DEMC Journal

hors-serie n° 1 | 2026
Les nouvelles formes éditoriales autour du numérique et des données

Programming Historian en français

Les défis de la fabrication d'une revue francophone sur les humanités numériques

Alexandre WAUTHIER *Ingénieur d'études*
Service Accompagnement des projets et science ouverte
UAR 2011
Campus Condorcet
Daphné MATHELIER

Matthias GILLE LEVENSON

Marie FLESCH

Édition électronique :
URL : https://demc-journal.org/articles/hors-serie-1/3579-programming-historian-en-francais
Date de publication : 06/07/2026

Pour **citer cette publication** : WAUTHIER, A., MATHELIER, D., GILLE LEVENSON, M., FLESCH, M. (2026) Programming Historian en français. *DEMC Journal*, (hors-serie n°1). https://doi.org/10.34745/

Cet article interroge la place de la revue interdisciplinaire. Programming Historian en français, née en 2019, dans le paysage des humanités numériques francophones. Il présente son format de publication, la « leçon », son processus éditorial innovant et transparent, ainsi que ses engagements pour la science ouverte et le logiciel libre. À partir d'une analyse quantitative des leçons publiées, il met en perspective les spécificités du chapitre francophone au sein de la famille multilingue Programming Historian, en soulignant les défis qu'il doit relever : accroître sa visibilité, renforcer sa légitimité académique, garantir la pérennité de ses contenus et affirmer ses valeurs d'inclusivité et d'accessibilité.

Mots-clés :
Édition numérique, Edition scientifique, Accessibilité, Multilinguisme, Communauté universitaire

---

Introduction

Dans le domaine des humanités numériques, largement dominé par l'anglais, le multilinguisme demeure un critère essentiel pour la recherche et l'enseignement : il permet une large diffusion des savoirs et méthodes tout en promouvant la diversité des communautés de recherche (Fiormonte, 2021). Ce constat est au cœur de la philosophie de la revue en libre accès *Programming Historian*, qui publie des présentations didactiques, appelées « leçons », évaluées par les pairs et réparties dans quatre chapitres selon la langue : anglais, espagnol, français ou portugais (Isasi *et al*., 2023). Celles-ci peuvent être traduites ou originales, c'est-à-dire, dans notre cas, rédigées en français. Elles permettent de s'initier et d'apprendre à utiliser un panel étendu d'outils et de méthodes computationnelles. L'objectif principal est de faciliter la recherche et l'enseignement en sciences humaines et sociales (SHS).

Malgré son titre, la revue *Programming Historian* ne propose pas seulement des leçons sur la programmation. De même, son lectorat n'est plus seulement constitué

d’historiens et d’historiennes : la revue s’adresse à l’ensemble des personnes qui travaillent ou étudient dans le domaine des sciences humaines et sociales. *Programming Historian en français,* qui publie majoritairement des traductions de leçons publiées en anglais dans *Programming Historian*, a une identité double. Construite sur un projet éditorial initialement anglo-saxon, qui a produit 117 leçons depuis 2008, elle s’adresse à un public francophone tout en prenant en compte les attentes et pratiques de son lectorat dans ses choix d’édition et de publication.

Bâti sur les valeurs de la science ouverte et de l’*open source*, le processus éditorial de la revue repose sur un circuit innovant qui comprend, entre autres, la rédaction des articles en Markdown par les auteur·es ou traducteur·ices, une évaluation ouverte sur GitHub, une publication sur un site statique et la transparence concernant les personnes impliquées dans ce processus.

Cet article propose d’explorer les défis auxquels est confrontée la revue, et en particulier son chapitre francophone qui, depuis 2019, a publié 41 leçons (en mai 2026). Nous proposerons ainsi une analyse du processus éditorial de la revue, une mise en perspective de la question de la traduction des leçons et une réflexion sur les enjeux majeurs de la revue : ceux de la diversité et de l’inclusivité, sous le prisme de l’accessibilité, tant des leçons que des outils numériques.

1. Le numérique : un changement de paradigme pour la recherche et la publication en SHS

Les débuts de l’apprentissage autonome de l’informatique en SHS

En 1971, Edward Shorter, historien de la médecine à l’Université de Toronto, publie *The Historian and the Computer: A Practical Guide*. Dans l’introduction de son ouvrage, il constate que « rien n’existe actuellement pour l’historien qui veut utiliser un ordinateur, mais n’a pas la moindre idée de comment procéder »[1]. Le livre guide le lecteur pas à

pas à travers le processus de conception, de mise en œuvre et d'interprétation des résultats d'une étude historique quantitative exploitant les ordinateurs centraux IBM de l'époque. Selon Adam Crymble (2021), Shorter démystifie ce que les ordinateurs sont et peuvent faire, et est le premier à replacer tout cela dans le contexte de la profession historique. Il s'agit d'un exemple précoce d'apprentissage autonome de l'informatique à des fins de recherche en sciences humaines et sociales.

Comme Shorter, d'autres historien·nes ont saisi l'opportunité d'utiliser un ordinateur dans le cadre de leur recherche et ont fait face au manque d'accompagnement pour y parvenir. En ce sens, *Programming Historian* est né de ce besoin, dans le monde académique et au sein des communautés de pratiques, de mettre en commun de ressources éducatives libres dédiées aux technologies numériques (Gille Levenson, Patat, 2022). Les humanités numériques sont précisément à l'intersection de la recherche en sciences humaines et de l'enseignement de l'informatique (Terras, 2011 ; Dacos & Mounier, 2015). Elles étudient l'impact de ces technologies sur le patrimoine culturel, les institutions de mémoire, les bibliothèques, les archives et la culture numérique. De plus, elles ne se limitent pas à l'usage d'outils informatiques génériques, mais constituent un paradigme à part entière, dans lequel un dialogue entre elles et l'informatique doit être constamment nourri (Longhi, 2019). Ce dialogue s'opère par le développement de méthodes et d'outils spécifiquement pensés pour les besoins et les regards des humanités sur le numérique.

L'émergence des humanités numériques a été favorisée, dès la fin des années 2000, par les THATCamp (*The Humanities and Technology Camp*), initiés aux États-Unis par des étudiants en histoire. En se diffusant rapidement en Europe (notamment en 2010 à Paris), ces non-conférences ont permis d'abattre les barrières institutionnelles et disciplinaires, tout en encourageant l'innovation et la co-construction du savoir (Clavert, 2010). En définitive, la singularité la plus manifeste d'un THATCamp tient à l'insaisissable : elle développe un état d'esprit, la rendant plus vivante et plus collective (Thély *et al*., 2014). Le caractère informel, léger, collaboratif, peu coûteux, à but non lucratif, spontané, opportun et disponible en ligne a fait la force des THATCamp (McGrath, 2020). Dans différents pays occidentaux, ces derniers ont favorisé la création de réseaux interdisciplinaires ouverts, la circulation des savoirs et l'expérimentation collective de nouveaux outils et méthodes.

L'évolution des pratiques éditoriales

L'essor des humanités numériques et la montée en puissance des réseaux

interdisciplinaires ont profondément transformé les pratiques éditoriales de nombreuses revues scientifiques. L'enjeu ne se limite plus à la diffusion des résultats de recherche, mais s'étend à l'ouverture des processus de production scientifique : évaluation par les pairs, révisions successives, partage des codes sources, des données brutes et des méthodologies. Cette évolution répond à une double exigence, à la fois éthique et pratique : la transparence pour renforcer la confiance dans les travaux publiés et la reproductibilité, qui permet à la communauté scientifique de s'approprier, critiquer ou réutiliser les méthodes et les résultats.

