspect primordial de la méthode scientifique [8]. Malheureusement, il est courant que les enseignants obtiennent des résultats différents lors de la réexécution des notebooks qu'ils reçoivent. Cela est généralement dû à des différences dans l'ordre d'exécution suivi par les étudiants et l'enseignant. Dans ce chapitre les activités des notebooks sont conçues pour être exécutées linéairement du haut vers le bas. Mais les étudiants bénéficient des avantages de l'exécution non linéaire pendant la phase exploratoire, tandis que les enseignants suivent une exécution linéaire pour la lecture et la notation. Des résultats différents dans les sorties ne signifient pas nécessairement que les étudiants n'ont pas fait correctement leur travail. Par conséquent, l'enseignant doit comprendre comment l'étudiant en est arrivé à cet état pour comprendre son raisonnement et noter de manière appropriée. Cette tâche est complexe et pourrait être évitée si le notebook était reproductible dès le départ.

Une solution naïve consisterait à forcer les étudiants à exécuter leur notebook de manière linéaire au moins une fois juste avant le rendu, afin de vérifier qu'une exécution linéaire produit les mêmes résultats. Cependant, cela exécuterait chaque cellule de code sur l'état actuel de la mémoire du notebook, qui n'est pas partagée avec l'enseignant. Rappelons que l'enseignant reçoit le notebook sous sa forme JSON (sérialisation de la vue de l'étudiant). Même si l'étudiant efface l'état de la mémoire avant d'exécuter le notebook pour se mettre dans les mêmes conditions que l'enseignant, il se heurterait alors au même problème d'essayer de comprendre comment le notebook en est arrivé à un état non reproductible.

Dans notre approche, nous proposons de résoudre ce problème en utilisant un plugin Jupyter qui fournit un retour visuel immédiat aux étudiants concernant la reproductibilité pendant la rédaction de leurs notebooks [3, 6, 1, 10]. Notre plugin tente continuellement de reproduire le notebook en arrière-plan et met en évidence les cellules qui sont reproductibles et celles qui ne le sont pas en utilisant des couleurs. Il permet aux étudiants de prendre conscience des problèmes de reproductibilité dès que possible afin de déterminer et d'agir sur les causes de ces problèmes par eux-mêmes ou avec l'aide de l'enseignant.

En tant que futurs scientifiques et ingénieurs, il est crucial de confronter les étudiants en CPGE aux problèmes de reproductibilité et de les sensibiliser au fait que c'est un point essentiel dans la méthode scientifique [7].

Pour évaluer l'utilité et l'impact de notre outil, nous avons réalisé une étude pilote contrôlée impliquant 37 étudiants en C.P.G.E sur une activité d'apprentissage en informatique de deux heures afin d'explorer les questions suivantes :

RQ1 Quelles sont les erreurs de reproductibilité les plus courantes dans les notebooks non reproductibles ?

RQ2 Notre plug-in améliore-t-il la reproductibilité des notebooks des étudiants ?

RQ3 La productivité des étudiants est-elle affectée par l'utilisation de NORM ?

Parmi les notebooks non reproductibles nous observons les mêmes problèmes d'exécution que ceux déjà identifiés dans de nombreux articles [4, 9, 2, 13, 12, 5]. Nos résultats montrent que les étudiants utilisant le plugin ont pu améliorer significativement la reproductibilité de leurs notebooks, avec seulement 1 (6%) notebook non reproductible, comparé à 47% de notebooks non reproductibles dans le groupe témoin. De plus, nous

constatons que les étudiants utilisant le plugin ont pu terminer l'activité proposée aussi bien que ceux du groupe témoin, suggérant que le plugin ne gêne pas leur travail. Le nombre de cellules produites et le nombre de bonnes réponses sont similaires dans les deux groupes. Parmi les notebooks non reproductibles dans le groupe témoin, nous avons identifié trois causes principales :

- exécution dans le désordre de cellules dépendantes,
- oubli d'exécution d'une cellule après modification du code contenu dans cette cellule,
- référence à une variable renommée ou supprimée.

Nous avons également mené un groupe de discussion pour recueillir l'avis des étudiants sur le plugin et les résultats obtenus sont très positifs.

4.2 Reproductibilité des notebooks

Dans cette section, nous commençons par donner une définition simple et une notation mathématique pour les notebooks. Ensuite, nous soulignons comment les problèmes de reproductibilité surviennent du point de vue de l'auteur (les étudiants) et du lecteur (l'enseignant) dans un contexte éducatif. Enfin, nous donnons une définition de la reproductibilité des notebooks qui permet de détecter les problèmes entre les lecteurs et les auteurs.

Pour étudier la reproductibilité, nous considérons qu'un notebook est une liste de cellules de code et de leurs sorties. En effet, les notebooks contiennent généralement des cellules de texte supplémentaires qui ne produisent aucune sortie et peuvent être ignorées en toute sécurité. À titre d'exemple concret, examinons le notebook présenté dans la figure 4.1c. Ce notebook présente l'algorithme de Syracuse ainsi que deux exemples d'utilisation dans les cellules 3 et 5. Les cellules 3 et 5 produisent toutes deux une sortie, affichée juste en dessous des cellules de code correspondantes. Un notebook est une liste ordonnée de cellules de code (notées c_i), chacune associée à une sortie (notée o_i). La sortie est une chaîne de caractères et peut être vide ($o_i = \emptyset$)

$$\mathrm{N} = [\,(c_i, o_i)\,]_{1 \le i \le n} \tag{4.1}$$

En utilisant cette notation le notebook de la figure peut-être représenté sous la forme suivante :

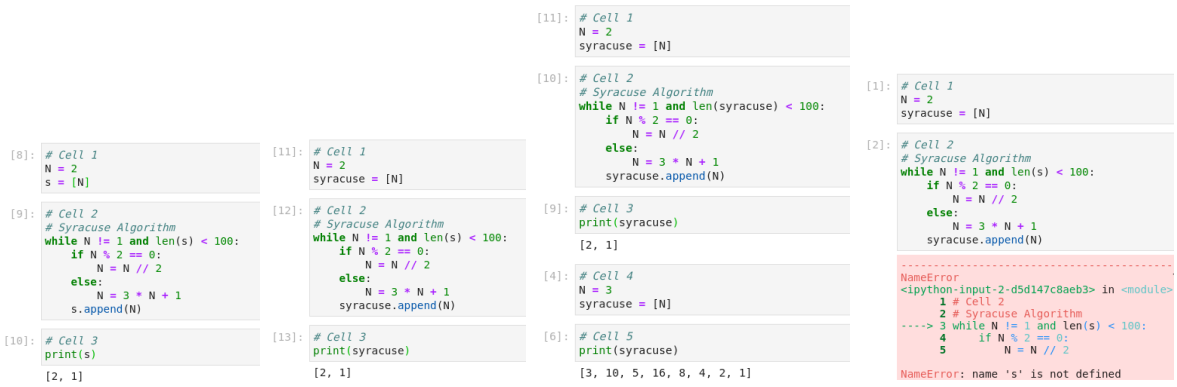
$$\mathrm{N}_{\mathrm{Student}} = [\,(c_1, \emptyset),\ (c_2, \emptyset),\ (c_3, \textit{"[2, 1]"}),\ (c_4, \emptyset),\ (c_5,\ \textit{"[3, 10, 5, 16, 8, 4, 2, 1]"})\,] \tag{4.2}$$

4.2.1 Du point de vue de l'étudiant

Lorsque qu'un étudiant exécute son notebook, il utilise inconsciemment un noyau spécifique à un langage de programmation pour son notebook, [2] en l'occurrence Python pour notre étude, pour exécuter les cellules de code, stocker l'état mémoire actuel, et écrire dans les cellules de sortie correspondantes (Figure 4.1).

Les étudiants utilisent principalement la commande *Exécuter la cellule* de Jupyter pour exécuter les cellules une par une dans un ordre arbitraire et non linéaire. Ce mode d'exécution est particulièrement adapté à la phase exploratoire, où les étudiants rédigent

2. https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html



(a) Première version (b) Deuxième version (c) Version finale (d) Éxécution de la deuxième version

FIGURE 4.1 – Versions successives d'un notebook étudiant pour l'implémentation de l'algorithme de Syracuse. La conjecture de Syracuse, encore appelée conjecture de Collatz ou problème $3x+1$, est l'hypothèse mathématique selon laquelle la suite de Syracuse de n'importe quel entier strictement positif atteint 1. Une suite de Syracuse est une suite d'entiers naturels définie de la manière suivante : on part d'un nombre entier strictement positif; s'il est pair, on le divise par 2; s'il est impair, on le multiplie par 3 et l'on ajoute 1. En répétant l'opération, on obtient une suite d'entiers strictement positifs dont chacun ne dépend que de son prédécesseur.

leurs notebooks de manière incrémentale en suivant une approche d'essais et d'erreurs. Il leur permet de revenir en arrière, de modifier et de réexécuter les cellules de code précédentes selon leurs besoins pour expérimenter avec leur code.

Dans la figure 4.1a, l'étudiante a écrit une première version de l'algorithme de Syracuse. Après avoir demandé des retours à son enseignant, l'étudiante modifie son code avec des noms de variables significatifs (`s` devient `syracuse`) mais oublie de mettre à jour une référence (Figure 4.1b, cellule 2, ligne 3). Cela ne pose aucun problème immédiat pour l'étudiante, car la variable `s` est toujours stockée en mémoire et l'algorithme produit toujours le résultat attendu lors de l'exécution complète et linéaire du notebook. Ensuite, dans la figure 4.1c, l'étudiante étend le notebook avec un autre exemple d'utilisation de l'algorithme dans les cellules 4 et 5. Cependant, le lecteur devra d'abord exécuter la cellule 4 pour initialiser les variables, puis la cellule 2 pour calculer les termes de la suite de Syracuse, et enfin la cellule 5 afin d'afficher les résultats.

Une fois l'activité terminée, l'étudiante partage son notebook avec son enseignant. Les cellules de code et les sorties associées sont sérialisées dans un fichier JSON, mais l'état mémoire correspondant est perdu.

4.2.2 Du point de vue de l'enseignant

Lorsque l'enseignant reçoit le notebook de l'étudiante, il n'est pas conscient de l'ordre d'exécution suivi par l'étudiante pour obtenir les sorties. Au lieu de cela, il s'attend à reproduire les mêmes résultats en exécutant les cellules de code du notebook linéairement du haut vers le bas. Cependant, ce faisant, le notebook de la Figure 4.1c produit une sortie différente pour la cellule 5. En effet, dans l'esprit de l'étudiante, la cellule 2 doit être réexécutée chaque fois que la variable `N` reçoit une nouvelle valeur pour calculer les nouveaux termes de la suite de Syracuse. Cet exemple met en évidence la différence entre les exécutions non linéaires et linéaires : le notebook s'exécute avec succès mais ne produit pas les mêmes sorties. *Le notebook n'est pas reproductible.*

Si l'enseignant avait reçu la deuxième version du notebook (Figure 4.1b), la situation serait pire. Tenter d'exécuter les cellules linéairement de haut en bas provoquerait une `NameError` Python sur la Cellule 2, comme le montre la Figure 4.1d. Dans ce cas, le notebook n'est même pas entièrement exécutable. Bien que cette version fonctionne avec succès pour l'étudiante, ce n'est pas le cas pour l'enseignant car leurs états mémoire diffèrent.

Dans les notebooks de la Figure 4.1, Jupyter affiche un *compteur d'exécution* à gauche de chaque cellule de code (`[9]` pour la cellule 3 dans la Figure 4.1c). Il indique le nombre total d'exécutions de cellules de code depuis le démarrage du noyau, au moment de l'exécution de cette cellule. Lorsque l'étudiante réexécute une cellule, elle écrase le compteur d'exécution qui était affiché précédemment. Par conséquent, la liste des compteurs d'exécution ne commence pas nécessairement à partir de 1 (si la première cellule exécutée a été réexécutée par la suite) et n'est pas nécessairement continue (comme c'est le cas dans la Figure 4.1c).

On pourrait penser qu'exécuter le notebook dans l'ordre donné par les *compteurs d'exécution* pourrait être une solution, mais malheureusement cette valeur ne donne pas forcément l'ordre d'exécution des cellules [9, 4]. Par exemple, exécuter le notebook de la figure 4.1c en suivant les compteurs d'exécution nous amènerait à exécuter les cellules dans l'ordre $[c_4, c_5, c_3, c_2, c_1]$ produisant également des sorties différentes. En outre, les compteurs d'exécution risquent de ne pas être visibles si l'étudiante n'a pas exécuté toutes les cellules avant de sauvegarder et de partager son notebook avec l'enseignant, ou si le notebook a été réinitialisé.

Lors de la notation, l'enseignant doit effectuer une analyse approfondie du code pour démêler les divers problèmes qui peuvent survenir. Cela implique de vérifier si les erreurs résultent de fautes de programmation, de malentendus des concepts, ou de dysfonctionnements liés à la non-reproductibilité du notebook. L'enseignant doit donc examiner méticuleusement les différentes cellules, les ordres d'exécution, et les résultats associés pour identifier la source des erreurs. Cette tâche peut s'avérer longue et exigeante, car elle nécessite une compréhension complète du code et de l'intention de l'étudiante.

4.2.3 Reproductibilité

Nous considérons qu'il existe principalement deux catégories de personne manipulant les notebooks : l'auteur c'est à dire le concepteur du notebook et le lecteur dont le rôle peut aller de la simple lecture à celui d'utilisateur devant exécuter les différentes cellules du notebook. Dans notre contexte l'auteur est l'étudiant qui va devoir implémenter des solutions et le lecteur est l'enseignant qui devra exécuter les notebooks des étudiants. Comme nous l'avons vu, il est naturel pour l'auteur de bénéficier de la liberté d'exécution non linéaire lors des phases exploratoires et il est convenu pour le lecteur d'exécuter et de lire le même notebook de manière linéaire, de haut en bas. Idéalement, le notebook devrait produire les mêmes sorties dans les deux cas.

Un notebook est *reproductible* si les sorties obtenues en exécutant chaque cellule de code de haut en bas sont les mêmes que celles enregistrées dans le notebook original. On peut évaluer la reproductibilité d'un notebook en comparant le *notebook de référence* (qui contient les sorties originales) avec le *notebook reproduit* (qui contient les sorties lorsqu'il est exécuté linéairement de haut en bas). En utilisant cette terminologie, nous définissons la reproductibilité d'un notebook comme suit.

Soit le *notebook de référence* noté

$$\mathrm{N}_{\mathrm{ref}} = [\,(c_i, o_i)\,]_{1 \leq i \leq n}$$

et le *notebook reproduit* noté

$$N_{rep} = [\,(c_i, o'_i)\,]_{1 \leq i \leq n}$$

Le notebook est reproductible si :

$$\forall (c_i, o_i) \in N_{ref}, (c_i, o'_i) \in N_{Rep} : o_i = o'_i \tag{4.3}$$

L'ensemble des cellules reproductibles dans un notebook de référence est donc défini comme suit :

$$R_N = \{c_i \mid (c_i, o_i) \in N_{ref}, (c_i, o'_i) \in N_{rep} : o_i = o'_i\} \tag{4.4}$$

Dans l'exemple de la figure 4.1c, le notebook $N_{Student}$ n'est pas reproductible mais contient les cellules reproductibles $R_{N_{Student}} = c_1, c_2, c_3, c_4$.

4.3 NORM : Une approche pratique à la reproductibilité

Notre hypothèse de travail est que les étudiants doivent être avertis le plus tôt possible d'éventuels problèmes de reproductibilité dans leurs notebooks pour ne pas avoir à déméler un état de la mémoire trop complexe.

Notre approche consiste à exécuter continuellement les notebooks de manière linéaire pendant leur rédaction pour vérifier si les sorties stockées dans le notebook peuvent être reproduites lors de l'exécution du même notebook de haut en bas. Si la sortie d'une cellule de code peut être reproduite avec succès, la cellule est colorée en vert. Si la sortie ne peut pas être reproduite, la cellule est colorée en rouge.

Dans cette section, nous présentons brièvement NORM un plugin pour l'environnement Jupyter qui fournit un retour immédiat et visuel sur la reproductibilité d'un notebook en cours de développement. NORM fonctionne de manière transparente et indépendamment du langage de programmation utilisé dans le notebook : chaque fois qu'un utilisateur exécute une cellule, le plug-in colore automatiquement chaque cellule du notebook avec la couleur appropriée (défi n°1 : 2.4.1).

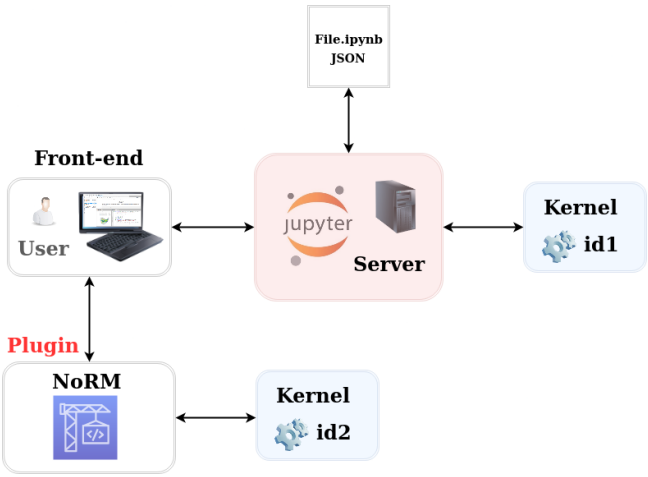
4.3.1 Architecture

Dans les applications du projet Jupyter, les utilisateurs interagissent la plupart du temps avec une interface web pour éditer et exécuter leurs notebooks (Figure 4.2a). Lorsqu'un utilisateur démarre un notebook, le serveur l'associe à un nouveau noyau *id1*, chargé d'exécuter le code.

