Université Paris-Panthéon-Sorbonne

M2 - ESH

Le figure de l'économiste programmeur

Par Ambroise WARNERY

Mémoire de recherche

Dirigée par Francesco SERGI

Et par Dorian JULLIEN

Présenté et soutenue le XX/06/2025

Chapitre 1

La figure de l'économiste programmeur

Mary Morgan, dans son livre de 2012, The World in the Model : How Economists Work and Think[18], décrit l'économie comme une "science outillée", et analyse le rôle des modèles. Un autre outil, dont le développement a entre autre permit de créer des modèles de plus en plus complexe, joue un rôle essentiel dans le travail de l'économiste.

Il s'agit de l'outil informatique.

Depuis l'invention de l'informatique, l'usage de cet outil en économie n'a cessé de s'intensifier, jusqu'à devenir un élément central de nombreuses pratiques : simulation, expérimentation, traitement de données, modélisation agent, etc. Derrière cette mutation technique, souvent analysée en termes d'innovations méthodologiques ou de nouvelles normes scientifiques, se trouvent des trajectoires individuelles bien concrètes : celles d'économistes-programmeurs, à la fois chercheurs et développeurs, dont les contributions ne se limitent pas à l'usage d'outils existants, mais incluent la création de logiciels devenus centraux dans certains sous-champs de la discipline (voir Boumans et al. 2023[5], Cherrier et al. 2023[6], Backhouse et al. 2017 [2]). Ce travail s'appuie sur une série d'entretiens réalisés dans le cadre du projet Oral Histories of Economics, avec sept figures ayant contribué de manière significative à cette hybridation entre économie et informatique : David Hendry[12], Joshua Epstein[8], Robert Axtell[1], Theodore Turocy[22], Urs Fischbacher[10], Richard Pierse[19] et Agnès Gramain[11]. À travers leurs récits, il s'agit de documenter, comparer et analyser leurs trajectoires, afin de mieux comprendre comment se forment, s'exercent et se diffusent les compétences de programmation en économie, quels types de profils les incarnent, et comment ces contributions informatiques s'articulent à des engagements scientifiques plus

larges. L'objectif est triple. Il s'agit d'abord d'identifier les traits communs qui caractérisent ces économistes-programmeurs : formes d'apprentissage de l'informatique, profils académiques hybrides, diversité des langages mobilisés, rapports de genre, ou encore tensions face aux limites techniques et institutionnelles rencontrées. Il s'agit ensuite de mettre en lumière les différenciations, générationnelles, institutionnelles, techniques, qui traversent ces parcours, et d'interroger des questions transversales : comment évoluent les langages utilisés ? quelles sont les conditions de reconnaissance de ces contributions logicielles ? existe-t-il un profil type de l'économiste-programmeur ? Enfin, cette synthèse propose d'approfondir l'analyse du lien entre la programmation et la contribution scientifique en économie : les logiciels sont-ils conçus pour soi, pour les autres, ou pour les deux ? Comment ces outils redéfinissent-ils les objets, les méthodes et les normes de la discipline ? Ce travail entend ainsi contribuer à une meilleure compréhension des savoirs économiques outillés, en montrant que la programmation ne relève pas simplement d'une compétence technique, mais d'une manière singulière de faire de la science économique, avec pour objectif de mieux comprendre comment ces économistes contribuent à transformer l'économie par leurs innovations techniques et théoriques. Ce travail suppose toutefois de rester attentif aux limites propres à la méthode de l'entretien, qui constitue une source située, construite, et traversée par des enjeux de légitimité scientifique : les discours produits par les économistes sur leurs parcours et leurs pratiques sont marqués par des stratégies narratives, des oublis ou des silences (Jullien, 2018[14]). Nous tacherons donc de croiser ces matériaux avec d'autres types de sources, et d'en proposer une lecture réflexive, attentive aux conditions de leur production comme à leur performativité dans l'écriture de l'histoire.

Biographies

Avant de commencer l'analyse de nos différents personnages, il est important d'avoir en tête quelques éléments de leurs biographies.

David Hendry Économètre britannique, formé en mathématiques appliquées, il a obtenut son PhD en 1970. David Hendry a profondément influencé l'économie empirique en développant des outils logiciels (PcGive, OxMetrics) pour automatiser et fiabiliser les tests économétriques. Professeur à Oxford, il a également joué un rôle majeur dans la structuration institutionnelle de l'économétrie au Royaume-Uni.