Plusieurs initiatives illustrent cette mutation vers des modèles éditoriaux ouverts et collaboratifs. Des revues se concentrent sur la pédagogie des outils numériques, en publiant des articles destinés à former les chercheurs et les chercheuses à l'usage d'outils disciplinaires. L'objectif de ces revues est double : diffuser des savoir-faire techniques tout en favorisant leur appropriation critique par la communauté. Dans le monde anglophone, des titres comme *The Journal of Open Source Software*[2] ou *The Journal of Open Source Education*[3] publient des articles présentant des logiciels et des supports éducatifs en *open source*, avec des processus de *peer review* ouverts et collaboratifs. *The Carpentries*[4] et les plateformes *OpenMethods* et *Campus* du consortium européen *DARIAH*[5] proposent, quant à elles, des articles et tutoriels en accès libre, encourageant ainsi les contributions communautaires. En langue française, *Rzine*[6] adopte une approche similaire, en présentant des méthodes d'analyse en sciences humaines et sociales mises en application avec le logiciel R, de façon reproductible et didactique. Le processus de relecture et l'article sont hébergés et déployés grâce à la forge logicielle GitHub, sur laquelle nous reviendrons dans le présent article (Pecout, 2022).

D'autres revues se spécialisent dans les articles de données, ou *data papers*, qui décrivent des jeux de données, leur contexte de collecte, leur méthodologie de traitement et leur potentiel de réutilisation. Ces publications répondent à un besoin croissant de valorisation des données brutes et des protocoles. En France, des titres comme *DEMC*[7] ou *Data & Corpus*[8] se sont engagés dans cette voie, en offrant un espace dédié à la publication de ce type d'articles en sciences humaines et sociales.

C'est dans ce paysage en pleine reconfiguration que la revue *Programming Historian*[9] contribue à rendre visible et participatif l'ensemble de la chaîne de production scientifique, depuis la soumission d'une leçon jusqu'à sa publication finale.

2. Un processus éditorial ouvert et transparent

Historique de la revue

L'histoire de *Programming Historian* commence en 2008 lorsque William J. Turkel et Alan MacEachern, professeurs au département d'Histoire de l'Université Western Ontario au Canada, publient une première ressource intitulée *The Programming Historian* sous la forme d'une série de tutoriels regroupés au sein d'un ebook au format PDF, encore consultable aujourd'hui (Turkel & MacEachern, 2011). Les leçons étaient hébergées par le *Network in Canadian History & Environment* (NiCHE) sur un site WordPress.

Initialement pensée comme une introduction à la programmation Python à destination des historien·nes et autres « humanistes » avec de faibles compétences en traitement des données numériques, cette première publication devait être complétée par un autre volume, mais c'est finalement sous la forme d'un journal en libre accès que l'aventure continue en 2012, tandis que les thématiques et l'équipe éditoriale s'élargissent (Turkel, 2008). Dès lors, les bases et la philosophie ouverte du projet sont rapidement posées : une revue scientifique en libre accès immédiat (*diamond open access*), évaluation ouverte par les pairs (*open peer review*), licences libres et données ouvertes. Le travail éditorial collaboratif est centralisé sur GitHub dès 2014. La publication, d'abord proposée sur un site indépendant utilisant le système de gestion de contenu WordPress, s'oriente rapidement vers un site statique fondé sur la technologie Jekyll[10] et hébergé sur GitHub Pages (Lincoln *et al*., 2022).

Quatre ans plus tard, le projet continue de se développer avec l'élargissement de l'équipe éditoriale : initialement anglo-saxonne, elle comprend désormais un chapitre espagnol, qui prend d'abord en charge la traduction de plusieurs leçons en anglais en 2017, puis commence à publier des leçons originales en espagnol en 2018. Face au succès de cet élargissement linguistique, les deux autres chapitres de la revue, française et portugaise, sont rapidement inaugurés : les premières leçons en français sont publiées en 2019, celles en portugais, en 2021. *Programming Historian* reste donc une seule revue, déclinée sous la forme de quatre chapitres linguistiques.

Programming Historian est, depuis ses débuts, un projet indépendant à visée internationale, non affilié à une ou plusieurs institutions et porté par des bénévoles. Depuis 2019, elle est administrée par ProgHist Limited, une organisation de droit britannique à but non lucratif, et financée par un programme de partenariats institutionnels et dons individuels. Elle emploie actuellement une responsable de publication.

Présentation de la revue

Programming Historian est une revue en libre accès, évaluée par les pairs, qui publie des articles, appelés « leçons », pour initier et former aux outils, techniques et méthodes numériques, principalement à destination des chercheurs, enseignants et étudiants en sciences humaines et sociales. Le projet vise à faciliter l'apprentissage des technologies du numérique et à rendre accessible la maîtrise de nouveaux outils numériques pour la recherche et l'enseignement.

Formellement, *Programming Historian* reprend les codes des revues scientifiques : chaque article est structuré en plusieurs parties et sous-parties, comprend un résumé, des mots-clés, une argumentation ainsi qu'une bibliographie. De plus, chaque leçon est publiée selon un processus identique à celui de l'article scientifique, avec un processus d'évaluation ouverte par les pairs (*open peer reviewing*)[11]. C'est sur le contenu des leçons que la revue se distingue : il s'agit de textes à vocation pédagogique, qui mettent en scène une question de sciences humaines pour présenter et décrire des méthodologies informatiques. De même, une leçon peut ne pas être originale, mais être la traduction d'une leçon rédigée dans une autre langue : la qualité didactique d'une leçon prime sur le caractère inédit du savoir transmis. Il s'agit ici de la plus grande différence entre *Programming Historian* et les revues scientifiques traditionnelles.

Les leçons publiées sur *Programming Historian* forment un objet singulier, faisant la particularité de la revue. De fait, les leçons vont au-delà du simple tutoriel. Elles s'appuient sur des données authentiques, sont étayées par des réflexions méthodologiques et évaluées par les pairs. En mai 2026, la revue comptait 283 leçons – dont 41 pour le chapitre français – organisées en fonction de thématiques (Python, manipulation des données, gestion des données, etc.) et des cinq principales actions du processus de recherche : acquérir, transformer, analyser, présenter, préserver.

Le lectorat recherché va au-delà d'un public historien : il s'agit de viser plus largement des personnes intéressées par le traitement informatique et computationnel d'objets en sciences sociales (histoire, histoire littéraire, linguistique, philologie, géographie, archéologie, démographie, etc.). Un grand nombre de leçons présentent des méthodologies facilement applicables à d'autres objets que les exemples détaillés. Par exemple, la leçon traduite « Comprendre les expressions régulières » prend comme exemple des statistiques de mortalité et de morbidité aux États-Unis au xxe siècle pour dérouler les apprentissages. Si la leçon est originellement produite dans une optique historique, les savoirs qu'elle contient sont applicables à tous les champs scientifiques ayant pour objet des données textuelles.

Un processus éditorial transparent sur GitHub

Le processus éditorial de la revue est très richement documenté dans les consignes aux auteur·es[12], évaluateur·trices, traducteur·trices et rédacteur·trices, ainsi que dans un wiki au sein du dépôt GitHub *ph-submissions* de *Programming Historian*[13].