Lorsque l'utilisateur active NORM, il crée et connecte un nouveau noyau *id2* au notebook. Ce noyau exécute toutes les cellules de haut en bas (*Run All*) à chaque fois qu'une cellule est exécutée. Les sorties produites par cette exécution sont comparées aux sorties affichées sur l'interface, récupérées à partir du noyau *id1*. Si les sorties produites par *id1* et *id2* pour la même cellule sont identiques, la cellule est considérée comme reproductible et NORM colore la partie gauche des cellules correspondantes (le prompt) en vert. Sinon, elle est colorée en rouge, comme illustré dans la figure 4.2b.

4.3.2 Comportement

Lorsque l'utilisateur exécute une cellule dans son notebook, le noyau *id1* exécute l'extrait de code de la cellule sur l'état mémoire actuel du notebook. Cet état dépend de la phase exploratoire de l'utilisateur et peut empêcher le notebook d'être reproductible.



(a) Les flèches représentent l'échange d'informations entre les différents composants

```
[4]: # Cell 4
     N = 3
     syracuse = [N]

[6]: # Cell 5
     print(syracuse)

     [3, 10, 5, 16, 8, 4, 2, 1]
```

(b) La cellule 5 est en rouge car elle n'est pas reproductible lors de l'exécution du notebook du haut vers le bas

FIGURE 4.2 – La figure présente l'utilisation de NORM par un étudiant et le retour sur l'interface utilisateur

Avec NORM (Figure 4.3), lorsque l'utilisateur exécute une cellule, un autre noyau *id2* est créé initialisant ainsi un nouvel état de la mémoire pour exécuter le notebook entièrement, de haut en bas, en arrière-plan. Si une autre exécution est déjà en cours, elle est arrêtée sans attendre les résultats, garantissant qu'il n'y a au plus qu'une seule exécution en arrière-plan à la fois. Le nouveau noyau exécute toutes les cellules de code de manière linéaire et stocke leurs sorties correspondantes. NORM récupère ensuite les sorties affichées sur l'interface (obtenues à partir de *id1*) et les compare à celles obtenues à partir du noyau *id2*. La comparaison est une simple comparaison exacte de chaînes de caractères. Enfin, pour chaque cellule, si les sorties produites par *id1* et *id2* sont identiques, le prompt de la cellule est coloré en vert ; sinon, il est coloré en rouge.

4.3.3 Limitations

La version actuelle de NORM compare après chaque exécution de cellule du notebook les sorties du notebook original et du notebook reproduit par un deuxième noyau en utilisant une simple comparaison de chaînes de caractères. Cette comparaison peut échouer en raison des causes suivantes [13] :

Code non déterministe Toute cellule dont la sortie dépend d'un code non déterministe (génération de nombres aléatoires, date actuelle, mélange aléatoire) peut produire une sortie différente à chaque exécution ;

Graphiques Certaines cellules de sortie affichant des graphiques peuvent produire un résultat différent en raison de différences dans l'environnement où le notebook est exécuté (navigateur et résolution de l'écran) ;

Nombres en virgule flottante Les cellules de code impliquant des calculs en virgule flottante peuvent produire des résultats légèrement différents entre les architectures 32 bits et 64 bits ;

Environnement d'exécution Les cellules de code qui invoquent des fonctions systèmes peuvent produire une sortie différente en fonction de l'environnement d'exécution (par exemple la fonction `os.listdir()` a cet effet). C'est un problème connu faisant l'objet de recherches en cours [12].

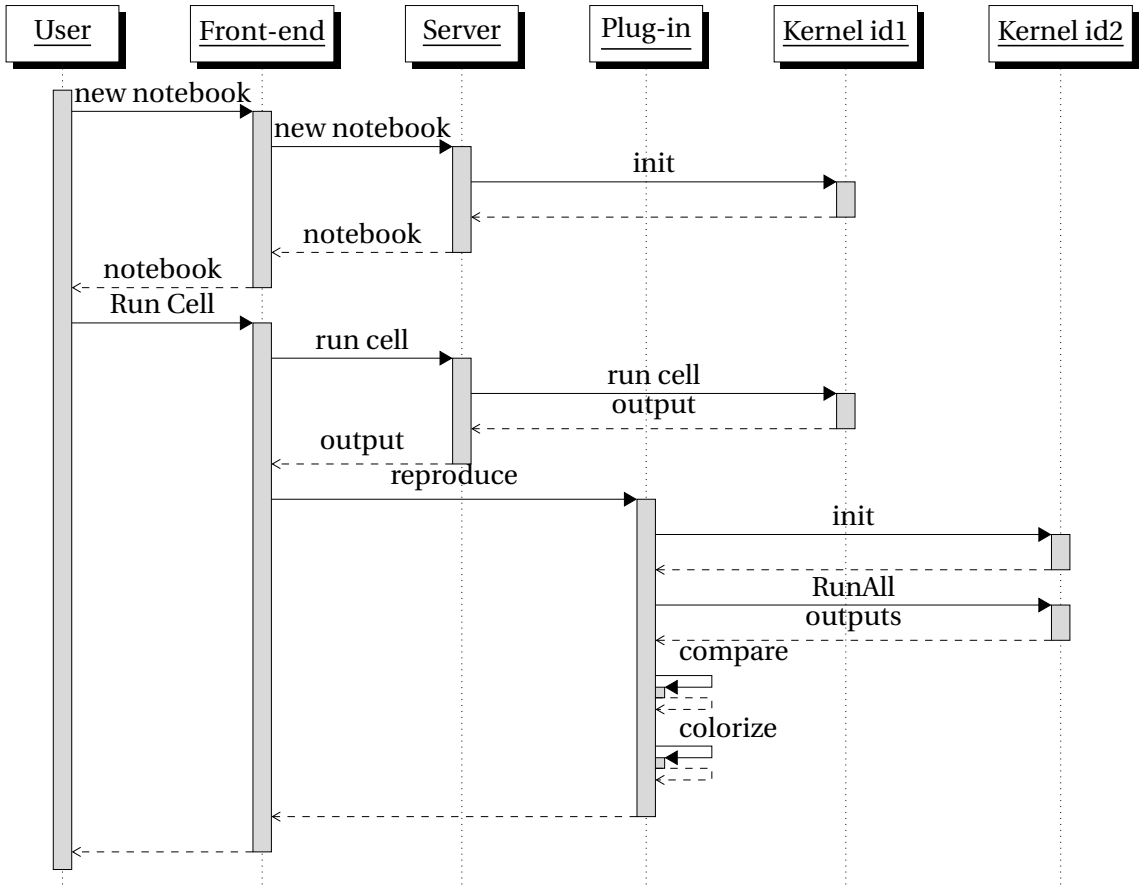


FIGURE 4.3 – Diagramme conceptuel de séquence : à chaque fois qu'une cellule de code est exécutée NORM évalue la reproductibilité des sorties du notebook.

Une autre limitation de NORM est qu'il ne gère pas bien les notebooks nécessitant des calculs intensifs et des temps d'exécution importants. Réexécuter les notebooks de manière linéaire après chaque exécution de cellule lorsque le notebook interagit avec des ensembles de données ou des bases de données volumineux n'est pas pratique.

Heureusement, ces limitations ne sont pas rédhibitoires dans un contexte éducatif, car l'environnement d'exécution est contrôlé (machines de laboratoire) ainsi que les constructions de programmation à utiliser. De plus, les activités des étudiants impliquent généralement de petits ensembles de données et un nombre limité de cellules de code.

4.4 Conception de l'étude

4.4.1 Participants

Nos participants sont 37 étudiants en CPGE (Classes Préparatoires aux Grandes Écoles), qui ont déjà été initiés à l'informatique au cours des trois dernières années du cycle secondaire. Ils ont appris à écrire des programmes simples (conditionnels, structures de contrôle, fonctions) avec le langage de programmation Python.

Les étudiants en CPGE sont une population idéale pour notre étude car ils sont destinés à devenir ingénieurs et scientifiques dans un large éventail de domaines (génie civil, chimie, etc.). En CPGE, le cours d'informatique enseigne des sujets de calcul scientifique, y compris l'utilisation de Python pour l'analyse de données et la simulation. Dans le cadre de leur formation, les étudiants doivent prendre conscience de l'importance de la

reproductibilité de leurs expériences et analyses. Les notebooks Jupyter sont bien adaptés pour répondre à ces exigences car ils sont populaires dans le contexte de l'analyse des données et préparent les étudiants à formater leur code et leurs analyses dans le même document [14].

4.4.2 Procédure

Les 37 étudiants en CPGE ont utilisé l'environnement Jupyter notebook pour le cours d'informatique pendant 3 mois avant notre étude. Juste avant le début de l'étude, l'enseignant (l'auteur de cette thèse) a pris une heure pour les initier à la notion de reproductibilité, et quinze minutes pour présenter l'utilisation de NORM.

Les étudiants ont été assignés à une séance de travaux pratiques de deux heures où ils devaient tous compléter les mêmes tâches. Les missions consistaient en cinq tâches de programmation consécutives avec un objectif commun : écrire un programme simple pour détecter la langue d'un texte. Au total, l'activité se composait de 13 questions et il était demandé aux étudiants d'utiliser autant de cellules que nécessaire pour y répondre. Le sujet contenait, pour chaque question, les résultats attendus pour certaines valeurs d'entrée données en exemple. Les étudiants étaient conscients que cette activité ne serait pas notée, mais qu'ils recevraient des commentaires sur leurs erreurs de programmation commises. Les étudiants devaient répondre aux questions tout en essayant de maintenir la reproductibilité du notebook de haut en bas.

Les 37 étudiants en CPGE ont été répartis en trois groupes au début de l'année. Pour l'étude, nous avons assigné de manière aléatoire l'ensemble des étudiants à deux groupes : un groupe *groupe témoin* et un *groupe expérimental* (qui a utilisé le plug-in NORM).

4.4.3 Mesures

Pour répondre aux questions de recherche RQ1, RQ2 et RQ3, nous avons utilisé les métriques suivantes :

Reproductibilité est une variable booléenne indiquant si un notebook est reproductible ou non lorsqu'il est exécuté de haut en bas.

Exécutabilité est une variable booléenne indiquant si un notebook est exécutable (ne génère pas d'erreur) ou non lorsqu'il est exécuté de haut en bas.

Nombre de cellules de code est un entier représentant le nombre total de cellules de code d'un notebook. Les cellules de texte ne sont pas comptées.

Nombre de réponses correctes est un entier représentant le nombre de questions correctement traitées par les étudiants dans un notebook. Les réponses correctes sont évaluées par un enseignant (l'auteur de cette thèse) en se basant uniquement sur la correction du code attendu.

4.4.4 Analyse

Pour la première question de recherche (RQ1), nous procédons à une analyse manuelle pour découvrir les causes de la non-reproductibilité. L'auteur de cette thèse a utilisé NORM sur les notebooks non reproductibles pour évaluer quelles cellules ne sont pas reproductibles. Ensuite, il a lu tout le code lié à ces cellules et a examiné le compteur d'exécution, essayant de comprendre la logique de cet état. Après avoir examiné tous les notebooks non reproductibles, deux des auteurs ont analysé les causes et les ont regroupées par similitude afin de créer une courte taxonomie.

Pour la question de recherche (RQ2), nous testons les deux hypothèses nulles suivantes : *NORM n'a aucun effet sur la reproductibilité des notebooks* et *NORM n'a aucun effet sur l'exécutabilité des notebooks*. Comme la reproductibilité et l'exécutabilité d'un notebook sont des variables booléennes et que notre taille d'échantillon est petite, nous utilisons le *test exact de Fisher* pour évaluer la significativité. Si le résultat est significatif, nous évaluons la taille de l'effet à l'aide du ratio de risque.

Pour la question de recherche (RQ3), nous testons les hypothèses nulles suivantes : *NORM n'a aucun effet sur le nombre de cellules des notebooks* et *NORM n'a aucun effet sur le nombre de bonnes réponses des étudiants*. Comme nous n'avons aucune hypothèse sur la distribution du nombre de cellules et de bonnes réponses, nous utilisons un test de Mann-Whitney pour évaluer la significativité. En cas de résultat significatif, nous évaluons la taille de l'effet à l'aide du delta de Cliff.

Pour les questions de recherche (RQ2 et RQ3), nous évaluons la significativité des p-values selon l'échelle suivante : * indique une p-value sous le seuil lâche de 0.1, ** une p-value sous le seuil classique de 0.05, *** une p-value sous le seuil strict de 0.01. Nous utilisons délibérément une échelle commençant à 0.1 en raison de la très petite taille de notre échantillon. Une p-value non étoilée est considérée comme non significative.

Enfin, en tant que moyen de trianguler nos résultats quantitatifs, nous procédons à un groupe de discussion avec les deux groupes de participants de notre étude.

4.5 Résultats

Dans cette section, nous décrivons les résultats obtenus pour nos trois questions de recherche. Les exemples, les devoirs, les notebooks, les données et les analyses sont disponibles sur cette page web [3] à consulter.

4.5.1 Les erreurs courantes qui freinent la reproductibilité

Comme expliqué dans la section 4.4.4, nous avons examiné tous les notebooks non reproductibles de notre étude afin de catégoriser les causes qui rendent un notebook non reproductible. Nous avons extrait trois catégories principales, décrites ci-dessous.

Exécution hors séquence de cellules dépendantes Dans cette catégorie, la non reproductibilité survient parce que l'auteur a exécuté les cellules dans un ordre autre que l'ordre de haut en bas. Par exemple, la Figure 4.4a, c_2 contient l'algorithme de Syracuse qui calcule la suite de Syracuse d'un entier N > 0. D'abord, N est initialisé dans c_1, puis c_2 calcule le résultat affiché dans c_3. Ensuite, N est modifié dans c_4, puis c_2 calcule le résultat affiché dans c_5. Dans une exécution de haut en bas, c_2 est exécuté une seule fois et ne peut pas être appelé pour différentes valeurs de N éparpillées dans plusieurs cellules. La sortie de c_5 est donc `[3]`.

Oubli d'exécution de cellule après une modification du code Dans cette catégorie, la non reproductibilité survient parce que l'auteur a modifié une cellule, mais a oublié d'exécuter la cellule elle-même ou une cellule dépendante. Par exemple, dans la Figure 4.4b, l'utilisateur a modifié c_1 pour changer la valeur de N de 2 à 3. Ensuite, il a exécuté la cellule, mais a oublié d'exécuter c_2 et c_3. Par conséquent, la sortie enregistrée est `[2, 1]` tandis que la sortie de l'exécution de haut en bas est `[3, 10, 5, 16, 8, 4, 2 , 1]`.

3. https://www.dropbox.com/s/e4e10v852dpyd92/archive.zip?dl=0

```
[11]: # cell 1
      N = 2
      syracuse = [N]

[16]: # cell 2
      while N !=1 and len(syracuse) < 100:
          if N % 2 == 0:
              N = N // 2
          else:
              N = 3 * N + 1
          syracuse.append(N)

[13]: # cell 3
      print(syracuse)

      [2, 1]

[14]: # cell 4
      N = 3
      syracuse = [N]

[17]: # cell 5
      print(syracuse)

      [3, 10, 5, 16, 8, 4, 2, 1]
```

(a) Exécution hors séquence de cellules dépendantes : les sorties ne sont pas les mêmes pour l'exécution $[c_1, c_2, c_3, c_4, c_2, c_5]$ et l'exécution de haut en bas.

```
[11]: # cell 1
      N = 3
      syracuse = [N]

[12]: # cell 2
      while N !=1 and len(syracuse) < 100:
          if N % 2 == 0:
              N = N // 2
          else:
              N = 3 * N + 1
          syracuse.append(N)

[10]: # cell 3
      print(syracuse)

      [2, 1]
```

(b) Oubli d'exécution de cellule après une modification du code : L'utilisateur change la valeur de N dans la première cellule, puis exécute les cellules 1 et 2 mais n'exécute pas la cellule 3.