Joshua Epstein Chercheur américain, formé en mathématiques et en sciences politiques, Joshua Epstein obtient son PhD en 1981. Il est un pionnier de la modélisation agent-based en sciences sociales, notamment avec le modèle Sugarscape co-développé avec Robert Axtell. Il a mené sa carrière entre think tanks, épidémiologie, économie et recherche interdisciplinaire, plaidant pour une épistémologie générative.

Richard Pierse Économiste britannique, diplomé en 1979 d'un master en Mathématique économique et econometrie de la LSE, Richard Pierse commence un PhD en économétrie qu'il n'achévera pas, faute de financements. Il est l'auteur des logiciels NIModels et Winsolve, des outils de simulation macroéconomique appliqués à la politique monétaire, très utilisés dasn les Banques Centrales.

Urs Fischbacher Chercheur suisse, Urs Fischbacher obtient un PhD en mathématiques théoriques en 1985 puis travail comme ingénieur logiciel. Il devient ensuite une figure clé de l'économie expérimentale grâce à la création de z-Tree, un logiciel de référence pour la conduite d'expériences en laboratoire, aujourd'hui utilisé dans le monde entier.

Robert Axtell Robert Axtell est un chercheur américain formé à l'ingénierie, puis au croisement de l'économie, de l'informatique et des politiques publiques. Il obtient son PhD en 1992. Il a co-développé avec Joshua Epstein le modèle Sugarscape, pierre fondatrice de la modélisation multi-agents en sciences sociales.

Agnès Gramain Formée à l'ENSAE, Agnès Gramain est chercheuse française spécialiste de microéconomie appliquée et de modélisation. Elle obtient son PhD en 1998 en économie de la santé.

Theodore Turocy Chercheur américain, diplômé de Caltech en informatique appliquée et sciences sociales, Theodore Turocy obtient un PhD en 2001 sur les enchères et les méthodes computationnelles pour déterminer les équilibres en situation complexe. Il s'est spécialisé dans la théorie des jeux computationnelle. Développeur principal du logiciel GAMBIT, il combine recherche académique, enseignement et projets d'intelligence artificielle appliquée à la décision stratégique.

1.1 Les traits communs entre les économistes programmeurs

Un trait partagé par la majorité des économistes-programmeurs interrogés est leur apprentissage autodidacte de l'informatique et des langages de programmation. Tous ont acquis leurs compétences par nécessité, par expérimentation, ou par curiosité, dans des environnements qui rendaient ces apprentissages possibles. Theodore Turocy illustre parfaitement cette trajectoire : il commence à coder dès l'enfance, sur un TI-99/4A que son grand-père lui offre. Il explore par lui-même différents langages, jouant avec les outils disponibles à la maison ou à l'école. Cette familiarité précoce avec la machine s'inscrit dans un

contexte nord-américain où l'accès aux micro-ordinateurs domestiques, bien que coûteux, est en pleine démocratisation, grace aux politiques publics de subventions. Il semble aussi que le niveau socio-économique soit un fort prédicteur de l'adoption de l'ordinateur dans les foyers (Bureau of Labor Statistics 1999[7], Schmitt 2002[21]). David Hendry, de son côté, découvre l'informatique à l'université, à l'époque des premiers ordinateurs centralisés, les "mainframe". Ces ordinateurs occupent une pièce entière, et le code doit être inscrit sur des cartes cartons à trous pour qu'il soit lu par l'ordinateur. Il relate une expérience frustrante lorsqu'il essaye d'apprendre Atlas Autocode, un langage spécifiquement prévu pour l'ordinateur Atlas du Center of London University, qui lui semble très bizarre. Il se tourne ensuite vers Fortran, le langage le plus utilisé à l'époque, car pouvant être utilisé sur les ordinateurs mainframe d'IBM, les plus performants de cette époque. Il reçoit alors quelques cours dans le cadre de son PhD, enseigné par Carol Hewlett. Ce sont les échanges entre pairs et les essais-erreurs qui forment le cœur de sa progression. Joshua Epstein, bien que formé en mathématiques et en sciences sociales, n'a pas reçu d'enseignement en programmation. C'est dans le cadre du projet Sugarscape, au début des années 1990, qu'il découvre de manière pragmatique la modélisation informatique, aux côtés de Robert Axtell. À l'époque, aucun outil dédié à la modélisation agent n'existait : ils doivent tout construire eux-mêmes, à partir de langages de bas niveau. Pour comprendre la différence entre langage de bas niveau et langage de haut niveau, on peut utiliser la métaphore de la construction d'une maison. Programmer en bas niveau revient à poser chaque brique soi-même : on contrôle précisément chaque étape, on comprend comment tout fonctionne, mais cela demande du temps et beaucoup d'efforts. À l'inverse, programmer en haut niveau, c'est comme assembler une maison à partir d'éléments préfabriqués : on avance plus vite, car beaucoup d'éléments sont fabriqués par d'autres personnes, mais on perd en transparence, on ne sait pas vraiment comment sont fabriqués ces éléments préfaits, et en maîtrise fine de la structure. Urs Fischbacher et Richard Pierse montrent également des trajectoires d'apprentissage autonome et pragmatique, acquérant leurs compétences informatiques principalement à travers des ordinateurs mis à disposition dans leurs lycées. Enfin, Agnès Gramain raconte avoir commencé à programmer en SAS à son entrée à l'ENSAE. C'est les habitudes et les manuels ("pieuvres" rédigés par un administrateur) de programmation développé à l'INSEE, qui sont mobilisé dans cet école formant les futurs cadres de l'institut. C'est donc la seule qui s'inscrit dans une formation par et pour une Institution. Dans tous les cas, cette autoformation s'est déroulée dans des environnements techniquement bien équipés : universités disposant de laboratoires informatiques, accès précoce à des machines coûteuses, parfois dès le lycée ou à domicile. Ces conditions matérielles, souvent invisibilisées, ont facilité l'exploration et l'appropriation des outils numériques, à une époque où la documentation était rare et les interfaces peu conviviales. L'apprentissage de la programmation n'est donc pas seulement une affaire de volonté individuelle : il est aussi rendu possible par des conditions d'accès matérielles et institutionnelles favorables, dans des contextes où la curiosité scien-