La revue est gérée sur GitHub à l'aide de deux répertoires principaux : le dépôt *ph-submissions* pour la préparation des leçons et le dépôt *jekyll* pour la publication et la maintenance des leçons.

De manière synthétique, la chaîne éditoriale (*workflow*) de *Programming Historian* comprend sept grandes étapes :

- Proposition (*proposal*) d'une leçon originale ou d'une traduction par la rédaction en chef.
- Réception de la leçon ou de la traduction par l'équipe éditoriale du chapitre concerné.
- Première lecture de la leçon par le rédacteur ou la rédactrice à qui la leçon a été attribuée (*initial edit*). Il s'agit principalement ici d'évaluer si elle est conforme aux principes d'ouverture, d'inclusivité et de durabilité de la revue, ainsi que son niveau de difficulté.
- Première révision de la leçon par l'auteur·e (*first revision*).

- Évaluation ouverte (*open peer review*) de la leçon à l'aide d'un ticket (*issue*) dans GitHub.

- Deuxième révision de la leçon par l'auteur·e (*second revision*).

- Réglages techniques pour garantir la pérennité et l'accessibilité de la leçon (*sustainability*, *accessibility*).

- Mise en ligne de la leçon sur le site de *Programming Historian* (*publication*).

Ces étapes structurent la revue et définissent le rôle de chaque contributeur ou contributrice dans le processus éditorial. Elles ont été intégrées à la revue francophone, à l'instar de ses sœurs hispanophones et lusophones.

Le choix d'une évaluation ouverte par les pairs : visibilité et fluidité des échanges

L'ensemble des chapitres linguistiques de *Programming Historian* fonctionne sur le modèle de l'évaluation ouverte par les pairs, expression qui désigne un ensemble de méthodes d'évaluation visant à rendre publiques les évaluations des articles et les différentes étapes et versions d'une production scientifique (Ross-Hellauer, 2017). Ces systèmes d'évaluation peuvent avoir lieu sur des infrastructures conçues *ad hoc*, comme celle de F1000Research.

Programming Historian a décidé d'adopter cette approche en quittant Wordpress pour GitHub en 2014 (McDaniel, 2014). En effet, GitHub permet – indirectement, car il ne s'agit pas d'une plateforme de publication en soi – de mettre en place un système d'évaluation ouvert, par le biais des tickets (*issues*), qui servent en temps normal à soumettre des suggestions d'amélioration d'un projet ou de correction de bugs[14]. Dans le cas qui nous intéresse, c'est bien des commentaires sur une leçon soumise qui sont publiés dans un ticket et qui permettent de construire un dialogue entre des évaluateur·ices et l'auteur·e afin de proposer une leçon de qualité. L'ouverture de ce mode d'évaluation porte ainsi non seulement sur les commentaires laissés par les évaluateur·trices, mais aussi sur les modifications apportées à la leçon en cours de construction : pour retracer l'histoire d'une leçon, il suffit de regarder les différentes modifications apportées via le système de révision (*commits*).

Cette méthode globale a plusieurs mérites, dont celui de rendre public le travail d'édition et de correction des leçons soumises. Du point de vue relationnel, elle permet d'éviter des critiques trop abruptes ou violentes et favorise la modération de la part des évaluateur·trices[15].

Particularités du chapitre francophone : tentatives d'inclusivité et de diversité

L'équipe du volet francophone de la revue s'est constituée à son lancement en 2019 autour d'une première rédactrice en chef, Sofia Papastamkou, docteure en histoire puis s'est étoffée et régulièrement renouvelée, avec notamment un appel à participation en 2023 et un changement de direction la même année.

Dès son lancement, *Programming Historian en français* a fait le choix d'utiliser l'écriture inclusive, en suivant les principes arrêtés par l'Office québécois de la langue française (Papastamkou, 2019), afin de visibiliser les femmes et de tenter de réduire les barrières liées au genre dans la recherche et l'enseignement des humanités numériques. Cela entre en ligne de compte sur la manière dont les leçons sont soit traduites, soit proposées spontanément par des auteur·ices francophones.

D'autre part, *Programming Historian en français* cherche à s'adresser à un public francophone le plus large possible en cherchant, dès ses débuts, à diversifier la composition géographique de son comité de rédaction et en tissant notamment des liens avec la communauté québécoise des humanités numériques. Ainsi, lors de sa création, en 2019, l'équipe de la revue comprenait deux membres dont les activités n'étaient pas localisées en France (Papastamkou, 2018).

Actuellement, les huit rédacteur·trices du chapitre francophone de *Programming Historian* sont tous membres du milieu universitaire – français et québécois –, plus particulièrement dans le domaine des sciences humaines et sociales, et ont divers statuts : doctorant·e, postdoctorant·e, enseignant·e chercheur·e ou ingénieur·e. La participation à l'aventure *Programming Historian* est basée sur le volontariat, bien que cette activité puisse être soutenue institutionnellement, en allouant du temps de travail à cette activité.

Programming Historian en français, comme toutes les autres branches et sur le modèle de la branche anglophone, prend le soin de nommer chaque personne ayant contribué à une leçon lors de l'une des étapes du processus d'édition. Dans l'en-tête de chaque leçon, on trouve ainsi la personne ayant assuré le suivi éditorial et celles qui l'ont évaluée. Pour les traductions, sont ajouté·es : l'auteur·e de la traduction, la personne chargée du suivi de traduction et les évaluateur·trices de celle-ci.

Les personnes gravitant autour de la revue échappent donc généralement au profil des « petites mains », définies par Françoise Waquet comme « celles et ceux qui sont derrière ces multiples produits numériques que sont les catalogues de bibliothèques en ligne, les sites Web présentant des institutions et des projets de recherche, les bases de données, les portails d'archives ouvertes », en les nommant « travailleurs du clic » (Waquet, 2022, p. 35). L'auteure va même plus loin en parlant de « non-personnes invisibles » et autres « petites mains effacées » notamment composées de personnels statutaires, mais aussi d'un précariat de contractuels et de vacataires. *Programming Historian en français* a pu compter, à ses débuts, sur l'appui d'étudiant·es pour traduire et relire des leçons issues du chapitre anglophone, par exemple dans le cadre d'un cours d'anglais du master « Technologies numériques appliquées à l'histoire » dispensé à l'École nationale des chartes (Papastamkou, 2019), mais toutes ces contributions ont été assumées et affichées au début des leçons concernées.

En ce qui concerne les réseaux autour desquelles elle gravite, la revue francophone peut compter sur les listes de diffusion professionnelles que les membres de l'équipe ont l'habitude de consulter, notamment la liste DH d'Humanistica (Association francophone des humanités numériques) et celle du réseau français d'ingénieur·e·s Mate SHS (Méthodes, analyses, terrains, enquêtes en sciences humaines et sociales). Ainsi, des appels à traductions ou à leçons originales sont diffusés sur ces listes, mais aussi publiés sur le site de la revue (Flesch & Chevrie, 2024). Ces appels à traduction correspondent, le plus souvent et implicitement, à une traduction de l'anglais vers le français. Toutefois, en février 2025, un appel à traduction de l'espagnol vers le français a été publié afin de pallier le manque de leçons francophones et anglophones sur le langage de balisage XML-TEI[16], en comparaison à nos homologues hispanophones (Flesch & Chevrie, 2025). Plus récemment, en février 2026, le chapitre français a lancé un appel général à contributions, accueillant à la fois propositions spontanées de leçons et traductions suggérées de leçons issues des autres chapitres de la revue (Flesch & Hawes, 2026).