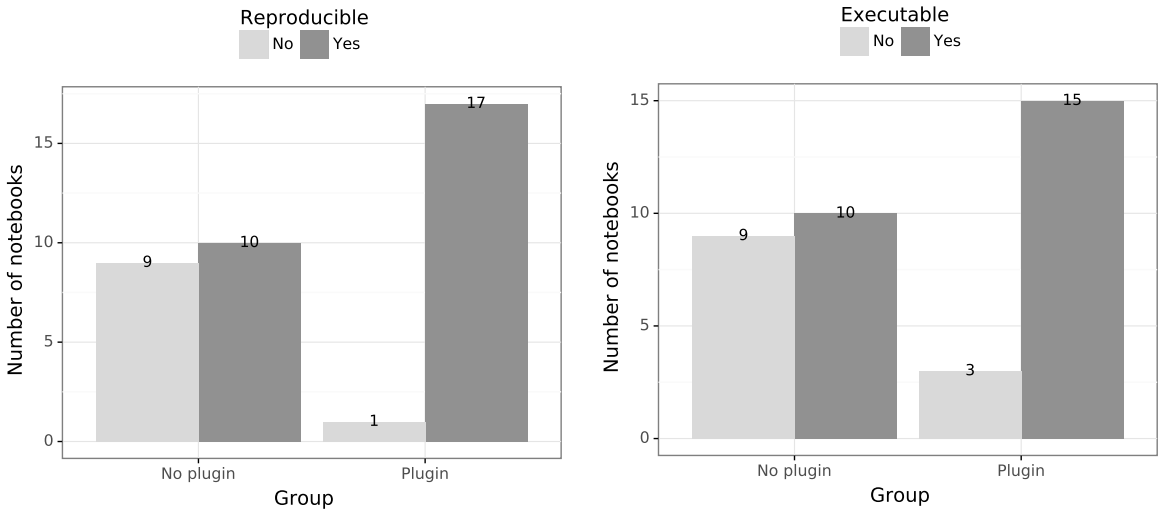
FIGURE 4.4 – Notebooks avec des cellules non reproductibles

Référence à une variable renommée ou supprimée Dans cette catégorie, la non reproductibilité survient parce que l'auteur modifie (ou supprime) le nom d'une fonction ou d'une variable et exécute la cellule correspondante. Cependant, l'ancien nom existe toujours dans la mémoire s'il n'a pas été explicitement réinitialisé. Dans ce cas, les cellules contenant du code faisant référence à l'ancien nom continueront à fonctionner, et l'auteur pourrait négliger le fait qu'il fait référence à un symbole obsolète. Cependant, une nouvelle exécution de haut en bas provoquera une erreur de nom (NameError).

4.5.2 Analyse comparative de la reproductibilité

Dans la Figure 4.5a, nous observons que 47% (9 sur 19) des notebooks du groupe témoin (sans notre plugin) ne sont pas reproductibles, tandis que seulement 6% (1 sur 18) des notebooks du groupe expérimental se trouvent dans cette situation. Dans l'étude de Pimentel et al. [9], 96% des notebooks avec un ordre d'exécution non ambigu n'étaient pas reproductibles, ce qui est plus élevé que dans notre groupe témoin. Dans notre étude, nous ne sommes pas touchés par certains problèmes de reproductibilité rencontrés dans la pratique, tels que des problèmes d'environnement incompatible ou de données manquantes. Cela pourrait expliquer pourquoi notre taux de notebooks non reproductibles est plus bas. Une autre raison pour expliquer le *faible* taux de notebooks non reproductibles est que nous avons explicitement informé les étudiants de rendre des notebooks reproductibles, alors que rien n'indique que la reproductibilité est importante dans les notebooks analysés par Pimentel et al.

Cependant, dans le groupe témoin, environ la moitié des notebooks ne sont toujours pas reproductibles. Dans le groupe expérimental, le taux de notebooks non reproductibles est considérablement réduit, avec seulement un notebook non reproductible sur les



(a) Nombre de notebooks non reproductibles et reproductibles dans les groupes témoin et expérimental.

(b) Nombre de notebooks exécutable et non exécutable dans les groupes témoin et expérimental.

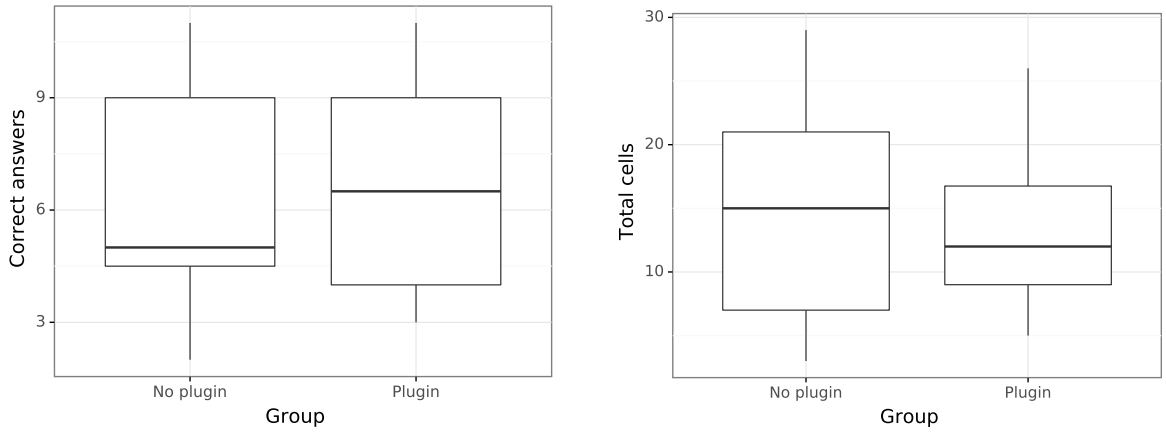
FIGURE 4.5 – Analyse des notebooks exécutables et reproductibles

18 notebooks rendus. Pour évaluer statistiquement la différence entre les deux groupes, nous réalisons un test exact de Fisher sur le tableau de contingence du nombre de notebooks reproductibles et non reproductibles dans les groupes témoin et expérimental. Le test donne une valeur p de 0.008 $**$, qui est significative même avec un seuil strict de 0.01. De plus, le rapport de risque est de 0.12, indiquant qu'un notebook développé avec le plugin a 88% moins de risques d'être non reproductible.

Dans la Figure 4.5b, nous observons que 47% (9 sur 19) des notebooks du groupe témoin (sans NORM) ne sont pas reproductibles, tandis que 17% (3 sur 18) des notebooks du groupe expérimental (avec NORM) se trouvent dans cette situation. Dans l'étude de Pimentel et al. [9], 75% des notebooks avec un ordre d'exécution non ambigu n'étaient pas exécutables, ce qui est également plus élevé que dans notre groupe témoin. De manière similaire à la reproductibilité, dans notre étude, nous ne sommes pas affectés par certains problèmes d'exécution rencontrés dans la pratique, tels que des dépendances manquantes ou des erreurs de chemin d'accès au disque. Dans le groupe expérimental, le taux de notebooks non exécutables est considérablement réduit, avec seulement trois notebooks non reproductibles. Pour évaluer statistiquement la différence entre les deux groupes, nous réalisons un test exact de Fisher sur le tableau de contingence du nombre de notebooks reproductibles et non reproductibles dans les groupes témoin et expérimental. Le test donne une valeur p de 0.078$*$, qui est significative avec un seuil de 0.1. De plus, le rapport de risque est de 0.35, indiquant qu'un notebook développé avec le plugin a 65% moins de risques d'être non exécutable.

Défi n°1

Nos résultats montrent que les étudiants utilisant NORM améliorent significativement la reproductibilité de leurs notebooks, avec seulement 6% de notebooks non reproductibles, comparé à 47% de notebooks non reproductibles dans le groupe témoin.



(a) Réponses correctes

(b) Nombre total de cellules de code

FIGURE 4.6 – Réponses correctes et nombre de cellules de code dans les groupes témoin et expérimental.

4.5.3 Effet de NORM sur la productivité des étudiants

Nous avons analysé les notebooks des étudiants pour identifier une possible influence du plugin sur le nombre de réponses correctes ou sur le nombre total de cellules. En effet, l'alerte de reproductibilité visuelle pourrait perturber le travail de l'étudiant, qui pourrait passer plus de temps à résoudre des problèmes de reproductibilité qu'à terminer les devoirs.

La Figure 4.6a montre la boîte à moustaches des distributions des réponses correctes dans les groupes témoin et expérimental. Nous remarquons que la médiane est légèrement plus élevée dans le groupe expérimental, et la plage interquartile est également plus grande dans ce groupe. Ces données sont légèrement en faveur du groupe expérimental. Pour évaluer statistiquement la différence entre les deux groupes, nous réalisons un test de Mann-Whitney non paramétrique (puisque nous n'avons aucune hypothèse sur la distribution du nombre de réponses correctes), qui donne une valeur p de 0.78. Avec une telle valeur p, nous ne pouvons pas rejeter l'hypothèse nulle selon laquelle il n'y a pas de différence entre les deux groupes.

La Figure 4.6b montre la boîte à moustaches des distributions du nombre de cellules dans les groupes témoin et expérimental. Nous remarquons que la médiane est légèrement plus basse dans le groupe expérimental, et la plage interquartile est également plus basse dans ce groupe. Ces données indiquent que le nombre de cellules semble être plus bas mais plus stable dans le groupe expérimental. Pour évaluer statistiquement la différence entre les deux groupes, nous réalisons un test de Mann-Whitney non paramétrique (puisque nous n'avons aucune hypothèse sur la distribution du nombre de cellules), qui donne une valeur p de 0.21. Avec une telle valeur p, nous ne pouvons pas rejeter l'hypothèse nulle selon laquelle il n'y a pas de différence entre les deux groupes. Par conséquent, NORM ne semble pas avoir d'impact négatif significatif sur la capacité des étudiants à compléter les exercices de leurs notebooks.

4.5.4 Groupe de discussion

Nous avons d'abord interrogé les étudiants sur la notion de reproductibilité et l'importance qu'ils y attachent. La majorité des étudiants pensent que c'est une notion importante lorsqu'il s'agit de partager leur travail. En tant que futurs ingénieurs, ils ont évo-

qué les protocoles d'expériences en physique et en chimie, mais aussi la démonstration d'un théorème en mathématiques et la notion de programmation en informatique.

Nous leur avons ensuite demandé s'ils pouvaient définir le lien entre la reproductibilité et le notebook. Nous avons reçu deux réponses spontanées, la première étant que lors du partage du notebook, l'exécution des différentes cellules du notebook doit donner le même résultat. La deuxième était de dire qu'il peut y avoir un lien entre les cellules et que la modification du code d'une cellule peut modifier la sortie d'une autre cellule. **Dans l'ensemble, les étudiants ont une bonne idée de la notion de reproductibilité et de sa relation avec un notebook.**

Dans une deuxième étape, nous avons demandé aux étudiants s'ils faisaient un effort pour prendre en compte la reproductibilité lors du développement d'un notebook et quelle stratégie ils utilisaient. Certains disent qu'ils font attention car s'ils enregistrent un notebook, ils veulent obtenir les mêmes résultats lorsqu'ils le rouvrent. D'autres disent qu'ils ont l'habitude d'utiliser des notebooks pour tester des extraits de code et n'ont donc pas l'habitude de se soucier de la reproductibilité lorsqu'ils remettent un devoir à l'enseignant. Dans l'ensemble, les étudiants conviennent qu'il est difficile d'être sûr que le notebook est reproductible. Ils n'ont vraiment pas de stratégie à cet effet, certains disent qu'ils font attention aux noms de variables, d'autres utilisent des fonctions, mais dans l'ensemble, **les étudiants disent qu'ils n'ont pas de bonne solution pour être sûrs qu'un notebook est reproductible**.

Enfin, nous avons demandé aux étudiants du groupe expérimental s'ils pouvaient donner un avantage et/ou un inconvénient de NORM. Tous les étudiants sont d'accord pour dire qu'il est facile d'identifier les cellules non reproductibles avec la couleur rouge et que la couleur verte est rassurante. Certains étudiants se plaignent que le plugin ne fournit aucune assistance pour rendre le notebook à nouveau reproductible.

4.5.5 Obstacles à la validité

Dans cette section, nous discutons des principaux obstacles à la validité affectant notre étude et nos résultats, suivant la classification de WOHLIN et al. [15].

Validité interne Tout d'abord, tous les étudiants n'avaient pas nécessairement le même niveau en programmation et les meilleurs étudiants ont peut-être fini dans le même groupe. Les groupes sont formés au début de l'année scolaire et il n'était pas possible de les changer. Une étude statistique des bonnes réponses suggère qu'il n'y a pas de différence significative.

Validité externe Deuxièmement, le problème donné aux étudiants a été spécialement conçu (avec des problèmes de reproductibilité possibles) pour évaluer le plugin dans un environnement contrôlé. Nous avons évité d'écrire un code non déterministe et d'utiliser des modules qui pourraient poser problème. Troisièmement, notre étude a une petite taille d'échantillon avec un nombre limité d'étudiants et elle a duré seulement deux heures. Le plugin semble efficace dans notre cas, mais nous ne savons pas s'il sera aussi efficace dans d'autres salles de classe et dans n'importe quel langage de programmation.

4.6 Conclusion

Dans ce chapitre, nous avons étudié la reproductibilité des notebooks dans le contexte des cours de programmation destinés à des étudiants futurs ingénieurs. Ils doivent com-

prendre que la reproductibilité est une notion essentielle. Un notebook est un objet complexe contenant des cellules de code pouvant être modifiées, ajoutées, supprimées, déplacées dans n'importe quel ordre. Cela entraîne des états successifs qui ne sont pas nécessairement exécutables de manière linéaire de haut en bas, de sorte qu'ils ne sont pas toujours reproductibles après avoir été enregistrés pour le partage ou la réutilisation. Les travaux antérieurs sur ce sujet détaillent les problèmes de reproductibilité et proposent des approches post-mortem pour détecter et restaurer les notebooks non reproductibles ou alors des approches basées sur de l'analyse statique c'est à dire uniquement valable pour un langage de programmation donné. En revanche, nous avons conçu et mis en œuvre une approche fournissant un retour visuel immédiat sur la reproductibilité des cellules d'un notebook, ainsi qu'une mise en œuvre d'un prototype : NORM. Les étudiants sont alertés le plus tôt possible d'une cellule non reproductible, les engageant à réfléchir et à résoudre le problème. Pour démontrer les avantages de notre approche, nous avons réalisé une étude contrôlée montrant que :

— Elle réduit significativement les problèmes de reproductibilité des notebooks des étudiants,

— Elle n'a pas d'effets indésirables significatifs sur la productivité des étudiants,

— Elle est bien perçue par les étudiants.

Lors du groupe de discussion, les étudiants ont soulevé un point important : le manque de diagnostic lorsqu'une cellule de notebook est dans un état non reproductible. Comme nous l'avons déjà expliqué, il peut être difficile de déterminer les causes d'une cellule qui n'est pas reproductible [12, 13].

4.7 Références

[1] Luciana BENOTTI et al. “The Effect of a Web-based Coding Tool with Automatic Feedback on Students’ Performance and Perceptions”. In : *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE ’18. New York, NY, USA : Association for Computing Machinery, fév. 2018, p. 2-7. ISBN : 978-1-4503-5103-4. DOI : 10.1145/3159450.3159579. URL : https://doi.org/10.1145/3159450.3159579 (visité le 07/05/2021).

[2] Souti CHATTOPADHYAY et al. “What’s Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities”. In : *CHI ’20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*. Sous la dir. de Regina BERNHAUPT et al. ACM, 2020, p. 1-12. DOI : 10.1145/3313831.3376729. URL : https://doi.org/10.1145/3313831.3376729.

[3] Sebastian GROSS et al. “Learning Feedback in Intelligent Tutoring Systems”. In : *KI - Künstliche Intelligenz* 29 (mai 2015). DOI : 10.1007/s13218-015-0367-y.

[4] David KOOP et Jay PATEL. “Dataflow notebooks: encoding and tracking dependencies of cells”. In : *9th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2017)*. 2017.

[5] Stephen MACKE et al. “Fine-Grained Lineage for Safer Notebook Interactions. CoRR abs/2012.06981 (2020)”. In : *arXiv preprint arXiv:2012.06981* (2020).

[6] Susanne NARCISS, Katja HUTH et Dr NARCISS. “How to design informative tutoring feedback for multi-media learning”. In : *Instructional Design for Multimedia Learning* (déc. 2002).

[7] Prasad PATIL, Roger PENG et Jeffrey LEEK. *A statistical definition for reproducibility and replicability*. Juil. 2016. DOI : 10.1101/066803.

[8] Roger PENG. “The reproducibility crisis in science: A statistical counterattack”. In : *Significance* 12.3 (2015), p. 30-32.

[9] João Felipe PIMENTEL et al. “A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks”. In : *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, p. 507-517. DOI : 10.1109/MSR.2019.00077.

[10] Felipe RESTREPO-CALLE, Jhon ECHEVERRY et Fabio GONZÁLEZ. “Using an interactive software tool for the formative and summative evaluation in a computer programming course: an experience report”. In : *Global Journal of Engineering Education* 22 (nov. 2020), p. 174-185.

[11] Adam RULE, Aurélien TABARD et James D. HOLLAN. “Exploration and Explanation in Computational Notebooks”. In : *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. Sous la dir. de Regan L. MANDRYK et al. ACM, 2018, p. 32. DOI : 10.1145/3173574.3173606. URL : https://doi.org/10.1145/3173574.3173606.

[12] Jiawei WANG, Li LI et Andreas ZELLER. “Restoring Execution Environments of Jupyter Notebooks”. en. In : *Proceedings of the 43rd International Conference on Software Engineering*. 2021, p. 12.

[13] Jiawei WANG et al. “Assessing and Restoring Reproducibility of Jupyter Notebooks”. In : *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2020, p. 138-149.

[14] Alistair WILLIS, Patricia CHARLTON et Tony HIRST. “Developing Students’ Written Communication Skills with Jupyter Notebooks”. In : *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE ’20. New York, NY, USA : Association for Computing Machinery, fév. 2020, p. 1089-1095. ISBN : 978-1-4503-6793-6. DOI : 10.1145/3328778.3366927. URL : https://doi.org/10.1145/3328778.3366927 (visité le 07/05/2021).