3. Publication d’une leçon dans une revue multilingue : traductions et temporalités différenciées

En mai 2026, le chapitre francophone compte 41 leçons, dont 35 traductions. 7 ans s’écoulent en moyenne entre une leçon et sa traduction. Le délai minimal est d’un an (relativement incompressible étant donné le temps de l’édition) et le délai maximal de 13 ans, ce qui correspond à la traduction des premières leçons de *Programming Historian* qui a débuté autour de 2012. Il y a donc un grand nombre d’années entre la date de publication d’une leçon et la date de sa traduction, ce qui explique la nécessité d’apporter au contenu des leçons un certain nombre de modifications, au moins du point de vue technique. Cet écart explique en partie que la traduction ne soit pas qu’un simple portage d’une leçon dans une autre langue, mais souvent une adaptation à la langue cible et aux évolutions techniques qui ont eu lieu depuis la rédaction de la leçon originale, comme nous allons le voir dans la section qui suit.

La traduction entre adaptation et réécriture

La traduction des leçons n’est donc pas un simple calque des leçons existantes. Elle peut impliquer une adaptation des contenus au public visé (francophone dans le cas qui nous intéresse) et suppose souvent l’adaptation aux changements technologiques, voire des amendements pour améliorer la leçon.

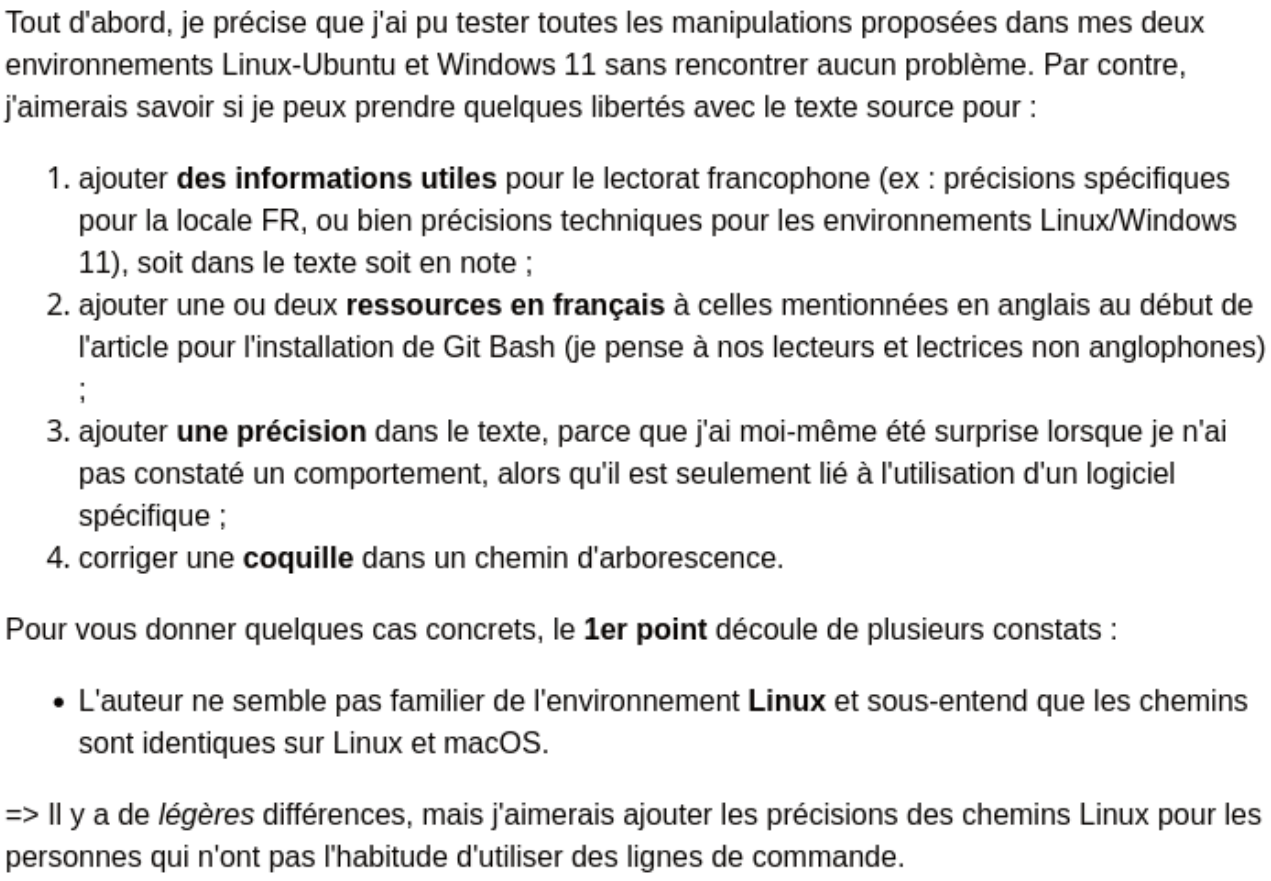
Premièrement, la traduction requiert une explicitation des référents culturels pour les rendre compréhensibles pour un public qui ne les connaît pas[17]. Pour les leçons portant sur des objets textuels, pour lesquels la langue est donc particulièrement importante, il est possible d’envisager un changement complet des exemples choisis. C’est le cas pour la traduction en cours de suivi de l’outil de publication statique CETEIcean (Calarco, Riande, 2025)[18] : l’exemple source est une chronique qui prend pour objet une partie du Cône Sud et du Río de la Plata, *La Argentina Manuscrita*, de Ruy Díaz de Guzmán (1559-1629). La traduction de cette leçon prend pour objet le roman *Splendeurs et misères des courtisanes* de Balzac. Toutes les leçons ne voient cependant pas leurs exemples adaptés : il s’agit alors plutôt de privilégier l’explicitation des exemples quand ils ne le sont pas dans la langue originale.

L’adaptation des leçons est rendue possible par le caractère reproductible des exemples et des expériences caractéristiques de chaque leçon soumise. N’importe qui doit être en mesure de reproduire les manipulations et traitements montrés dans la leçon. Ainsi, la modification des exemples n’a pas – en théorie – d’impact sur la réplicabilité des expériences de la leçon : la structure des leçons et la philosophie de l’ouverture qui les sous-tend favorisent en elles-mêmes leur traductibilité.

La traduction d’une leçon peut être considérée comme une forme de seconde relecture par les pairs, d’abord prise en charge par la personne effectuant la traduction. En effet, des erreurs peuvent avoir été publiées malgré la première relecture ; par ailleurs, les sensibilités diffèrent logiquement entre personnes et la personne en charge de la traduction peut considérer comme important de réorienter ou de reformuler un paragraphe considéré comme peu clair.

Un premier exemple d’adaptation porte sur l’évolution technologique qui demande souvent des modifications dans la traduction, modifications qui seront souvent reportées dans la leçon originale. Ainsi, dans la leçon mentionnée plus haut sur CETEIcean, la traductrice a pris l’initiative de changer l’éditeur de code, car l’éditeur utilisé dans la leçon originale était devenu obsolète.

Un second exemple concerne des modifications apportées au contenu des leçons pour le rendre plus clair pour le lectorat. Nous proposons ci-dessous une capture d’un commentaire à l’intention d’un éditeur, publié sur GitHub par une traductrice :



Tout d'abord, je précise que j'ai pu tester toutes les manipulations proposées dans mes deux environnements Linux-Ubuntu et Windows 11 sans rencontrer aucun problème. Par contre, j'aimerais savoir si je peux prendre quelques libertés avec le texte source pour :

1. ajouter **des informations utiles** pour le lectorat francophone (ex : précisions spécifiques pour la locale FR, ou bien précisions techniques pour les environnements Linux/Windows 11), soit dans le texte soit en note ;
2. ajouter une ou deux **ressources en français** à celles mentionnées en anglais au début de l'article pour l'installation de Git Bash (je pense à nos lecteurs et lectrices non anglophones) ;
3. ajouter **une précision** dans le texte, parce que j'ai moi-même été surprise lorsque je n'ai pas constaté un comportement, alors qu'il est seulement lié à l'utilisation d'un logiciel spécifique ;
4. corriger une **coquille** dans un chemin d'arborescence.

Pour vous donner quelques cas concrets, le **1er point** découle de plusieurs constats :

- L'auteur ne semble pas familier de l'environnement **Linux** et sous-entend que les chemins sont identiques sur Linux et macOS.

=> Il y a de *légères* différences, mais j'aimerais ajouter les précisions des chemins Linux pour les personnes qui n'ont pas l'habitude d'utiliser des lignes de commande.

Figure 1 – Extrait d'une correspondance entre une traductrice et un évaluateur. La traductrice suggère des modifications qui portent à la fois sur l'adaptation à un public francophone et sur l'amélioration de la leçon.

Dans ce message, la traductrice suggère à la fois des ajouts pour améliorer la lisibilité et la compréhension du texte pour le public francophone, et des modifications du texte source qui contient des erreurs ou inexactitudes. Une traduction de leçon n'est donc pas la seule adaptation d'un texte, il s'agit bien d'améliorer la version originale en la corrigeant ou la complétant éventuellement. Dans ce sens, la leçon repasse au crible de l'évaluation et le traducteur fait office de nouvel évaluateur.

4. Une politique d'ouverture, de diversité et d'inclusivité

Accès ouvert et multilinguisme

Programming Historian souhaite rendre ses leçons les plus ouvertes et accessibles possibles. Le choix d'avoir recours à une licence CC BY (*Creative Commons Attribution*) permet de réutiliser les contenus de la revue, de les partager, de les modifier et de les diffuser, y compris à des fins commerciales, à condition de créditer l'auteur·e original·e. Le processus de publication, décrit en première partie, se veut également le plus ouvert possible, notamment à travers le déploiement du site et le suivi des leçons, intégralement déposés sur GitHub. Cela permet d'exposer les dimensions sociale et technique de l'élaboration d'une leçon de façon transparente, ouverte et documentée (Gille Levenson *et al*., 2022). En un sens, l'ouverture du suivi contribue à lutter contre les éventuelles discriminations ou inégalités d'accès. C'est dans ce suivi qu'apparaît systématiquement la politique de la revue contre le harcèlement, également publiée dans les consignes aux rédacteur·trices.

Par ailleurs, dans la même optique d'accessibilité, les outils et logiciels présentés dans chaque leçon publiée sont *open source* et retenus par le comité de rédaction pour leur pérennité. Les leçons doivent ainsi pouvoir être reproduites sans contrainte par le lectorat (enjeu de liberté) et suivies le plus longtemps possible (enjeu de pérennité). Quelques exceptions peuvent cependant être notées, comme la leçon « La reconnaissance automatique d'écriture à l'épreuve des langues peu dotées » (Vidal-Gorène, 2023). Cette leçon s'appuie sur une infrastructure technique d'HTR[19] fermée et propriétaire, mais il a été jugé que la méthodologie présentée, très généraliste, pouvait être utile et servir hors de l'utilisation de la plateforme promue par son auteur. De même, si l'attribution d'un identifiant pérenne permet de citer une leçon par son DOI, à la manière d'un article scientifique, son contenu didactique n'est pas nécessairement figé, puisqu'il est modifié au gré de l'obsolescence des technologies proposées.

Par son multilinguisme, *Programming Historian* a fait de la traduction l'un de ses piliers en termes d'ouverture et d'efficacité didactique, assurant un nombre de visites du site toujours plus grand (estimé à 1 million par an en 2023 ; Crymble & Im, 2023).

Par ailleurs, d'un point de vue technique, le multilinguisme a été intégré au site statique déployé sur GitHub à travers une configuration avancée de Jekyll (Lincoln, 2020), permettant d'établir une table des traductions de chaque leçon[20]. À partir de cette normalisation de la structure des leçons en quatre langues, il a été possible d'aligner à la phrase et de manière automatisée les leçons du catalogue de *Programming Historian* afin d'étudier les modalités de traduction des termes techniques et dans l'optique de

créer un vocabulaire technique multilingue (Gille Levenson *et al.*, 2024).

Faire communauté

L'évaluation ouverte par les pairs proposée par *Programming Historian* est une manière de construire une communauté de pratique ouverte, qui se veut inclusive et diverse, en proposant notamment plusieurs niveaux de difficulté au gré du contenu des leçons. De fait, pour que les leçons puissent demeurer académiques et accessibles au plus grand nombre, la revue cherche à préserver une ouverture à un lectorat composé de non-spécialistes, tout en gardant une exigence rédactionnelle de recherche. En ce sens, trois niveaux de difficulté ont été établis pour chaque leçon publiée, en fonction des opérations à réaliser : facile, moyen et avancé. Ces critères constituent une mise en garde, mais instaurent aussi une gradation de difficulté dans les leçons portant sur un outil ou un sujet similaire. Par exemple, la leçon « Installation de bibliothèques Python avec pip » (Gibbs 2013) a été jugée d'un niveau facile, car elle ne supposait pas de manipulation technique ou de compréhension de concept complexe, et pouvait être suivie par une personne débutante. La leçon « Décomptes d'occurrences de mots en Python » (Turkel et Crymble 2012) est considérée comme moyenne, car elle présente des structures de données propres à Python et requiert pour le lectorat un grand nombre d'opérations différentes. « La reconnaissance automatique d'écriture à l'épreuve des langues peu dotées » (Vidal-Gorène 2023) est considérée comme une leçon complexe, car elle introduit à la fois des concepts en apprentissage supervisé (cycle de production des données, notion d'affinage des modèles) et des notions techniques importantes pour le traitement automatique du texte (question de la normalisation Unicode et de son impact sur l'apprentissage).

Par ailleurs, dans ses rangs, la revue tente d'assurer une parité de genre et une pluralité de statut dans les comités de rédaction ainsi que dans le choix des relecteurs·trices. Une analyse du profil des personnes qui ont contribué à la revue francophone depuis sa naissance en tant que traducteur·trices ou auteur·es permet de dresser un portrait sociodémographique de la communauté, et de savoir si les engagements de la revue se traduisent par une réelle diversité. En termes de genre, en mai 2026, la distribution hommes/femmes (analysée à partir des données dont nous disposons) est relativement égalitaire (14 femmes et 19 hommes). En ce qui concerne les fonctions des contributeur·ices, les leçons ont principalement été traduites ou écrites par des ingénieur·es d'études ou de recherche (9 personnes) et des chercheur·es en début de carrière (9 doctorant·es et postdoctorant·es). Six personnes sont issues du monde des bibliothèques, des musées et de la gestion documentaire, tandis que trois sont chercheur·es. Une seule enseignante-chercheuse a contribué à la revue, en tant que traductrice. Enfin, un constat se dessine : la vaste majorité des contributeur·ices

travaillent en France et au Québec, dans une moindre mesure. L'absence de personnes issues d'Afrique francophone révèle la difficulté à ouvrir davantage la communauté.