[15] Claes WOHLIN et al. *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA : Kluwer Academic Publishers, 2000. ISBN : 0-7923-8682-5.

Chapitre 5

MOON : Améliorer le suivi du scénario d'une activité dans un notebook

> « *Les scénarios sont des histoires qui parlent de l'avenir, mais leur but est de permettre la prise de meilleurs décisions dans le présent.* »
>
> Ged Davis

Sommaire

Dans ce chapitre nous verrons le lien entre la reproductibilité du scénario d'une activité et l'ordre d'exécution des cellules de code dans un notebook. À l'aide d'un cas d'étude nous illustrerons les problèmes engendrés (des erreurs d'exécution et/ou résultats incorrects) lorsque les étudiants ne suivent pas l'ordre d'exécution prévu par l'enseignant.

Puis, nous présentons une approche novatrice, MOON, conçue pour renforcer la reproductibilité du scénario d'une activité. L'idée centrale est de fournir aux enseignants un langage qui leur permet de formaliser l'utilisation attendue de leurs notebooks sous la forme d'un script et d'interpréter ce script pour guider les étudiants avec des indications visuelles en temps réel pendant leur interaction avec les notebooks.

Nous évaluons ensuite notre approche à l'aide d'une expérience contrôlée randomisée impliquant 21 étudiants, qui montre que MOON aide les étudiants à mieux respecter le scénario prévu sans entraver leur capacité à progresser. Nous terminons avec une enquête utilisateurs qui montre qu'environ 75% des étudiants interrogés ont perçu MOON comme plutôt utile ou très utile.

5.1 Introduction

Les notebooks offrent une opportunité unique de créer des activités d'apprentissage plus facile à prendre en main, car tout le matériel nécessaire est accessible aux étudiants sur une seule plateforme. Cependant, concevoir une activité pédagogique dans un notebook comporte des défis significatifs.

L'un des défis principaux réside dans la structure des notebooks, composés de cellules contenant du texte explicatif ou du code exécutable, avec une nature non linéaire qui permet aux apprenants de les exécuter dans n'importe quel ordre. Cette flexibilité peut conduire à des résultats inattendus pour les étudiants. En phase exploratoire, les étudiants peuvent involontairement créer des états cachés, souvent étudiés dans le contexte de l'analyse des données [3, 7, 9, 12], ce qui peut causer de nombreux problèmes dans un cadre éducatif.

Un autre défi tient à l'efficacité de la lecture sur les supports numériques, souvent inférieure à celle sur papier. De nombreuses études le démontrent [5, 11, 13, 4], suggérant que l'enseignant doit élaborer avec soin les instructions et le scénario de l'activité pour le notebook [6].

Ainsi, bien que les notebooks facilitent l'accès aux ressources éducatives, les enseignants doivent anticiper et gérer les risques liés à la non-linéarité et à la structure des activités pour garantir une expérience d'apprentissage efficace. S'appuyant sur cette idée, nous avons conçu MOON, une approche destinée à opérationnaliser et améliorer le suivi du scénario de l'activité contenue dans un notebook éducatif. Cette approche repose sur deux contributions principales.

- Tout d'abord, nous proposons un langage compilé avec un automate fini déterministe (DFA) qui permet aux enseignants d'exprimer à l'aide d'un script comment les étudiants sont censés manipuler les différentes cellules du notebook lié au scénario (au déroulement) de l'activité d'apprentissage.
- Ensuite, nous matérialisons le script avec un système de couleurs complété par des emojis qui assiste directement les étudiants à l'intérieur des notebooks (Figure 5.2). L'objectif de cette assistance est de permettre aux étudiants de maintenir un notebook cohérent avec le scénario de l'activité sans les empêcher de s'engager dans des activités exploratoires.

Pour évaluer l'utilité de notre approche, nous avons mené une étude contrôlée randomisée impliquant des étudiants en première année de licence d'informatique à l'université de Bordeaux. Durant cette séance nous leur proposons une activité de 2H40 sur la théorie des graphes. Nous observons que les étudiants du groupe témoin (sans MOON) complètent autant de cellules que ceux utilisant MOON. Cependant, les étudiants du groupe témoin éprouvent beaucoup plus de difficultés à suivre le déroulement de l'activité fournie par l'enseignant. Nous avons également réalisé une enquête auprès des étudiants qui confirme les résultats de notre expérience contrôlée et montre que la plupart des étudiants ont trouvé MOON utile. L'implémentation de MOON, les notebooks utilisés dans l'évaluation, ainsi que les données brutes de l'étude contrôlée et de l'étude utilisateur sont disponibles dans l'artefact Zenodo accompagnant [2].

5.2 Étude d'un notebook éducatif

5.2.1 Conception par l'enseignant

Lors de la conception d'une activité dans un notebook, l'enseignant est confronté à deux principaux défis :

- rédiger un texte qui transmet efficacement les connaissances,
- fournir des instructions claires pour que les étudiants exécutent les cellules de code en cohérence avec le scénario de l'activité.

Il est très fréquent qu'un étudiant passe par des phases exploratoires pour accomplir les différentes tâches proposées par l'activité. Ces phases exploratoires entraînent la plupart du temps de nombreuses modications et exécutions de cellules de code mais également des ajouts et suppressions de cellules. Les implications de ces phases exploratoires pour l'apprentissage ne doivent pas être négligées, car les étudiants peuvent être confrontés à toutes sortes d'erreurs durant l'exécution qui détournent leur attention des objectifs pédagogiques. Un cas fréquent est celui où les étudiants tentent d'utiliser des variables et des fonctions avant quelles ne soient définies ou bien au contraire utilisent un nom de variable encore présent dans l'espace global des noms du notebook mais dont la cellule servant à l'initialiser a été supprimée. Dans le premier cas l'étudiant obtient une erreur à l'exécution. Dans le deuxième cas l'exécution peut se faire correctement pour l'étudiant mais être une source d'erreur pour l'enseignant lors de la phase de rendu, entraînant ainsi des difficultés supplémentaires pour l'enseignant lors de l'évaluation pour retracer le raisonnement de l'étudiant.

Dans cette partie nous nous concentrons principalement sur l'utilisation de notebooks éducatifs interactifs en tant que feuille d'exercices. Prenons comme exemple un cours d'initiation au traitement d'images en Python où les étudiants sont invités à expérimenter avec `Pillow`, une bibliothèque d'imagerie, et à implémenter des algorithmes bien connus (Figure 5.1). L'enseignant décide d'organiser la session en trois parties. Remarquez que nous présentons un notebook simplifié par rapport à celui donné aux étudiants. Dans la première partie, les étudiants doivent importer la bibliothèque, charger une image d'exemple, calculer sa taille et le ratio de pixels noirs, puis réessayer avec une autre image. Dans la deuxième partie, les étudiants doivent mettre en œuvre un algorithme simple pour convertir une image en niveaux de gris. Ils sont ensuite invités à le tester sur une image qui leurs est proposée. Dans la troisième partie, les étudiants doivent mettre en œuvre un algorithme naïf de détection de contours et l'utiliser sur l'image précédente.

T0 Run the cell below to import the image functions from `PIL`.

C1 [ ]:
```
from PIL.Image import *
```

T2 Run the cell below to load the image.

C3 [ ]:
```
img = open("images/white_square_250x250.png")
```

T4 Run the cell below to compute and display the size of the image.

C5 [ ]:
```
width, height = img.size
print("width = ", width)
print("height =", height)
```

T6 Run the cell below and give a brief description of its purpose. Double click on this cell to append your answer.

Answer:

C7 [ ]:
```
greyscale = img.convert("L")
colors = greyscale.histogram()
black_pixels_ratio = colors[0] / (width * height) * 100
print("Black pixels ratio = ", black_pixels_ratio)
```

T8 Check your answer to the previous question with the image `black_square_100x100.png`.

T9 **Optional** Read the documentation of the histogram() function

C10 [ ]:
```
%pdoc greyscale.histogram
```

T11

Part A: Greyscale

Write a function `monochrome(img)` that takes a colored image as input and returns an image of the same size but in grayscale.

C12 [ ]:
```
def monochrome(img):
    # your code
    pass
```

T13 Test the `monochrome(img)` function in the cell below with the image *toon.png*

C14 [ ]:
```
# your code
```

T15

Part B: Naive approach to edge detection

Write a function `contoursV(img)` which takes an image as input and returns a new image of the same size with, for each pixel, the value of the difference in intensities between pixel (i, j) and pixel $(i-1, j)$.

C16 [ ]:
```
def contoursV(img):
    # your code
    pass
```

T17 Apply the `contoursV` function to the image *contoursV.png* then *forms.png*. **Note that these images are already monochrome.**

C18 [ ]:
```
# your code
```

FIGURE 5.1 – Exemple de notebook éducatif contenant une activité d'apprentissage. Les consignes sont rédigées dans des cellules Markdown. Toutes les cellules de code du notebook sont repérées avec la notation `Ci` et les cellules Markdown avec `Tj`. Les indices `i` et `j` représentent les numéros de cellules dans l'ordre d'apparition haut-bas.

Pour des raisons de lisibilité, chaque cellule est associée à un identifiant unique correspondant à sa position dans le notebook. La lettre `C` indique les cellules de code, tandis que la lettre `T` indique les cellules de texte. Comme le suggèrent les instructions de la pre-

mière partie, les étudiants sont invités à exécuter séquentiellement les cellules `C1`, `C3`, `C5` et `C7`, puis éventuellement à exécuter la cellule `C10`. Après l'exécution de `C7`, les étudiants devraient modifier `C3` pour charger une autre image et relancer les cellules intermédiaires `C5` et `C7` pour vérifier leurs réponses. Le reste du notebook est divisé en deux parties pouvant être traitées indépendamment.

5.2.2 Utilisation par les étudiants

Lorsque les étudiants ouvrent un notebook contenant une activité, ils doivent lire les cellules de texte permettant de suivre le scénario de cette activité. De nombreuses études se sont penchées en profondeur sur les modèles de compréhension de la lecture et ont montré que la compréhension dépend de nombreux facteurs, tels que les caractéristiques du lecteur, le contenu et la conception du texte, ainsi que les instructions de lecture [6, 10]. Une étude récente démontre même clairement que les scores de compréhension de la lecture sont plus bas pour les textes numériques que pour les textes imprimés, quel que soit l'âge [4]. Il est donc possible que même avec les meilleures intentions, les étudiants puissent involontairement aboutir à un état du notebook non prévu par l'enseignant, c'est à dire à un état du notebook qui n'est pas cohérent avec le scénario mis en place par l'enseignant. Imaginez un notebook dans lequel l'étudiant exécute séquentiellement les cellules `C1`, `C3`, `C5` et `C7`, puis modifie l'image utilisée dans la cellule `C3`, comme demandé. Cependant, il ne relance pas la cellule `C5`, pensant qu'elle est destinée uniquement à afficher les dimensions de l'image, et exécute plutôt directement `C7` pour calculer le ratio de pixels noirs. Le résultat obtenu est incorrect car le ratio de pixels noirs pour l'image *black_square_100x100.png* est calculé avec les dimensions en hauteur et en largeur de l'image *white_square_255x255* qui n'a pas été mise à jour (cellule C5). Sans un retour immédiat sur le problème, il existe un risque élevé que l'étudiant ne remarque pas l'erreur et continue, ce qui peut entraîner d'autres problèmes et nuire à son apprentissage. Enfin, parce que les notebooks favorisent une programmation exploratoire, les étudiants peuvent insérer de nouvelles cellules de code n'importe où dans le notebook pour expérimenter, supprimer des cellules ou modifier des cellules existantes sans garantir la cohérence de l'ensemble du notebook. Il est donc facile pour les étudiants de corrompre leurs notebooks, parfois sans le vouloir, ce qui nuit à la cohérence des résultats avec le scénario de l'activité.

Pour résoudre ces problèmes et soutenir les étudiants dans la manipulation des notebooks éducatifs, nous proposons une nouvelle approche, MOON, entièrement intégrée à Jupyter, qui offre à l'enseignant un langage simple pour scripter le scénario d'une activité directement dans le notebook. Grâce à MOON les étudiants peuvent ensuite lancer le script et bénéficier d'une assistance visuelle leur permettant de les guider à travers les différentes étapes du scénario de l'activité.

5.3 Présentation de MOON

Notre approche, MOON, a deux objectifs principaux :

- permettre à l'enseignant d'exprimer le scénario de son activité sous la forme d'un script directement dans le notebook,
- interpréter le script pour enrichir le notebook avec des indications visuelles pour l'étudiant. Ces indications permettent d'identifier les prochaines cellules à exécuter et donnent des informations sur les exécutions passées ou à venir de toutes les

cellules du notebook.

5.3.1 Du scénario vers le script

Un scénario est incarné par les cellules de code et de texte du notebook éducatif. Pour mieux comprendre les schémas d'exécution des cellules de code utilisées par les enseignants dans leurs activités, nous avons mené une expérience informelle. Nous avons analysé environ une centaine de notebooks sur GitHub et Kaggle qui incluaient des instructions pour le lecteur. Notre corpus de notebooks était principalement constitué de notebooks éducatifs, essentiellement issus de cours universitaires. Ces notebooks ont été sélectionnés pour couvrir un large éventail de disciplines académiques et de sujets, afin d'avoir une collection exhaustive de matériel éducatif (feuilles d'exercices, devoirs et manuels interactifs). Nous avons identifié trois principaux modèles d'exécution des cellules : i) exécution linéaire, ii) exécution non linéaire, et iii) exécution facultative. Nous avons ensuite créé un langage de script simple qui couvre ces trois modèles d'exécution et permet à l'enseignant de définir la séquence d'exécution prévue des cellules de code dans un notebook. Nous nous limitons à ces trois schémas car ils sont suffisants pour exprimer tous les scénarios que nous avons rencontrés.

La structure de base manipulée dans notre langage de script est la cellule de code, chacune étant éventuellement associée à un ensemble de cellules de texte contenant les instructions correspondantes. Les cellules de code sont écrites comme suit : `Ci~Tj~...~Tn`, où $i, j, n \in \mathbb{N}$ désignent les indices des cellules impliquées dans le notebook. Par exemple, dans le notebook de la Figure 5.1, `C1~T0` désigne la cellule de code `C1` et ses instructions associées dans la cellule de texte `T0`. Dans la suite, nous omettons les cellules textuelles des scripts pour des raisons de clarté. L'ordre d'exécution pour l'ensemble du notebook est ensuite spécifié en combinant les cellules à l'aide de trois opérateurs dérivés des schémas d'exécution mentionnés ci-dessus.

Le modèle d'exécution linéaire Le modèle d'exécution linéaire est naturellement le plus courant, car il correspond à l'ordre de lecture standard de haut en bas avec la possibilité d'exécuter une même cellule plusieurs fois dans la séquence. Pour exprimer qu'un ensemble de cellules de code doit être exécuté linéairement, notre langage de script utilise l'opérateur parenthèses `()`. Par exemple la première partie du scénario (Figure 5.1) implique un schéma d'exécution linéaire qui nécessite l'exécution séquentielle des cellules de code suivantes : `C1 C3 C5 C7 C3 C5 C7`. En utilisant le langage de script, cela s'exprime comme suit : `(C1 C3 C5 C7 C3 C5 C7)`.

Le modèle d'exécution non linéaire Le modèle d'exécution non linéaire indique qu'un ensemble de cellules peut être exécuté dans n'importe quel ordre. Pour exprimer cette possibilité nous utilisons l'opérateur crochets `[]`. Par exemple, avec deux cellules de code notées `Ci` et `Cj`, nous écrivons `[Ci Cj]` ce qui donne deux possibilités d'exécution `Ci` puis `Cj` ou `Cj` puis `Ci`.

Le modèle d'exécution optionnel Le modèle d'exécution optionnel, noté avec un point d'interrogation, indique qu'une cellule de code peut être exécutée ou non. Par exemple, la première partie du notebook (Figure 5.1) propose la cellule `C10` comme optionnelle. Le script est exprimé comme suit : `(C1 C3 C5 C7 ?C10)`, dans une séquence d'exécution linéaire, l'exécution de la cellule `C10` est en option.

Notre langage de script permet surtout de composer ces modèles d'exécution. Pour illustrer cette combinaison d'opérateurs, considérons le notebook de la Figure 5.1 dans sa totalité. L'étudiant doit commencer par la première partie, puis les deux parties restantes

(Partie A et Partie B) peuvent être faites dans n'importe quel ordre. La première partie est écrite comme suit dans le script : `(C1 C3 C5 C7 C3 C5 C7 ?C10)`, avec `C10` comme cellule optionnelle. Les scripts pour les parties A et B sont les suivants : A→`(C12 C14)` et B→`(C16 C18)`. Les trois parties sont réunies en combinant leurs scripts avec les modèles d'exécution linéaires et d'ordre quelconque, comme suit :

```
((C1 C3 C5 C7 C3 C5 C7 ?C10)[(C12 C14)(C16 C18)]).
```

Les opérateurs parenthèses ou crochets peuvent être précédés d'un point d'interrogation pour indiquer des groupes de cellules optionnelles.