Programming Historian continue donc à rechercher des moyens d'élargir son audience et sa communauté, au-delà de la France et du Québec, et à atteindre les locuteur·trices francophones des pays du Sud. Si le chapitre hispanophone a su atteindre un lectorat sud-américain, cette extension linguistique est en cours au sein du chapitre francophone. *Programming Historian en français* a tout intérêt à renforcer la diversité des pratiques et problématiques de recherche en élargissant sa communauté à des pays jusqu'à présent moins représentés à travers les personnes impliquées dans la rédaction de leçons. Cette volonté est partagée dans le milieu des humanités numériques, notamment à travers l'association francophone des humanités numériques Humanistica, qui a organisé son colloque annuel de 2024 à Meknès (Maroc)[21], puis à Dakar (Sénégal)[22] l'année suivante.

Traduire une leçon… et ses exemples ?

Le chapitre anglophone de *Programming Historian* comprend actuellement le plus de leçons (119 en mai 2026) susceptibles d'être traduites. Certaines sont en ligne depuis les débuts de la revue, en 2008, et elles ont toutes en commun une licence de réutilisation CC-BY-4.0 autorisant la traduction. Toutefois, certains exemples de jeux de données ou de mise en contexte comptent parmi les éléments plus complexes à traduire. En effet, les exemples choisis par la leçon d'origine peuvent parfois sembler lointains. C'est notamment le cas d'une série de leçons écrites en anglais dans les premiers temps de la revue et qui s'appuient sur le site internet d'Old Bailey[23]. Ces leçons ont été traduites en français sans « localisation », c'est-à-dire sans changer l'étude de cas pour l'adapter à un public francophone. Le constat a été fait depuis que la localisation est une donnée essentielle à la traduction. En outre, les captures d'écran proposées doivent être traduites dans la mesure du possible pour permettre une meilleure compréhension. En 2022, une présentation du chapitre francophone au colloque Humanistica a montré la nécessité d'appréhender la traduction de l'anglais vers le français en marquant une distance vis-à-vis d'une langue trop technique : l'enjeu consiste à surmonter les difficultés didactiques qui en découlent, tout en préservant un français standard et compréhensible par le plus grand nombre (Gille Levenson & Patat, 2022).

Récemment, en janvier 2025, la question s'est posée lors de la traduction d'une leçon portant sur les données ouvertes liées, initialement publiée en anglais en 2017 et dont

la traduction est en cours (Blaney, 2017). Dans la leçon originale, il est question de différencier deux Jack Straw homonymes grâce aux données liées par l'intermédiaire de leurs identifiants VIAF (Fichier d'autorité international virtuel). Le premier « Jack Straw » est un rebelle anglais du xive siècle, tandis que le second est un important ministre britannique de l'administration de Tony Blair. La leçon montre qu'il est possible de distinguer les deux Jack Straw en localisant le lieu d'activité. En l'occurrence, l'homme politique était membre du parlement britannique, représentant de la circonscription de Blackburn. Ces référentiels culturels, s'ils siéent à un lectorat anglophone, ne peuvent être aussi immédiatement compréhensibles auprès d'un lectorat francophone[24]. Ce type de réflexion, aussi anecdotique soit-il, doit trouver un juste milieu entre la compréhension immédiate du lectorat et l'alignement multilingue des leçons.

Enfin, puisque chaque chapitre publie ses leçons au fil de l'eau, comment quantifier les spécificités disciplinaires ou les outils représentés par les leçons ? L'index des leçons de chaque chapitre est pourvu de deux filtres intégrés. Au-delà du nombre total de leçons publiées que nous évoquions en introduction, ces filtres permettent de distinguer certaines tendances dans les activités attribuées. D'abord, une liste fermée de cinq activités, sous forme de verbes d'action, indique à quelle étape du traitement des données la leçon porte. Les leçons se distribuent de la façon suivante :

Nombre total de leçons	Chapitre de *Programming Historian*	Acquérir	Transformer	Analyser	Présenter	Préserver
41	En français	4	21	7	6	3
119	In English	13	39	37	28	2
68	En español	9	27	16	14	2
55	Em português	8	23	10	11	3

Tableau 1 – Répartition des leçons publiées par langue et par étape du traitement des données (mai 2026).

Dans tous les chapitres, les leçons portant sur la transformation des données sont les plus nombreuses. Cependant, le chapitre français montre une surreprésentation de cette catégorie, correspondant à la moitié de toutes les leçons publiées dans cette langue. Ce tableau témoigne aussi d'un manque d'homogénéité dans la distribution des leçons d'une étape à l'autre. Tous les chapitres ont, pour le moment, peu exploré ou rédigé de leçon en lien avec la préservation. À ce titre, le chapitre français peut s'enorgueillir d'avoir récemment publié une leçon originale sur l'archive Software

Heritage (Granger *et al.*, 2024).

Le second filtre des 279 leçons correspond aux sujets renseignés dans les métadonnées de chaque leçon pour en documenter le contenu. Il y a en moyenne 1,7 sujet renseigné dans une leçon. Ces 479 sujets attribués, triés par chapitre, permettent d'apporter une analyse plus fine de la couverture thématique proposée par *Programming Historian* :

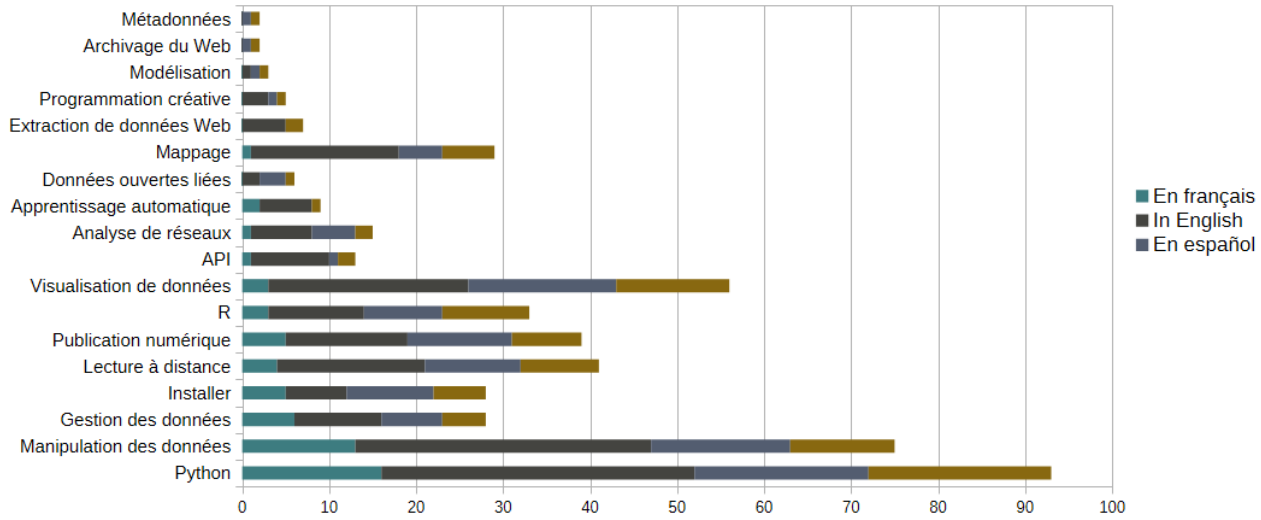


Figure 1 – Répartition des leçons publiées par thématiques et par langue (mai 2026). Les couleurs attribuées correspondent à l'habillage graphique de chaque chapitre linguistique.