5.3.2 Notebook éducatif augmenté

Une fois que l'enseignant a rédigé le script qui met en œuvre son scénario, celui-ci est intégré à MOON pour accompagner les étudiants lors de leurs travaux sur les notebooks. MOON utilise un système de codage visuel à trois couleurs pour guider les étudiants dans l'exécution du scénario. Plus précisément, à chaque étape, MOON met en évidence les cellules pouvant être exécutées en vert, les cellules déjà exécutées en orange et les cellules qui ne sont pas encore prêtes pour l'exécution en rouge. Attention MOON n'offre aucune garantie sur ce que l'étudiant a pu écrire dans les cellules de code, il se contente de guider l'utilisateur vers le prochain état valide du scénario. Dans l'exemple de la Figure 5.2, nous pouvons voir que les cellules `C1`, `C3`, `C5` et `C7` sont colorées en orange, ce qui indique que l'étudiant les a déjà exécutées. MOON propose maintenant trois cellules vertes prêtes à être exécutées : la cellule de code optionnelle `C10~T9`; la Partie A avec la cellule de code `C12~T11`; et la Partie B avec la cellule de code `C16~T15`. Notez que les cellules de texte associées sont également mises en évidence pour guider l'étudiant vers les cellules contenant les instructions des différentes tâches immédiatement accessibles. Si l'étudiant ignore la cellule de code optionnelle `C10`, elle devient rouge à l'étape suivante, tandis que si elle est exécutée, elle devient orange. Les tâches qui ne sont pas encore accessibles et qui ne doivent pas être exécutées sont colorées en rouge. Si un étudiant exécute une cellule rouge, les couleurs des cellules restent inchangées. La seule façon de progresser dans le scénario et de mettre en évidence le prochain ensemble de cellules vertes, c'est à dire les prochains états valides, est d'exécuter l'une des cellules vertes. Ces indicateurs visuels informent les étudiants de leur progression dans le notebook.

De plus, MOON décore la dernière cellule exécutée avec des boutons pointant vers les prochaines cellules à exécuter. Cette fonctionnalité sert deux objectifs principaux : i) donner un aperçu des prochaines cellules possibles sans avoir à faire défiler la page si elles se trouvent en dehors de la zone visible de l'écran ; ii) permettre aux étudiants de naviguer rapidement dans le notebook en cliquant sur le bouton correspondant à la cellule de code désirée. Dans notre exemple (Figure 5.2), il y a trois boutons pour indiquer et accéder aux trois cellules qui peuvent maintenant être exécutées, celui de la cellule optionnelle (`C10`), celui de la cellule (`C12`) menant à la partie A et celui de la cellule (`C16`) menant à la partie B de l'activité.

5.3.3 Support de la programmation exploratoire

Le notebook Jupyter de part sa nature est un outil puissant pour les activités d'apprentissage, il permet aux étudiants d'apprendre et d'expérimenter la programmation avec une grande flexibilité. À première vue, MOON pourrait réduire considérablement cette flexibilité en obligeant les étudiants à suivre le scénario de l'enseignant. Cependant, pour

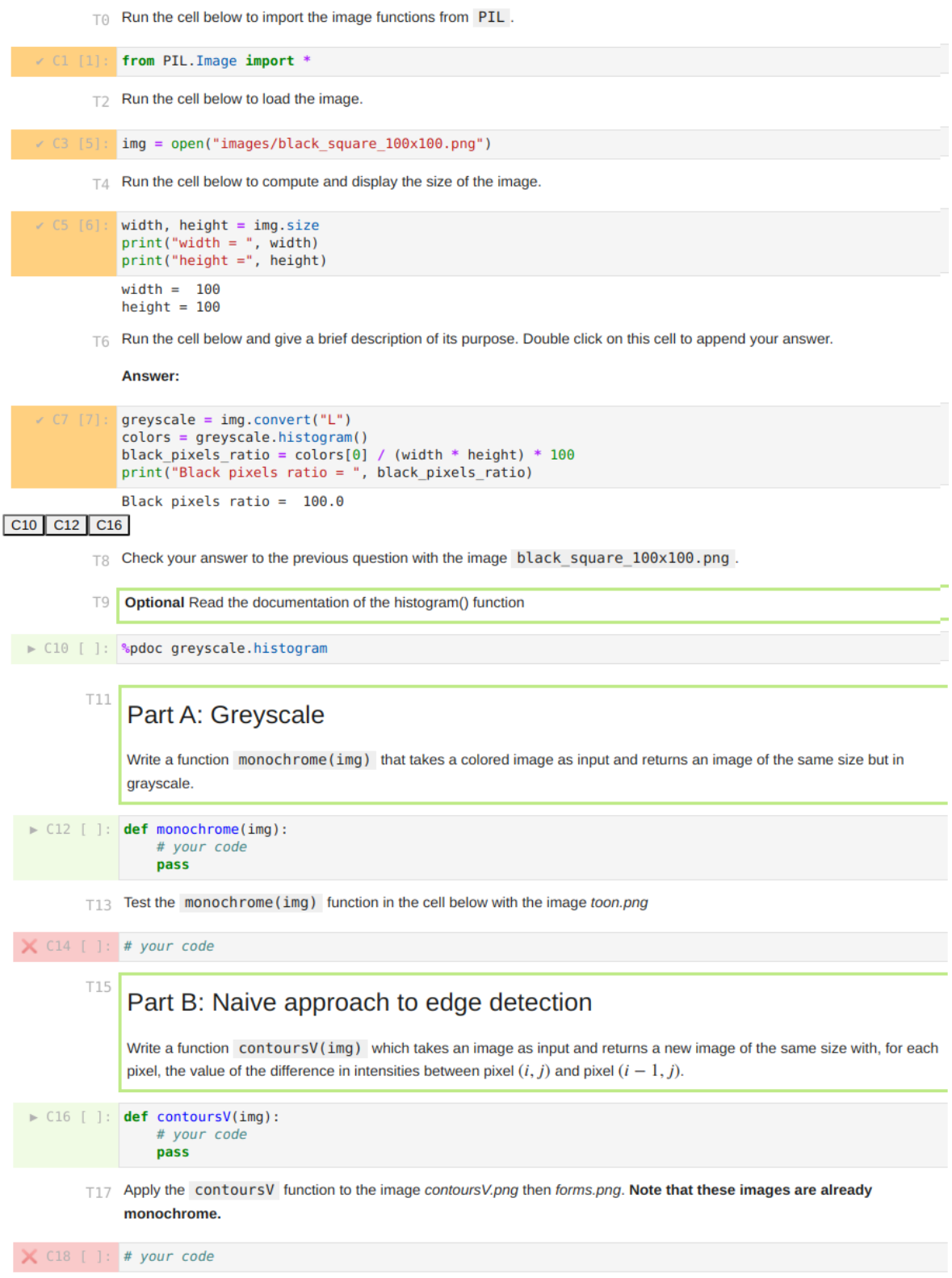


FIGURE 5.2 – Dans ce notebook la progression de l'étudiant dans le scénario de l'activité est mis en couleur avec MOON. En vert les prochaines tâches à réaliser. En orange, ce qui a déjà été fait par l'étudiant et en rouge les tâches qui ne sont pas encore accessibles. Cet exemple présente l'état du notebook après que l'étudiant ait exécuté les cellules de code dans l'ordre suivant : `C1` `C3` `C5` `C7` `C3` `C5` `C7`

ne pas freiner l'exploration des étudiants, MOON laisse une grande liberté en permettant aux étudiants de réexécuter des cellules de code, d'insérer ou de supprimer de nouvelles cellules de code tout en maintenant le scénario prévu à jour. Cette mise à jour des prochains états valides est assurée par la fonctionnalité : *Automatic backtracking*.

Automatic Backtracking Durant la phase exploratoire, les étudiants peuvent modifier et réexécuter les cellules qu'ils ont déjà exécutées, généralement pour corriger leur code ou explorer des alternatives. Dans MOON, cela signifie qu'un étudiant peut exécuter une cellule orange ne correspondant pas au prochain état valide du scénario. Ces phases d'exploration sont nécessaires pour les étudiants mais potentiellement dangereuses pour le modèle d'exécution du notebook, comme le montre l'exemple de la Figure 5.2. Dans ce cas, MOON offre la fonctionnalité *Automatic Backtracking* qui aide l'étudiant à retrouver un état valide cohérent avec le scénario de l'enseignant aussi facilement que possible. Cette fonctionnalité est accessible à partir d'un menu du même nom. Lorsqu'une cellule orange est exécutée, MOON ramène les étudiants à la dernière exécution de cette cellule prévue par le script et met à jour les couleurs de toutes les cellules du notebook en conséquence. Bien que les couleurs attribuées aux différentes cellules sont mises à jour, l'état de la mémoire du notebook ne change pas, charge aux étudiants de gérer de potentielles erreurs.

Si nous reprenons notre exemple (Figure 5.1) associé au script `(C1 C3 C5 C7 ?C10)`, lorsque l'étudiant décide d'utiliser l'image du carré noir pour vérifier le ratio de pixels noirs et qu'il oublie de valider la cellule `C5`, il obtiendra des résultats trompeurs à l'exécution de la cellule `C7`. Avec MOON, l'exécution de la cellule de code orange `C3` ramène l'étudiant dans le script en colorant à nouveau la cellule `C5` en vert. L'exécution de la cellule `C5` permet d'obtenir la mise à jour des variables `width` et `height` de l'image *black_square.png* et la mise en évidence en vert de la cellule de code `C7`. Lorsque l'étudiant exécute la cellule `C7`, il obtient maintenant le résultat correct pour le ratio de pixels noirs. Enfin, MOON inclut également un bouton *retour* qui permet aux étudiants de revenir à l'étape précédente du scénario (nous verrons un exemple dans la section suivante 5.4.2).

Ajout et Suppression de Cellules MOON conserve la possibilité d'ajouter de nouvelles cellules de code et de texte et de les supprimer par la suite. Par exemple, les étudiants peuvent créer des cellules de code supplémentaires au-delà de celles fournies par le notebook initial, qui peuvent servir de cellules permettant d'expérimenter plusieurs hypothèses, ou créer des cellules de texte pouvant être utilisées pour la prise de notes, ce qui leur permet de conserver tout leur travail dans un seul document. Lorsqu'une de ces cellules est créée, elle reste blanche pour la distinguer des cellules du scénario. MOON ajuste également les indices des cellules appartenant au scénario dans le script, si nécessaire. Cependant, aucune des cellules impliquées dans le scénario ne peut être supprimée une fois que le script a été chargé.

5.4 Implementation

5.4.1 Compilation DFA

MOON est implémenté en tant que plugin JupyterLab qui prend en entrée un script tel que défini dans la section précédente. Notez que nos scripts sont des expressions régulières, à l'exception des modèles d'exécutions non linéaires. Nous convertissons ces modèles d'exécutions pour obtenir toutes les combinaisons possibles de schémas linéaires

correspondants et produisons un automate fini déterministe (DFA) en utilisant l'algorithme décrit dans *Introduction to automata theory, languages, and computation* [8]. Par conséquent, il n'est pas possible d'avoir un scénario qui serait composé d'un grand nombre de cellules qui pourraient être exécutées dans n'importe quel ordre en raison d'une complexité exponentielle liée à notre implémentation. Pour construire l'AST d'un script, nous utilisons tsPEG [1], un générateur de parseur PEG conçu pour TypeScript. Dans le contexte d'un notebook, le DFA est un modèle qui représente un ensemble d'états et de transitions, basé sur les exécutions de cellules de code. Si nous prenons notre notebook comme exemple (Figure 5.1) et le script suivant : `C7 ?C10 [(C12 C14) (C16 C18)]`, nous construisons le DFA illustré dans la Figure 5.3.

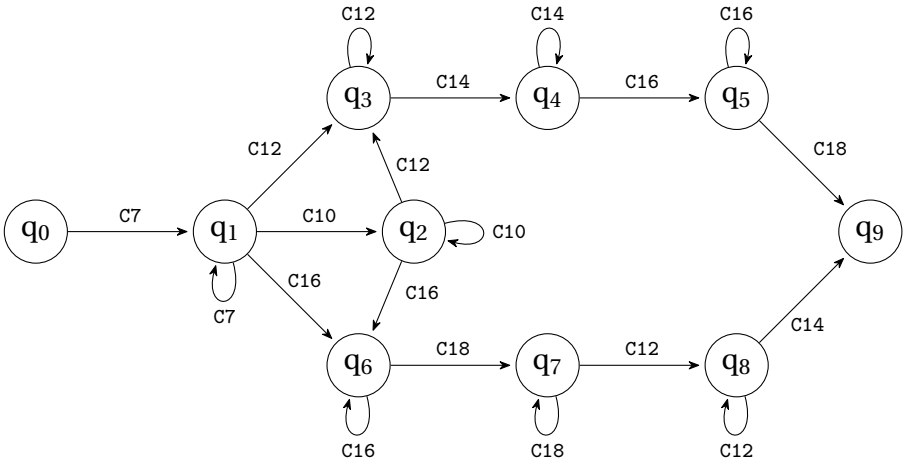


FIGURE 5.3 – Automate obtenu avec le script : `(C7 ?C10 [(C12 C14)(C16 C18)])`. Les étiquettes des arêtes indiquent les cellules possibles à exécuter pour avoir un état valide.

5.4.2 Trace de l'utilisateur

En plus du DFA, nous conservons également une trace de l'utilisateur qui est une séquence de paires $(\texttt{Ci}, q_j)$ correspondant à la liste des transitions valides `Ci` menant à un état q_j effectuées par un utilisateur donné. Notez que les transitions invalides ne sont pas enregistrées dans cette trace. Par exemple, si un utilisateur exécute `C7`, `C12` et `C18` avec le script précédemment décrit et en commençant par q_0, la trace de l'utilisateur contiendra : $(\texttt{C7}, q_1), (\texttt{C12}, q_3)$.

Lorsque l'utilisateur exécute une cellule de code, l'automate vérifie si cette exécution correspond à une transition possible pour l'état actuel. Si la transition est autorisée, l'automate change d'état et les couleurs des cellules du notebook (code et texte) sont mises à jour. Pour notre exemple (Figure 5.3), considérons l'automate dans l'état $q1$ avec une trace utilisateur contenant $(\texttt{C7}, q_1)$. Toutes les cellules appartenant à la trace de l'utilisateur sont orange. Il y a trois transitions autorisées par l'automate. L'exécution de la cellule de code `C10` qui est facultative et l'exécution des cellules de code `C12` et `C16` dans n'importe quel ordre. Ces trois cellules de code, qui représentent les prochaines tâches possibles, sont vertes, tout comme leurs cellules de texte associées, et les autres cellules de code sont rouges (Figure 5.2). Voyons maintenant ce qui se passe en fonction de l'exécution de la cellule de code optionnelle `C10`. Si l'utilisateur valide la transition associée à la cellule `C10`, elle est ajoutée à la trace $(\texttt{C7}, q_1), (\texttt{C10}, q_2)$ et devient orange.

1. https ://github.com/EoinDavey/tsPEG

Dans ce cas, il existe maintenant deux transitions possibles : l'exécution des cellules de code `C12` et `C16`, dont les cellules sont colorées en vert sur le notebook. Les autres cellules de code sont rouges. Si l'utilisateur valide la transition associée à la cellule de code `C12` sans avoir validé la transition associée à la cellule de code `C10`, la trace de l'utilisateur devient $(\texttt{C7}, q_1), (\texttt{C12}, q_3)$ et la cellule de code `C12` devient orange. Dans ce cas, la prochaine transition autorisée est associée à la cellule de code `C14` (vert) et les cellules de code `C10` et `C16` deviennent rouges sur le notebook. Nous avons également implémenté la possibilité pour l'étudiant de réexécuter une cellule orange et de placer l'automate dans l'état correspondant à cette transition dans le script. Par exemple, si nous avons la trace utilisateur suivante : $(\texttt{C7}, q_1), (\texttt{C12}, q_3), (\texttt{C14}, q_4)$, sur le notebook la prochaine tâche est désignée par la cellule de code verte `C16`. Imaginez maintenant que l'étudiant modifie la cellule de code orange `C12` puis l'exécute. L' *automatic backtracking* recherche dans la trace utilisateur la dernière paire contenant la transition associée à la cellule de code `C12` en supprimant toutes les suivantes dans la trace. Enfin, la trace utilisateur ne contient que $(\texttt{C7}, q_1), (\texttt{C12}, q_3)$ et le DFA est réinitialisé à l'état q_3. Les cellules de code `C7` et `C12` sont oranges, la cellule de code `C14` est verte et les autres cellules de code sont rouges.

Illustrons une autre situation possible avec la nouvelle trace utilisateur suivante :

$$(\texttt{C0}, q_1), (\texttt{C2}, q_2), (\texttt{C0}, q_3), (\texttt{C4}, q_4)$$

Le DFA est à l'état q_4. À partir de cette trace, nous pouvons observer que le scénario demande à l'étudiant d'exécuter la cellule de code `C0` deux fois, d'abord au début puis après l'exécution de la cellule de code `C2`. Si l'étudiant décide d'exécuter à nouveau la cellule de code `C0`, l'implémentation de l'*automatic backtracking* réinitialisera le DFA à l'état q_3 et la cellule de code `C4` deviendra verte. Dans ce cas, avec l'*automatic backtracking*, il n'est pas possible de réinitialiser le DFA à l'état q_1. Pour surmonter cette limitation, nous avons également implémenté un bouton *retour* qui permet aux utilisateurs de réinitialiser le DFA à l'état précédent dans la trace utilisateur. Chaque clic sur le bouton *retour* supprime la dernière paire dans la trace utilisateur, réinitialise le DFA à l'état précédent et met à jour les couleurs des cellules dans le notebook. Une fois que le DFA est à l'état q_4, il suffit de cliquer quatre fois sur le bouton *retour* pour réinitialiser le DFA à l'état q_1. Dans une trace utilisateur plus complexe, il est également possible de combiner l'*automatic backtracking* avec le bouton *retour* pour revenir à un état précédent.