Ces données confirment l'intérêt de *Programming Historian*, dès sa forme d'existence primitive, pour le langage de programmation Python. Par rapport à leur nombre de leçons publiées, les chapitres montrent également un vif intérêt pour les leçons portant sur la « manipulation des données », terme dont l'acception est suffisamment large pour couvrir de nombreuses thématiques. En revanche, le chapitre français semble pauvre en leçons traitant de la « visualisation des données » (trois leçons), en comparaison aux autres chapitres. Enfin, parmi les thématiques émergentes de la revue figure l'extraction de données web (*Web Scraping*) ; le chapitre français n'a, pour le moment, pas de leçon publiée sur ce thème. La figure 1 donne une idée de des thématiques couvertes par les 39 leçons francophones.

Perspectives

Cet état des lieux de *Programming Historian en français* nous permet de mettre en

perspective l'historique naissant de ce chapitre avec ses ambitions actuelles. Issu d'un projet universitaire, canadien et anglophone, *Programming Historian* a su prendre vie sous la forme d'une revue, s'adressant en priorité à un public novice sur le traitement des données en sciences humaines et sociales grâce à des outils numériques. À partir de ce grand principe, un processus éditorial a pu voir le jour, ainsi qu'un site rapidement déployé sur *GitHub Pages* et sur lequel les leçons sont publiées. Ce fonctionnement, réplicable et répliqué, a permis à trois autres langues d'intégrer la revue.

Dans un monde où le Web propose de tutoriels majoritairement anglophones, *Programming Historian* entend diversifier son offre de leçons, tutoriels « améliorés », pour être le plus en phase avec les besoins d'une communauté à construire, dont les valeurs poussent à mettre en avant les logiciels libres et gratuits. Bien que leur contenu ne soit pas figé, les leçons sont conçues comme des articles scientifiques, c'est pourquoi l'enjeu principal de la revue concerne sa pérennité. Là où d'autres revues publient des articles dans leur forme définitive, les leçons de *Programming Historian* doivent être mises à jour pour que les technologies présentées demeurent fonctionnelles. L'enjeu de pérennité est double : il concerne à la fois la structure du site et le contenu des leçons. Ainsi, d'un point de vue technologique, la plateforme dépend d'une solution poussée au maximum de ses capacités (Jekyll), ce qui invite à s'interroger et envisager, aujourd'hui, une migration vers une autre technologie (Janeway)[25]. Le site internet de *Programming Historian* est déployé dans un environnement Git, le contenu de son dépôt Jekyll est systématiquement archivé par Software Heritage, une archive ouverte pour les codes sources des logiciels[26].

Pour ce qui concerne le contenu, la pérennité des leçons passe par leur pertinence : un outil pris pour exemple peut ne plus être accessible, un logiciel peut tomber en désuétude au bout de quelques années. Les leçons sont donc adaptées (ou retirées, le cas échéant) pour éviter leur obsolescence. Ces risques sont aussi des forces pour l'équipe de la revue, dont les compétences nourrissent les réflexions sur l'adaptabilité de *Programming Historian* sur le long terme, après une décennie sur GitHub.

Parmi les perspectives les plus structurantes incarnées par le projet *Programming Historian*, il y a celle de faire vivre la revue autour d'une communauté de pratiques. L'objectif est de faire fructifier ce réseau de contributeurs et contributrices afin de renouveler le contenu des leçons au gré des besoins. Le suivi éditorial est ouvert et chaque personne impliquée dans la publication d'une leçon doit voir sa contribution valorisée et réalisée dans un cadre de confiance. Dans le domaine de l'édition scientifique, qui, souvent, obscurcit le travail gratuit réalisé par les évaluateur·trices et se caractérise par la violence des évaluations (Crymble, 2021), *Programming Historian* fait appel aux valeurs de la science ouverte pour visibiliser et protéger les personnes qui

vont la revue, tout en promouvant l'enseignement multilingue des humanités et sciences sociales numériques.

Références bibliographiques

Blaney, J. (2017). Introduction to the principles of linked open data. *Programming Historian*. https://doi.org/10.46430/phen0068

Calarco, G., & Riande, G. del R. (2025). Introduction à la publication web de fichiers TEI avec CETEIcean (brouillon). *Programming Historian*. https://web.archive.org/web/20250617111458/https://programminghistorian.github.io/ph-submissions/fr/en-cours/traductions/publier-archives-tei-ceteicean

Clavert, F. (2010). THATCamp Paris! *THATCamp Paris*. https://doi.org/10.58079/uo1a

Crymble, A. (2021). *Technology and the historian: Transformations in the digital age*. University of Illinois Press.

Crymble, A., & Im, C. M. H. (2023). Measuring digital humanities learning requirements in Spanish- and English-speaking practitioner communities. *International Journal of Digital Humanities, 5*(2), 253–282. https://doi.org/10.1007/s42803-023-00066-x

Dacos, M., & Mounier, P. (2015). *Humanités numériques*. Institut français. https://hal.science/hal-01228945

Fiormonte, D. (2021). Taxation against overrepresentation? The consequences of monolingualism for digital humanities. In D. Kim & A. Koh (Eds.), *Alternative histories of the digital humanities* (pp. 333–376). Punctum Books.

Flesch, M., & Chevrie, C. (2024). Appel à propositions (édition française). *Programming Historian*. https://programminghistorian.org/posts/appel-a-propositions

Flesch, M., & Chevrie, C. (2025). Appel à traductions de l'espagnol vers le français : Leçons sur la TEI. *Programming Historian*. https://programminghistorian.org/posts/appel-a-traductions

Flesch, M., & Hawes, A. (2026). Appel à contributions (édition française). *Programming Historian*. https://programminghistorian.org/posts/appel-a-contributions

Gibbs, F. (2013). Installing Python modules with pip. *Programming Historian*. https://programminghistorian.org/en/lessons/installing-python-modules-pip

Gille Levenson, M., Papastamkou, S., & Ringwald, C. (2022). Programming Historian : Un lieu de collaborations et d'interactions multiples. *DHNord 2022: Travailler en humanités numériques: collaborations, complémentarités et tensions*.

Gille Levenson, M., & Patat, G. (2022). Programming Historian en français : Faire communauté pour le partage de ressources éducatives libres sur les méthodes numériques en sciences humaines et sociales francophones. *Humanistica 2022*. https://hal.science/hal-03672420

Gille Levenson, M., Ringwald, C., Flesch, M., Isasi, J., Papastamkou, S., Quiroga, R., & Valentine, D. (2024). Connecter les chapitres linguistiques de Programming Historian? *HAL - Sciences de l'information et de la communication*.

https://hal.science/hal-04557817
Granger, S., Mélès, B., & Santos, F. (2024). Préserver et rendre identifiables les logiciels de recherche avec Software Heritage. *Programming Historian en français*. https://doi.org/10.46430/phfr0034
Isasi, J., Quiroga, R., Siddiqui, N., Paulino, J. V., & Wermer-Colan, A. (2023). A model for multilingual and multicultural digital scholarship methods publishing: The case of Programming Historian. In *The case of Programming Historian* (pp. 17–30). Taylor & Francis. https://doi.org/10.4324/9781003393696-3
Jekyll. (2025). *Jekyll: Simple, blog-aware, static sites*. https://jekyllrb.com/
Lincoln, M. D. (2020). Multilingual Jekyll: How The Programming Historian does that. https://matthewlincoln.net/2020/03/01/multilingual-jekyll.html
Lincoln, M., Isasi, J., Melton, S., & Laramée, F. D. (2022). Relocating complexity: The Programming Historian and multilingual static site generation. *Digital Humanities Quarterly, 16*(2).
Longhi, J. (2019). Contours, perspectives et tensions des humanités numériques. *Sens-Dessous, 24*(2), 123. https://doi.org/10.3917/sdes.024.0123
McDaniel, C. (2014). How we moved the Programming Historian to GitHub Pages. *Programming Historian*. https://programminghistorian.org/posts/how-we-moved-to-github
McGrath, J. (2020). THATCamp reflections: On the unfinished business of unconferences. *THATCamp Retrospective*. https://retrospective.thatcamp.org/2020/03/03/thatcamp-reflections-on-the-unfinished-business-of-unconferences/index.html
Papastamkou, S. (2018). François Dominic Laramée joins the Programming Historian project team. *Programming Historian*. https://programminghistorian.org/posts/fd-laramee
Papastamkou, S. (2019). Bienvenue au Programming Historian en français! *Programming Historian*. https://programminghistorian.org/posts/bienvenue-ph-fr
Pecout, H. (2022). Donner de l'R aux SHS. *La Lettre Humanités numériques*. CNRS Sciences humaines & sociales. https://www.inshs.cnrs.fr/fr/cnrsinfo/donner-de-lr-aux-shs
Ross-Hellauer, T. (2017). What is open peer review? A systematic review. *F1000Research, 6*, 588. https://doi.org/10.12688/f1000research.11369.2
Shorter, E. (1971). *The historian and the computer: A practical guide*. Prentice-Hall.
Terras, M. (2011). Quantifying digital humanities. *UCL Centre for Digital Humanities*. https://web.archive.org/web/20250629045645/https://www.ucl.ac.uk/infostudies/melissa-terras/DigitalHumanitiesInfographic.pdf
Thély, N., Serres, A., & Deuff, O. L. (2014). Le THATCamp comme nouvelle forme de communication scientifique? *Éditions de la Maison des Sciences de l'Homme*. https://doi.org/10.4000/books.editionsmsh.2183
Turkel, W. J. (2008). Digital history hacks (2005–08): The Programming Historian. *Digital History Hacks*. https://digitalhistoryhacks.blogspot.com/2008/01/programming-historian.html
Turkel, W. J., & Crymble, A. (2012). Décomptes d'occurrences de mots en Python. *Programming Historian*. https://programminghistorian.org/fr/lecons/decomptes-de-frequences-de-mots-en-pytho

n
Turkel, W. J., & MacEachern, A. (2011). The Programming Historian. *NiCHE: Network in Canadian History & Environment*. https://niche-canada.org/research/niche-digital-infrastructure-project/the-programming-historian/
Vidal-Gorène, C. (2023). La reconnaissance automatique d'écriture à l'épreuve des langues peu dotées. *Programming Historian*. https://doi.org/10.46430/phfr0023
Waquet, F. (2022). *Dans les coulisses de la science: Techniciens, petites mains et autres travailleurs invisibles*. CNRS Éditions.

Notes

[1] « Nothing exists at present for the historian who wants to use a computer but has not the slightest idea how to proceed » (Shorter, 1971: 1).
[2] The Journal of Open Source Software. https://joss.theoj.org/
[3] The Journal of Open Source Education. https://jose.theoj.org/
[4] The Carpentries. https://carpentries.org/
[5] DARIAH (Digital Research Infrastructure for the Arts and Humanities). OpenMethods : https://openmethods.dariah.eu/. Dariah-Campus. https://campus.dariah.eu/resources/
[6] Rzine, https://rzine.fr/
[7] DEMC Journal (Données Expériences Méthodes Code) https://demc-journal.org/
[8] Data & Corpus. https://dc.episciences.org/
[9] Programming Historian. https://programminghistorian.org/
[10] Jekyll. (2025, janvier 29). Jekyll. Simple, blog-aware, static sites. Jekyll. https://jekyllrb.com/
[11] Ross-Hellauer, Tony, « What is open peer review? A systematic review », F1000Research, 6, 2017.
[12] Programming Historian. (2025, mai 1). Contribuer. https://programminghistorian.org/fr/contribuer
[13] Programming Historian. (2025, avril 3). Publishing Workflow. GitHub. https://github.com/programminghistorian/ph-submissions/wiki/Home
[14] Ce système n'est pas inscrit dans le marbre et sera amené à évoluer avec la propre évolution de l'infrastructure de Programming Historian : des discussions sont en cours pour s'arrimer à des infrastructures éditoriales professionnelles.
[15] Il existe par ailleurs un rôle de médiateur·trice dans la revue, afin de gérer tout conflit qui pourrait survenir.
[16] Le XML-TEI (Text Encoding Initiative) est une norme d'encodage de textes fondée sur le langage balisé XML, destinée à structurer, décrire et échanger des documents textuels de manière interopérable. Il est principalement utilisé dans le domaine des sciences humaines et sociales.
[17] Programming Historian, Consignes aux auteur(e)s, 10 janvier 2025 : https://programminghistorian.org/fr/consignes-auteurs
[18] Calarco, G., & Riande, G. del R. (2025). Introduction à la publication web de fichiers

TEI avec CETEIcean (brouillon de la traduction française). Programming Historian. https://web.archive.org/web/20250617111458/https://programminghistorian.github.io/ph-submissions/fr/en-cours/traductions/publier-archives-tei-ceteicean

[19] L'HTR, pour Handwritten Text Recognition, est une technologie informatique qui vise à identifier, sur des images numériques, les signes (ou glyphes), ce qui permet notamment d'obtenir la transcription de documents manuscrits en texte numérique éditable.

[20] Programming Historian. (2021, janvier 11). Translation Concordance. https://programminghistorian.org/translation-concordance

[21] Colloque Humanistica 2024, colloque annuel de l'Association francophone des humanités numériques 3-9 mai 2024 FLSH-UMI Meknès (Maroc). https://humanistica2024.sciencesconf.org/

[22] Colloque Humanistica 2025, colloque annuel de l'Association francophone des humanités numériques 22-25 avril 2025 Dakar (Sénégal). https://humanistica2025.sciencesconf.org/

[23] Le Old Bailey est la cour centrale de la Couronne Britannique. Le projet « The proceedings of the OldÂ Bailey » recueille 200 000 minutes de procès ayant eu lieu entre 1674 et 1913. Il a servi pour un nombre important de leçons de Programming Historian en anglais (par exemple « Normalizing Textual Data with Python » en 2012 : https://programminghistorian.org/en/lessons/normalizing-data).

[24] Lien vers le ticket GitHub, traduction de la leçon « Introduction aux principes des données ouvertes liées » : https://github.com/programminghistorian/ph-submissions/issues/643

[25] Lien vers le ticket GitHub Technical Infrastructure Renewal - update and next steps, https://github.com/programminghistorian/jekyll/issues/3530

[26] Software Heritage, Dépôt GitHub « Jekyll » de Programming Historian, archivé le 5 mars 2026.Â https://archive.softwareheritage.org/swh:1:dir:fea288497026fd07a9c90616e375dd444a70a29b