5.5 Expérience contrôlée

Pour évaluer la capacité de MOON à assister les étudiants dans les activités des notebooks éducatifs, nous avons mené une expérience contrôlée. Plus précisément, notre évaluation vise à répondre aux deux questions de recherche suivantes :

RQ1 Est-ce que MOON aide les étudiants à mieux respecter les consignes d'un notebook éducatif?

RQ2 Est-ce que MOON entrave la capacité des étudiants à progresser dans le notebook?

Notre hypothèse pour RQ1 est que l'assistance fournie par MOON pourrait réduire la quantité d'utilisation incorrecte du notebook. Notre hypothèse pour RQ2 est que l'assistance fournie par MOON pourrait rendre plus longue et plus complexe la manipulation du notebook avec le risque d'entraver la progression des étudiants.

5.5.1 Conception expérimentale

Pour étudier l'impact de MOON, nous avons procédé à une expérience contrôlée de type A/B testing, où un groupe d'étudiants reçoit un notebook sans MOON, et l'autre groupe se voit attribuer le même notebook avec le plugin MOON à activer dès le début de la séance.

Nous avons mené notre expérience dans le cadre d'un cours d'introduction à la programmation impliquant 21 étudiants de première année de licence à l'université de Bordeaux au cours du premier semestre. Ce cours est organisé autour d'un ensemble de chapitres concernant les tableaux, les algorithmes de tri, les transformations géométriques dans les images et une introduction à la théorie des graphes. Chaque chapitre est implémenté sous la forme d'un notebook qui entrelace les concepts du cours, des médias, le code source et les questions de codage. L'étude a eu lieu dans la première semaine de janvier 2023, les étudiants ayant déjà manipulé des notebooks lors de 6 sessions pratiques de 2H40, et l'étude s'est déroulée lors de la 7e et dernière session. Par conséquent, les étudiants avaient déjà une certaine expérience avec les notebooks.

Pour l'expérience, nous avons assigné aléatoirement 11 étudiants au groupe témoin et 10 étudiants au groupe expérimental qui utilise MOON. Les étudiants utilisant MOON ont reçu une courte session de formation de vingt minutes où l'auteur de cette thèse, qui était l'enseignant des étudiants, a expliqué la signification des couleurs (5.3.2) et les fonctionnalités du menu de retour en arrière (5.3.3) sur un notebook servant d'exemple.

Le notebook utilisé dans cette expérience se compose de 3 exercices sur la théorie des graphes avec différents modèles d'exécution de cellules [2]. Le premier exercice suit un modèle d'exécution linéaire avec une cellule facultative. Le deuxième exercice combine des modèles d'exécution linéaire et non linéaire. Le troisième exercice n'implique qu'un modèle d'exécution linéaire. Les deux derniers exercices peuvent être réalisés indépendamment, de sorte que l'étudiant puisse commencer par l'un ou l'autre des exercices. Le notebook complet nécessite 23 exécutions de cellules réparties dans 20 cellules de code et 23 cellules de texte : trois cellules de code doivent être exécutées au moins deux fois. Chaque cellule de code nécessite que l'étudiant exécute, complète ou remplisse la cellule. Ce notebook a été distribué aux étudiants lors d'une session pratique de 2h40. Lors de la session, l'enseignant avait pour tâche de répondre uniquement aux questions liées au contenu du cours (théorie des graphes) et de ne pas aider les étudiants à manipuler le notebook afin de ne pas biaiser le comportement des étudiants. À la fin de la session pratique, l'enseignant a relevé tous les notebooks produits par les étudiants.

Pour répondre à nos questions de recherche, nous avons enregistrés la manière dont les étudiants manipulaient le notebook. À cette fin, nous avons instrumenté JupyterLab pour les deux groupes afin d'enregistrer une trace complète des actions d'exécution de cellules des étudiants dans le notebook. Cette trace a été enregistrée dans les métadonnées des notebooks. Notez que la trace diffère de la trace utilisateur décrite dans la section 5.4 car i) elle enregistre chaque exécution de cellule, même celles qui ne respectent pas le scénario, et ii) elle n'est jamais modifiée, même en cas d'utilisation de l'*automatic backtracking*. Enfin, nous simplifions les traces en supprimant et en remplaçant les séquences d'exécution de la même cellule par une seule exécution de cette cellule. Nous effectuons cette simplification car il est courant que les étudiants exécutent plusieurs fois la même cellule de code lorsqu'ils y travaillent. Cependant, cela n'affecte pas la conformité avec le scénario prévu. Pour étudier RQ1, nous introduisons une métrique pour capturer le nombre d'utilisations incorrectes du notebook. Notre métrique est donc définie en fonction du concept de *deviation* (terme anglais) par rapport au scénario, où une déviation est l'exécution d'une cellule rouge, c'est-à-dire l'exécution d'une cellule qui n'est pas au-

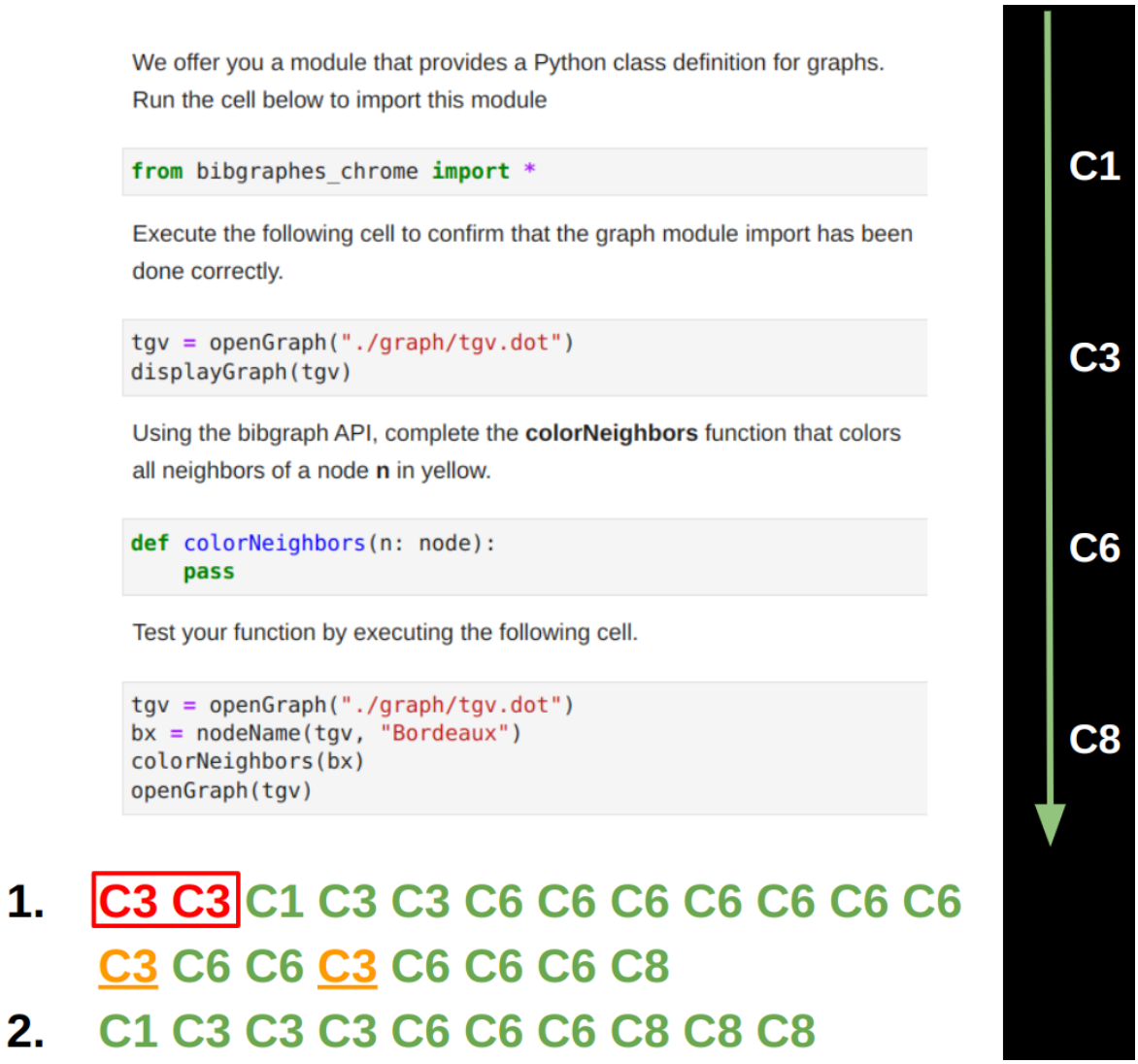
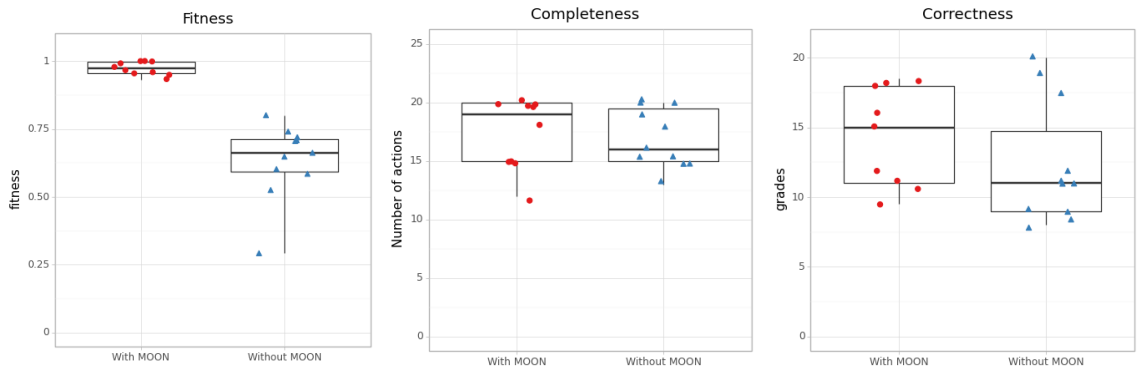


FIGURE 5.4 – Extrait du notebook de l'expérience contrôlée. L'exécution des 4 cellules de code présentées est linéaire du haut vers le bas (`C1 C3 C6 C8`). Sous le notebook 2 extraits de traces étudiantes enregistrées durant la séance. En vert l'exécution d'une cellule autorisée par le scénario. En rouge l'exécution d'une cellule qui n'est pas autorisée par le scénario. En orange l'exécution d'une cellule déjà exécutée. Dans la première trace, l'étudiant exécute la cellule `C3` sans avoir préalablement importé le module associé aux graphes. Ce faisant, il ne suit pas la consigne donnée dans la première cellule de texte. Ensuite il exécute plusieurs fois les cellules de code `C3` et `C6` ce qui n'est pas cohérent avec les consignes. La deuxième trace est valide c'est à dire conforme avec le scénario.

torisée dans le scénario (Figure 5.4). Pour déterminer si les actions enregistrées dans les traces correspondent à une transition valide (verte), un retour en arrière (orange) ou une transition invalide (rouge), nous rejouons les traces des étudiants hors ligne avec MOON. Enfin, pour définir notre métrique, appelée *fitness*, nous notons g, o et r le nombre total de cellules vertes, orange et rouges exécutées dans une trace étudiante. La métrique *fitness* est définie comme le rapport des exécutions de cellules vertes et oranges sur le nombre total d'exécutions de cellules : fitness $= (g+o)/(g+o+r)$

Une valeur de 1 indique qu'un étudiant n'a jamais exécuté de cellule rouge, tandis qu'une valeur de 0 indique qu'un étudiant n'a exécuté que des cellules rouges. Pour évaluer les différences entre les groupes témoin et expérimental, nous utilisons un test de



(a) Fitness dans le groupe expérimental et de contrôle.

(b) Completeness dans le groupe expérimental et de contrôle.

(c) Correctness dans le groupe expérimental et de contrôle.

FIGURE 5.5 – Analyse des notebooks étudiants dans le groupe expérimental (avec MOON) et le groupe de contrôle (sans MOON).

Mann-Whitney U non paramétrique avec une valeur de p seuil de 0,01. Notre hypothèse nulle est que la métrique fitness est la même dans les deux groupes, et notre hypothèse alternative est que la métrique fitness est différente dans les deux groupes.

Pour étudier RQ2, nous introduisons une métrique pour capturer jusqu'où les étudiants sont allés dans le notebook. À cet égard, nous définissons la métrique *completeness* comme le nombre de cellules distinctes exécutées dans une trace donnée. Un étudiant qui n'a exécuté aucune cellule du notebook correspond a une valeur de la métrique completeness de 0, tandis qu'un étudiant qui a exécuté toutes les cellules possibles à l'intérieur du notebook a une valeur de la métrique *completeness* de 20. Pour évaluer la différence entre les groupes témoin et expérimental, nous utilisons un test de Mann-Whitney U non paramétrique avec une valeur de p seuil de 0,01. Nous évaluons également le code des étudiants en attribuant une note à chaque notebook. Notre hypothèse nulle est que la métrique *completeness* et/ou la note sont les mêmes dans les deux groupes, et notre hypothèse alternative est que la métrique *completeness* et/ou la note sont différentes dans les deux groupes.

5.5.2 Résultats

RQ1 : Suivi du notebook avec MOON

La Figure 5.5a représente la métrique fitness des traces des deux groupes. Visuellement, nous constatons une nette différence entre les deux groupes. La métrique fitness dans le groupe témoin varie entre environ 0,25 (ce qui signifie que seul un quart des exécutions se trouvent dans une cellule verte ou orange) et environ 0,75. Dans le groupe expérimental, les valeurs de cette métrique sont très proches de 1 pour toutes les traces, avec trois traces ayant une valeur de 1 (aucune exécution d'une cellule rouge). Le test de Mann-Whitney U donne une valeur de p de 0,000122 et une taille d'effet $rbc = -1$. Nous rejetons donc l'hypothèse nulle et acceptons l'hypothèse alternative. La taille d'effet indique un effet nettement en faveur du groupe utilisant MOON, aucune valeur de la métrique fitness du groupe de contrôle (valeur maximale de 0,8) ne dépassant la valeur la plus faible du groupe expérimental (0,91). En conclusion, les données soutiennent l'hypothèse selon laquelle MOON réduit le nombre d'utilisations incorrectes du notebook.

RQ2 : Progression des étudiants avec MOON

La Figure 5.5b représente la métrique completeness calculée à partir des traces des deux groupes et la Figure 5.5c montre les scores des étudiants pour la session pratique. Pour ces deux figures, visuellement, la distribution des valeurs est similaire entre les deux groupes, avec une dispersion comparable et une médiane légèrement plus basse dans le groupe témoin. Pour cette métrique, le test de Mann-Whitney U donne une valeur de p de 0,53 et une taille d'effet $rbc = -0,145455$. Ainsi, nous ne pouvons pas rejeter l'hypothèse nulle selon laquelle la complétude est la même entre les groupes. Pour les notes, le test de Mann-Whitney U donne une valeur de p de 0,29 et une taille d'effet $rbc = -0,292929$. Ainsi, nous ne pouvons pas rejeter l'hypothèse nulle selon laquelle les notes sont les mêmes entre les groupes. Le calcul de la médiane révèle une différence de 4 points en faveur du groupe avec MOON. De même, lors du calcul de la moyenne, le groupe expérimental a une moyenne de 14,3 avec un écart-type de 3,6, tandis que le groupe témoin a une moyenne de 12,4 avec un écart-type de 4,4. De plus, dans les deux cas, la taille de l'effet est faible et en faveur du groupe expérimental. Par conséquent, les données ne soutiennent pas l'hypothèse selon laquelle MOON entrave la progression des étudiants. Au contraire, il existe un léger avantage dans le groupe expérimental.

En résumé, basé sur notre expérience contrôlée, nous répondons à nos questions de recherche et au défi n°2 comme suit :

Défi n°2

MOON aide les étudiants à mieux respecter le scénario de l'activité contenue dans le notebook sans entraver leur capacité à progresser dans ce scénario.

5.5.3 Obstacles à la validité

Concernant la validité de construction de cette étude, la métrique completeness pourrait ne pas capturer avec précision le progrès réel des étudiants dans le notebook, car exécuter une cellule est différent de compléter le travail requis dans cette cellule. Pour contrer cet obstacle, nous avons passé en revue tous les notebooks pour s'assurer qu'il n'y avait pas de divergences entre la trace et les cellules complétées dans les 21 notebooks. Nous avons également évalué les notebooks étudiants, mais nous avons des réserves quant à la subjectivité de l'évaluation puisqu'elle a été réalisée par l'enseignant ayant rédigé l'activité et réalisé l'expérience.

En ce qui concerne la validité interne, la capacité de JupyterLab à redémarrer le noyau d'exécution peut perturber considérablement l'utilisation de MOON car cela réinitialise la mémoire du notebook tout en conservant l'état de MOON. Par conséquent, l'enseignant a demandé aux étudiants de réinitialiser MOON s'ils devaient redémarrer le noyau, mais certains étudiants auraient pu ignorer cette instruction. L'enseignant a également conseillé aux étudiants de cliquer uniquement sur le bouton de rafraîchissement du navigateur s'ils étaient certains d'avoir sauvegardé leurs notebooks. Cette action réinitialise le plugin et efface la trace de l'utilisateur pour en commencer une nouvelle. MOON utilise largement les couleurs pour communiquer les retours aux étudiants, ce qui pourrait poser problème à certains étudiants en raison, par exemple, du daltonisme. Pour lutter contre cette menace, des emojis ont été ajoutés avec les couleurs. Dans le groupe expérimental, aucun étudiant n'a eu de problème à utiliser MOON.

En ce qui concerne la validité externe, nous avons réalisé notre expérience contrôlée dans un seul contexte. Par conséquent, les résultats pourraient varier en fonction du niveau des étudiants ou du notebook utilisé dans la session.

5.6 Étude utilisateur

Pour compléter notre expérience contrôlée, nous avons demandé aux 10 étudiants impliqués dans le groupe expérimental et à 10 autres étudiants qui ont participé aux tests bêta de MOON de remplir un sondage en ligne anonyme sur MOON. Nous avons reçu 16 réponses à ce sondage, ce qui donne un taux de réponse de 80%. La Figure 5.6 illustre la répartition des réponses aux quatre questions principales.

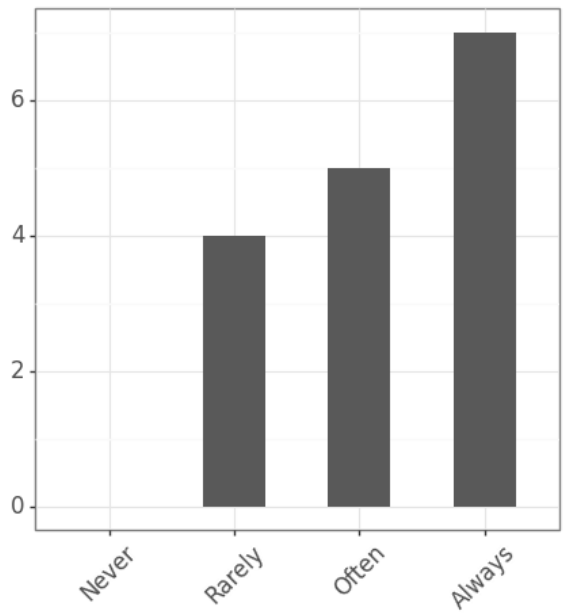
À la question "*Est-ce que MOON aide à suivre l'ordre d'exécution des cellules ?*", nous observons la répartition illustrée dans la Figure 5.6a. Nous notons que la réponse la plus fréquente est "Toujours", avec 7 réponses. Nous remarquons également qu'aucun participant n'a répondu "Jamais". Ce résultat corrobore ce que nous avons observé pour la RQ1 dans l'expérience contrôlée, où la précision observée dans le groupe expérimental était significativement plus élevée que dans le groupe témoin.

Pour obtenir une compréhension plus approfondie de la manière dont les étudiants perçoivent les actions suggérées par MOON, nous avons posé la question "Les actions suggérées par MOON sont-elles surprenantes ?" La Figure 5.6b représente les réponses à cette question. Nous constatons que la réponse la plus fréquente est "Jamais", avec 8 réponses, et seulement deux participants ont répondu "Souvent". Cela indique que les étudiants comprennent bien dans l'ensemble les actions suggérées par MOON, mais que certaines actions peuvent être surprenantes dans certains cas. Nous avons inclus une question à réponse libre pour ajouter plus de contexte à cette question. Deux étudiants ont mentionné qu'ils étaient surpris de devoir valider les cellules intermédiaires en cas de retour en arrière, même si ces cellules n'avaient pas été modifiées. Un étudiant a mentionné qu'il était surpris de pouvoir exécuter une cellule très éloignée de celle actuellement sélectionnée.

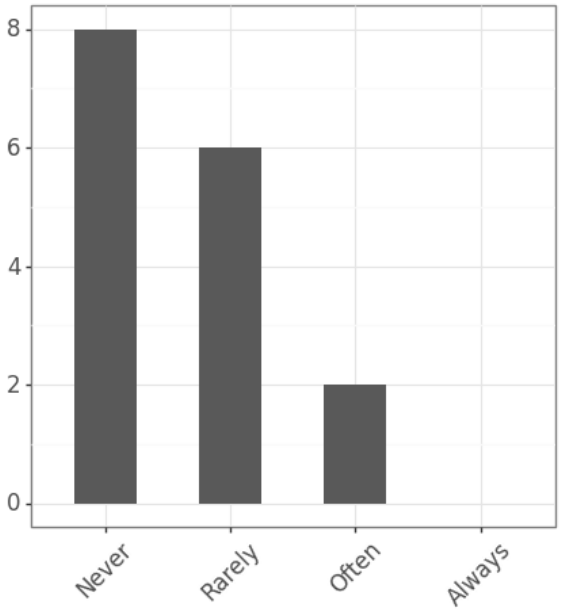
Pour corroborer les résultats de la RQ2 dans l'expérience contrôlée, nous avons demandé : "*Est-ce que MOON aide à progresser plus rapidement ?*" La Figure 5.6c représente les réponses à cette question. Nous notons que la réponse la plus fréquente est "*Oui*" avec 11 réponses, et seuls 5 étudiants ont répondu "*Non*". Cela correspond au fait que nous n'avons observé aucun effet de MOON sur la métrique *completeness*, et nous avons observé une médiane plus élevée dans le groupe expérimental.

Pour évaluer si les étudiants trouvent MOON utile, nous avons posé la question : "*Pensez - vous que MOON devrait être activé par défaut ?*", en d'autres termes, *est-ce qu'en tant qu'étudiant vous aimeriez utiliser MOON dans toutes vos séances de travaux pratiques ?* La Figure 5.6d représente les réponses à cette question. Inspirés par des travaux connexes [1], nous avons décidé d'utiliser une échelle de réponses de sondage asymétrique inspirée du modèle Kano, adaptée à l'évaluation des préférences. Nous notons que la réponse la plus fréquente est "*Plutôt utile*" avec 8 réponses, tandis que 4 étudiants ont répondu "*Indifférent*" et 4 "*Très utile*". Aucun des étudiants n'a répondu "*Mauvaise idée*". Cela indique que la plupart des étudiants trouvent MOON utile, tandis que certains ne semblent pas voir beaucoup d'avantages à l'utiliser. Nous conjecturons que cela pourrait être vrai pour les utilisateurs les plus avancés de notebooks.

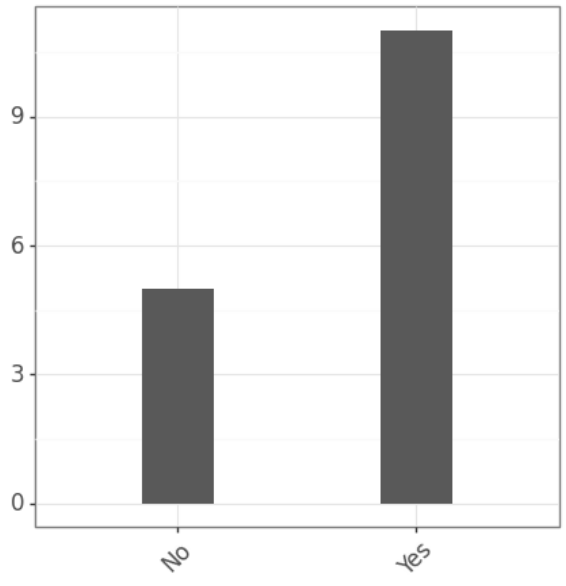
Enfin, nous avons inclus une dernière question à réponse libre demandant des commentaires sur MOON. Un étudiant a mentionné que la validation des cellules intermédiaires en cas de retour en arrière est fastidieuse. Un autre a mentionné que l'aide four-



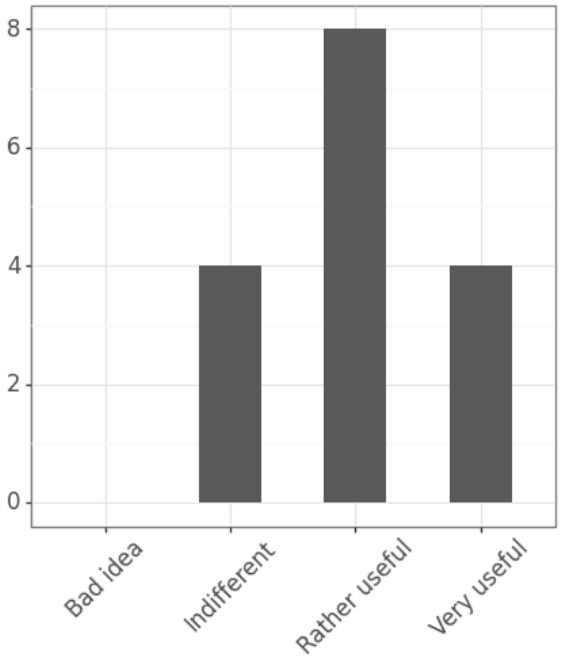
(a) Does MOON help to follow the execution order of cells?



(b) Are the actions suggested by MOON surprising?



(c) Does MOON help to progress faster?



(d) Do you think that MOON should be activated by default?

FIGURE 5.6 – Distributions des réponses étudiantes à notre questionnaire anonyme

nie par MOON n'était pas évidente au départ, mais lorsqu'il est revenu au notebook sans MOON la différence était apparente, le notebook semblait plus compliqué à utiliser.

5.7 Conclusion

Dans ce chapitre, nous présentons une approche innovante, MOON, conçue pour guider les étudiants à travers des scénarios de notebooks éducatifs. L'idée centrale est de fournir aux enseignants un langage qui leur permet de formaliser l'utilisation prévue de leurs notebooks sous forme de script, et d'interpréter ce script pour guider les étudiants avec des indications visuelles en temps réel pendant qu'ils interagissent avec les notebooks.

Nous avons évalué notre approche à l'aide d'une expérience contrôlée randomisée impliquant 21 étudiants, qui montre que MOON aide les étudiants à mieux respecter le scénario prévu sans entraver leur capacité à progresser. Notre étude de suivi auprès des utilisateurs montre qu'environ 75% des étudiants interrogés considèrent que MOON est assez utile ou très utile.

Dans nos travaux futurs, nous prévoyons d'explorer comment MOON peut être utilisé pour fournir des retours d'information aux enseignants. Une première idée serait d'analyser les traces et les scripts pour identifier les points problématiques dans un scénario et proposer des suggestions d'amélioration. Une deuxième idée serait de recueillir des données provenant des instances de MOON en temps réel afin d'assister directement les enseignants pendant une session pratique, en signalant les étudiants ayant des difficultés avec le notebook ou prenant du retard dans le scénario. Un autre point important à améliorer est l'écriture des scripts. Actuellement, notre approche impose d'écrire manuellement le script pour le scénario du notebook. Nous explorons des moyens de simplifier le processus d'écriture des scripts dans le cadre du scénario du notebook et de garantir la synchronisation du script lorsque des modifications sont apportées au scénario.

5.8 Références

[1] Andrew BEGEL et Thomas ZIMMERMANN. "Analyze this! 145 questions for data scientists in software engineering". In : *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*. Sous la dir. de Pankaj JALOTE, Lionel C. BRIAND et André van der HOEK. ACM, 2014, p. 12-23. DOI : 10.1145/2568225.2568233. URL : https://doi.org/10.1145/2568225.2568233.

[2] Christophe CASSEAU et al. *MOON: Assisting Students in Completing Educational Notebook Scenarios (Artifacts)*. Zenodo, juil. 2023. DOI : 10.5281/zenodo.8167588. URL : https://doi.org/10.5281/zenodo.8167588.

[3] Souti CHATTOPADHYAY et al. "What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities". In : *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*. Sous la dir. de Regina BERNHAUPT et al. ACM, 2020, p. 1-12. DOI : 10.1145/3313831.3376729. URL : https://doi.org/10.1145/3313831.3376729.

[4] Pablo DELGADO et al. "Don't throw away your printed books: A meta-analysis on the effects of reading media on reading comprehension". In : *Educational Research Review* 25 (2018), p. 23-38. ISSN : 1747-938X. DOI : 10.1016/j.edurev.2018.09.003. URL : https://doi.org/10.1016/j.edurev.2018.09.003.

[5] Andrew DILLON. "Reading from paper versus screens: a critical review of the empirical literature". In : *Ergonomics* 35.10 (1992), p. 1297-1326. URL : https://doi.org/10.1080/00140139208967394.

[6] Peter DIXON. "Plans and written directions for complex tasks". In : *Journal of verbal learning and verbal behavior* 21.1 (1982), p. 70-84.

[7] Andrew HEAD et al. "Managing Messes in Computational Notebooks". In : *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*. Sous la dir. de Stephen A. BREWSTER et al. ACM, 2019, p. 270. DOI : 10.1145/3290605.3300500. URL : https://doi.org/10.1145/3290605.3300500.

[8] John E. HOPCROFT, Rajeev MOTWANI et Jeffrey D. ULLMAN. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley, 2007. ISBN : 978-0-321-47617-3.

[9] Mary Beth KERY et al. "The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool". In : *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. Sous la dir. de Regan L. MANDRYK et al. ACM, 2018, p. 174. DOI : 10.1145/3173574.3173748. URL : https://doi.org/10.1145/3173574.3173748.

[10] Danielle S. MCNAMARA et Joe MAGLIANO. "Chapter 9 Toward a Comprehensive Model of Comprehension". In : *The Psychology of Learning and Motivation*. T. 51. Psychology of Learning and Motivation. Academic Press, 2009, p. 297-384. URL : https://www.sciencedirect.com/science/article/pii/S0079742109510092.

[11] Jan M. NOYES et Kate J. GARLAND. "Computer- vs. paper-based tasks: Are they equivalent?" In : *Ergonomics* 51.9 (2008). PMID: 18802819, p. 1352-1375. DOI : 10.1080/00140130802170387. URL : https://doi.org/10.1080/00140130802170387.

[12] Adam RULE, Aurélien TABARD et James D. HOLLAN. "Exploration and Explanation in Computational Notebooks". In : *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. Sous la dir. de Regan L. MANDRYK et al. ACM, 2018, p. 32. DOI : 10.1145/3173574.3173606. URL : https://doi.org/10.1145/3173574.3173606.

[13] Lauren M. SINGER et Patricia A. ALEXANDER. "Reading on Paper and Digitally: What the Past Decades of Empirical Research Reveal". In : *Review of Educational Research* 87.6 (2017), p. 1007-1041. DOI : 10.3102/0034654317722961. URL : https://doi.org/10.3102/0034654317722961.

Chapitre 6

Conclusion et Perspectives

« Très tôt, j'ai compris que l'imprévisible était la loi. »

Paul Auster

Sommaire

6.1 Conclusion

En conclusion, cette thèse a exploré l'utilisation des notebooks comme un outil d'apprentissage interactif dans le domaine de l'enseignement. À travers une analyse approfondie, nous avons mis en lumière l'importance croissante des notebooks dans les contextes éducatifs. Nous avons montré qu'il est utilisé dans de grandes universités comme le MIT ou encore l'EPFL et que l'état français avait investi dans un projet de recherche (CANDYCE) intégrant la plateforme Jupyter pour l'enseignement primaire et secondaire. Dans un contexte éducatif les notebooks Jupyter sont particulièrement utilisés dans le cadre de l'apprentissage et de l'enseignement des disciplines scientifiques et techniques mais également, dans une moindre mesure, dans des disciplines liées aux sciences humaines. Nous avons identifié plusieurs enjeux clés qui découlent de cette utilisation, notamment la reproductibilité des résultats affichés dans un notebook ainsi que la nécessité de suivre et de guider efficacement les étudiants dans l'exécution de scénarios d'activités préétablis.

Dans le chapitre 2 de cette thèse, nous avons défini la notion de notebook, puis nous avons focalisé notre attention sur les notebooks Jupyter, en soulignant leur importance pour les data scientists ainsi que dans le domaine de l'éducation. Ensuite nous avons introduit et explicité la notion de notebook éducatif ainsi que leur cycle de vie. Après avoir mis en lumière les nombreux avantages des notebooks pour l'enseignement nous avons mis en évidence deux défis de taille liés à la notion de reproductibilité.

Dans les chapitres 4 et 5 de cette thèse, nous avons proposé deux approches concrètes accompagnées d'outils directement accessibles depuis l'interface de JupyterLab et indépendant du langage de programmation utilisé dans le notebook. Tout d'abord, nous avons présenté NORM, une approche pratique visant à améliorer la reproductibilité des notebooks avec un scénario linéaire du haut vers le bas Nous avons ensuite détaillé la conception et la mise en œuvre d'une étude empirique visant à évaluer l'efficacité de NORM dans un contexte éducatif. Les résultats de cette étude ont démontré que NORM permettait une meilleure reproductibilité des résultats dans les notebooks étudiants, sans entraver leur progression. Dans la même lignée, nous avons introduit MOON, un outil conçu pour améliorer le suivi des scénarios d'activités dans les notebooks en fournissant des indications visuelles en temps réel aux étudiants. Nous avons détaillé la conception, l'implémentation et l'évaluation de MOON, en mettant en avant son potentiel à améliorer l'expérience de navigation dans le scénario d'une activité.

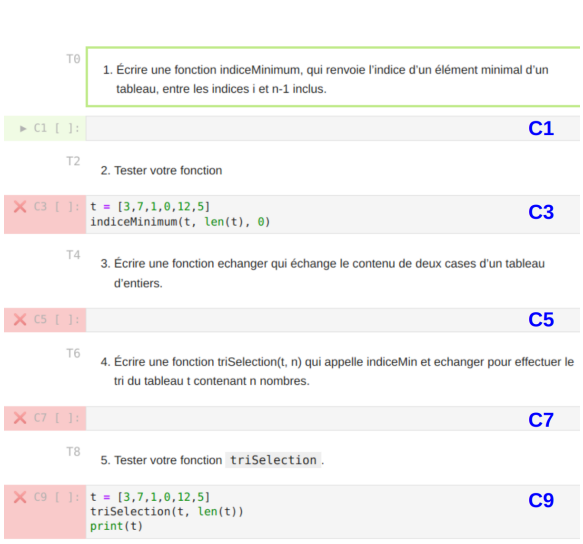
Avec NORM et MOON nous avons apporté une réponse aux deux défis que nous nous étions fixés. Enfin, nous discutons dans la section suivante des pistes pour des recherches futures. Nos résultats ouvrent de nouvelles perspectives pour l'intégration des notebooks dans les pratiques pédagogiques contemporaines.

6.2 Perspectives à court terme

6.2.1 Backtracking et exécution automatique

Pendant la phase exploratoire, les étudiants ont la possibilité de modifier et de réexécuter les cellules qu'ils ont déjà exécutées (cellules oranges avec MOON), généralement pour corriger leur code ou explorer des alternatives. Cependant, cette flexibilité peut poser des problèmes de cohérence du notebook (Figure 5.2). Pour remédier à cela, MOON propose un menu *backtracking* (retour automatique) qui guide les étudiants pour reprendre le scénario prévu par l'enseignant. Lorsqu'une cellule orange `Ci` est exécutée,

MOON ramène les étudiants à la dernière exécution prévue par le script de cette cellule, ajustant les couleurs des autres cellules en conséquence. Prenons l'exemple d'un exercice sur le tri sélection (Figure 6.1a) avec le scénario linéaire suivant (`C1 C3 C5 C7 C9`). L'étudiant va devoir écrire dans la cellule `C1` la fonction `indiceMinimum` puis la tester dans la cellule `C2`. Ensuite il doit écrire les fonctions `echanger` et `triSelection` respectivement dans les cellules `C5` et `C7`. Enfin il doit tester la fonction `triSelection` dans la cellule `C9` avec l'exemple proposé. Une fois le travail achevé toutes les cellules sont oranges (nous n'avons pas représenté ce notebook). Malheureusement le test de la cellule `C9` ne donne pas la sortie attendue. Le tableau n'est pas trié correctement (Figure 6.1b). L'étudiant décide alors de modifier le code de la fonction `indiceMinimum` dans la cellule `C1`



(a) Etat du notebook initial après avoir démarré le script dans MOON

(b) État du notebook après exécution des cellules `C1 C3 C5 C7 C9 C1`

FIGURE 6.1 – Extrait d'un notebook éducatif contenant un scénario linéaire dont le script est (`C1 C3 C5 C7 C9`) avec le *backtracking* activé.

Après une nouvelle exécution de la cellule `C1`, nous observons (Figure 6.1b) que la cellule `C3` est repassée en vert. Bien que les couleurs des cellules soient mises à jour, l'état de la mémoire du notebook reste inchangé, ce qui nécessite une nouvelle exécution des cellules dépendantes de `C1`. Dans cet exemple il y en a deux `C3` et `C9`. Normalement le script embarqué dans le notebook guide les étudiants vers ces cellules (la cellule `C3` est bien verte), c'est tout l'intérêt du *backtracking*.

Une perspective intéressante serait d'exécuter automatiquement les cellules de code dépendantes de `C1`, ce que MOON n'est pas capable de faire dans sa version actuelle. Pour atteindre cet objectif, une approche envisageable serait de réexécuter toutes les cellules de code prévues par le script de l'enseignant et qui ont déjà été exécutées par l'étudiant. Dans notre exemple (Figure 6.1b) depuis `C1` nous aurions à exécuter (`C3 C5 C7 C9`). Mais dans ce cas il est fort possible que nous exécutions des cellules qui n'ont aucune dépendance avec `C1`. Par exemple la cellule `C5` implémente correctement la fonction `echanger` qui est indépendante de la fonction `indiceMinimum`. Un autre facteur à prendre en compte est l'ajout par l'étudiant de cellules de code supplémentaires au notebook lui permettant ainsi d'explorer d'autres solutions possibles. Ces cellules n'appartiennent pas au script et ne seront donc pas réexécutées automatiquement.

À relativement court terme nous pourrions donc améliorer les fonctionnalités du *back-*

traking en proposant une exécution automatique des cellules de code dépendantes de la réexécution d'une cellule orange `Ci` en se posant les questions suivantes :

- Peut-on éviter de réexécuter des cellules de code qui n'ont aucune dépendance transitive avec la cellule de code `Ci` de manière indépendante du langage de programmation ?
- Doit-on exécuter les cellules de code ajoutées par l'étudiant ?
- Si oui, comment les intégrer au script enseignant ?

6.2.2 Unification de NORM et MOON

L'association des outils NORM et MOON offre une perspective intéressante. Elle souligne l'importance de la reproductibilité des résultats, qui suscite un réel intérêt chez les enseignants pour comprendre le processus par lequel les étudiants parviennent à leurs résultats. Il est crucial de savoir quels ont été les étapes successives qui ont conduit à la création du notebook final. En outre, il convient de se questionner sur la présence de toutes ces étapes dans le notebook, au-delà du simple suivi du chemin parcouru.

Considérons le scénario linéaire suivant (`C1, C2, C1, C3`) à titre d'exemple. Dans ce cas, l'utilisateur modifie et exécute d'abord `C1`, puis fait de même avec `C2`. Ensuite, le scénario requiert de l'utilisateur de modifier à nouveau `C1` avant d'exécuter `C3`. Si la sortie de la cellule `C2` dépend de l'exécution de la cellule `C1`, alors le notebook ne sera plus reproductible. En effet, après la modification de `C1`, la cellule `C2` ne sera pas mise à jour. Dans le cas d'un tel scénario, il est impossible d'étudier le contenu de la cellule `C1` après les exécutions intermédiaires de celle-ci, et l'évaluation devra en tenir compte.

Une première idée serait de conserver une trace exhaustive des différents états du notebook et de leur contenu. Cela permettrait à l'enseignant de reconstituer le notebook de l'étudiant dans son intégralité. Malheureusement, se contenter de collecter les états et les contenus du notebook ne suffit pas. Des recherches antérieures montrent que dans une utilisation réelle, cette approche produit un grand nombre de versions avec des dépendances complexes à analyser [14, 16]. Plusieurs pistes encouragentes travaillant sur l'historique des états d'un notebook existent et permettent des approches envisageables [5, 10, 11]. De futurs travaux pourraient essayer de coupler ces approches avec le script enseignant embarqué dans le notebook pour tenter de définir ce que pourrait être une version pertinente de l'exécution non linéaire d'un notebook étudiant.

6.3 Perspectives à plus long terme

6.3.1 Définir un DSL pour les scripts enseignants

Dans la version actuelle de MOON, l'édition du script correspondant au scénario de l'activité peut s'avérer complexe et fastidieuse en fonction du nombre de cellules contenues dans le notebook. De plus, il n'existe par exemple aucun moyen de situer l'erreur dans le script en cas de problème au démarrage de celui-ci. Afin de faciliter le travail de saisie et de correction du script, un des axes de notre futur recherche pourrait être de se concentrer sur le développement d'un langage dédié au domaine (DSL) pour la description et à la mise en œuvre des scénarios d'activités pédagogiques dans les notebooks. Dans un premier temps, il est important de s'attacher à identifier précisément les besoins des enseignants en matière de conception d'activités pédagogiques interactives.

Nous devront rencontrer des enseignants et discuter avec eux pour comprendre les défis importants lors de la création et de la gestion et/ou de la mise à jour des scénarios contenus dans les notebooks éducatifs. Sur la base des résultats de cette évaluation, nous nous efforcerons de concevoir un DSL intuitif et flexible, capable de répondre aux besoins identifiés. Ce langage spécifique sera conçu pour permettre aux enseignants de décrire efficacement leurs scénarios d'activités pédagogiques, en tenant compte de la diversité des pratiques pédagogiques et des sujets enseignés. Parallèlement au développement du DSL, nous mettrons en œuvre un framework ou une bibliothèque logicielle permettant l'interprétation et l'exécution des scénarios d'activités décrits. Cette étape de l'implémentation est essentielle pour rendre le DSL opérationnel et facilement utilisable dans un contexte éducatif réel. Une fois le DSL et l'outil associé développés, nous procéderons à une évaluation approfondie de leur utilité et de leur efficacité. Cette évaluation se fera en étroite collaboration avec des enseignants et des apprenants, à travers des études pilotes visant à tester la pertinence du DSL dans différents contextes pédagogiques. Enfin, sur la base des retours d'expérience obtenus lors des évaluations, nous formulerons des recommandations pour l'évolution future du DSL et de l'outil.

6.3.2 Tableau de bord pour l'enseignant

De nombreuses études indiquent qu'il y a eu ces dernières années des progrès significatifs dans le développement d'applications appelées *tableau de bord*. Ils permettent de capturer des données issues des salles de classe et de fournir de l'information aux enseignants. Plusieurs prototypes illustrent le potentiel et les opportunités que ces applications peuvent offrir [1, 3, 4, 6, 17, 18]. Cependant, la question de savoir quelles sont les informations pertinentes à afficher pour différents acteurs et comment ces informations devraient être collectées et présentées reste largement non résolue [15, 19].

Doter l'enseignant d'un système de surpervision automatique en temps réel de la progression des étudiants dans un notebook éducatif nous semble donc une piste intéressante à suivre. Cet outil, sous forme de tableau de bord pour l'enseignant, irait au-delà de la simple collecte de données comme le temps passé sur une tâche ou le nombre de clics. Il proposerait également une analyse du contenu et de la signification des actions de chaque étudiant, tout en tenant compte du scénario de l'activité contenue dans le notebook. Grâce à cet outil, l'enseignant pourrait suivre en temps réel la progression des étudiants dans le contexte de l'ensemble du notebook éducatif, ce qui lui permettrait d'obtenir des retours sur les défis ou les difficultés rencontrés par chacun d'eux. Ces informations pourraient ensuite être utilisées pour adapter l'enseignement de manière à mieux répondre aux besoins individuels des étudiants et à améliorer l'efficacité des cours. De plus, cette supervision automatique pourrait aider les enseignants à identifier des tendances ou des problèmes récurrents, facilitant ainsi l'ajustement des méthodes pédagogiques pour articuler les contenus du notebook éducatif.

6.3.3 Approches pédagogiques pour les mécanismes d'interaction dans les notebooks

Un outils d'assistance comme MOON se doit de répondre à plusieurs objectifs pour être résolument moderne comme la navigation, la conception, la disposition et la présentation de l'information, l'interactivité, le contenu, ou encore le feedback [2, 12, 9]. Dans cette thèse nous avons travaillé sur l'assistance à la navigation dans le scénario d'une activité contenue dans un notebook et sur la reproductibilité des résultats dans le cas d'ac-

tivités construites sur la base de scénarios linéaires.

Une reflexion importante doit maintenant être menée pour atteindre des objectifs plus ambitieux comme par exemple la disposition et la présentation de l'information dans un notebook éducatif. Des études récentes ont proposé de réoganiser l'agencement des cellules du notebook [8, 20, 13] afin de travailler sur la disposition de l'information en fournissant une vue simplifiée d'un notebook Jupyter, mettant en avant les résultats les plus importants tout en permettant une exploration interactive et libre de l'ensemble du notebook. Ces différentes approches illustrent une tentative de répondre à un défi lié à un aspect trop linéaire des notebooks. L'accent est mis sur la clarté et la facilité d'utilisation des notebooks pour les utilisateurs. Si par exemple un des exercices de l'activité contenu dans le notebook fait appel à plusieurs cellules de code qui ne sont pas contigues, nous pouvons imaginer un regroupement automatique le temps de l'exercice. Un autre exemple pourrait être celui d'un étudiant qui travaille sur une activité contenant dans la première cellule du notebook des informations textuelles qu'il doit consulter régulièrement. À mesure que l'étudiant avance dans l'activité, la première cellule finit par disparaître de l'écran en raison du défilement. Si l'étudiant doit relire les informations contenues dans la première cellule, il doit donc systématiquement remonter au début de son notebook perdant ainsi la position de la cellule active. En s'inspirant de Sticky-Land [20] nous pourrions envisager de faire apparaître temporairement les informations à relire dans une fenêtre flottante au niveau de la cellule de code dans laquelle l'étudiant travaille. Il serait donc judicieux d'explorer en parallèle du travail sur l'information, des techniques de navigation adaptatives, qui s'ajustent en fonction des besoins, des progrès de l'étudiant et de ses préférences individuelles, afin d'optimiser l'efficacité de l'apprentissage [7].

6.4 Références

[1] Dana ALZOUBI et al. "TEACHActive feedback dashboard: Using automated classroom analytics to visualize pedagogical strategies at a glance". In : *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, p. 1-6.

[2] Sophie BURY et Joanne OUD. "Usability testing of an online information literacy tutorial". In : *Reference services review* 33.1 (2005), p. 54-65.

[3] David CAMACHO et Maria D R-MORENO. "Towards an automatic monitoring for higher education learning design". In : *International Journal of Metadata, Semantics and Ontologies* 2.1 (2007), p. 1-10.

[4] Viviane GUÉRAUD et Jean-Michel CAGNAT. "Automatic semantic activity monitoring of distance learners guided by pedagogical scenarios". In : *European Conference on Technology Enhanced Learning*. Springer. 2006, p. 476-481.

[5] Andrew HEAD et al. "Managing Messes in Computational Notebooks". In : *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*. Sous la dir. de Stephen A. BREWSTER et al. ACM, 2019, p. 270. DOI : 10.1145/3290605.3300500. URL : https://doi.org/10.1145/3290605.3300500.

[6] H HOPPE et al. "A web-based tutoring tool with mining facilities to improve learning and teaching". In : *Artificial intelligence in education: Shaping the future of learning through intelligent technologies* 97.201 (2003), p. 49.

[7] Tumaini KABUDI, Ilias PAPPAS et Dag Håkon OLSEN. "AI-enabled adaptive learning systems: A systematic mapping of the literature". In : *Computers and Education: Artificial Intelligence* 2 (2021), p. 100017.

[8] Daye KANG et al. "ToonNote: Improving Communication in Computational Notebooks Using Interactive Data Comics". In : *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama, Japan, May 8-13, 2021*. Sous la dir. d'Yoshifumi KITAMURA et al. ACM, 2021, 727:1-727:14. DOI : 10.1145/3411764.3445434. URL : https://doi.org/10.1145/3411764.3445434.

[9] Caitlin KELLEHER et Randy PAUSCH. "Stencils-based tutorials: design and evaluation". In : *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2005, p. 541-550.

[10] Mary Beth KERY et Brad A MYERS. "Interactions for untangling messy history in a computational notebook". In : *2018 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE. 2018, p. 147-155.

[11] Mary Beth KERY et al. "Towards effective foraging by data scientists to find past analysis choices". In : *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, p. 1-13.

[12] Kevin KNABE. "Apple guide: a case study in user-aided design of online help". In : *Conference companion on Human factors in computing systems*. 1995, p. 286-287.

[13] Haotian LI et al. "Notable: On-the-fly Assistant for Data Storytelling in Computational Notebooks". In : *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*. Sous la dir. d'Albrecht SCHMIDT et al. ACM, 2023, 173:1-173:16. DOI : 10.1145/3544548.3580965. URL : https://doi.org/10.1145/3544548.3580965.

[14] João Felipe Nicolaci PIMENTEL et al. “Collecting and Analyzing Provenance on Interactive Notebooks: When {IPython} Meets {noWorkflow}”. In : *7th USENIX workshop on the theory and practice of provenance (TaPP 15)*. 2015.

[15] Beat A SCHWENDIMANN et al. “Perceiving learning at a glance: A systematic literature review of learning dashboard research”. In : *IEEE transactions on learning technologies* 10.1 (2016), p. 30-41.

[16] Sruti SRINIVASA RAGAVAN et al. “Foraging among an overabundance of similar variants”. In : *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, p. 3509-3521.

[17] Ahmed TLILI et al. “A smart collaborative educational game with learning analytics to support english vocabulary teaching”. In : *IJIMAI* 6.6 (2021), p. 215-224.

[18] Katrien VERBERT et al. “Learning dashboards: an overview and future research opportunities”. In : *Personal and Ubiquitous Computing* 18 (2014), p. 1499-1514.

[19] Han WANG et al. “The Impact of Dashboard Feedback Type on Learning Effectiveness, Focusing on Learner Differences”. In : *Sustainability* 15.5 (2023), p. 4474.

[20] Zijie J. WANG, Katie DAI et W. Keith EDWARDS. “StickyLand: Breaking the Linear Presentation of Computational Notebooks”. In : *CHI ’22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022, Extended Abstracts*. Sous la dir. de Simone D. J. BARBOSA et al. ACM, 2022, 269:1-269:7. DOI : 10.1145/3491101.3519653. URL : https://doi.org/10.1145/3491101.3519653.