tifique pouvait s'exprimer librement. Le système PLATO en est un bon exemple (Lee p.10 2004[16]).

Second trait commun, aucun d'entre eux n'a été formé comme "pur" économiste, et ils témoignent d'un intérêt précoce pour les mathématiques, l'informatique ou la physique avant de se tourner vers l'économie ou les sciences sociales. Dans leurs parcours académiques, ils ont souvent commencés dans des disciplines scientifiques, souvent exigeantes sur le plan mathématique, l'implication dans l'économie ou les sciences sociales, venant dans un mouvement de bifurcation intellectuelle guidé par la volonté de comprendre les phénomènes sociaux à l'aide d'outils formels. Cette hybridité se reflète dans leurs objets d'étude (théorie des jeux, économétrie, dynamiques sociales, expérimentations) et dans leur manière d'aborder les problèmes. Turocy incarne typiquement ce profil hybride : diplômé de Caltech, il y suit un double cursus en informatique appliquée et en sciences sociales, avant de s'orienter vers la théorie des jeux computationnelle. Dès ses premières années, il développe des outils logiciels et participe à des projets de recherche expérimentale. Epstein commence par la musique, puis se passionne pour les mathématiques pures, qu'il cherche ensuite à appliquer à l'étude des systèmes sociaux. Il choisit de faire un doctorat en sciences politiques au MIT, avec une forte composante en économie. C'est dans ce cadre qu'il développe ses premiers modèles mathématiques, avant de passer à la simulation agent-based, notamment avec le projet Sugarscape. Hendry suit initialement une formation en mathématiques appliquées, avant d'être orienté vers l'économétrie sous l'influence décisive de Denis Sargan, son mentor à la London School of Economics. Il devient ensuite une figure centrale de l'économétrie computationnelle au Royaume-Uni. Fischbacher étudie les mathématiques théoriques à l'université, obtient un doctorat dans cette discipline, puis travaille brièvement comme ingénieur logiciel. C'est par hasard, lors d'une opportunité professionnelle, qu'il intègre un laboratoire d'économie expérimentale. Il y découvre un champ en construction, où ses compétences en programmation se révèlent précieuses, ce qui l'amène à concevoir le logiciel z-Tree et à s'ancrer durablement dans l'économie expérimentale. Pierse a une formation en Philosophie, Politique et Économie (PPE), une filière généraliste bien connue au Royaume-Uni. Ce n'est pas sa formation universitaire, mais ses emplois d'été au National Institute of Economic and Social Research (NIESR) qui l'amènent à pratiquer l'économétrie appliquée, dans un environnement où l'informatique devient progressivement centrale. Gramain est la seule, avec Pierse, à avoir eu une formation en économie dès sa scolarité. Passée par une classe préparatoir B/L, très pluridsciplinaire (Philosophie, Histoire, Economie, Math), elle intègre l'ENSAE, où elle est formée à la fois à l'économie, aux statistiques et aux mathématiques appliquées. C'est dans le cadre de ses premiers travaux que la programmation devient un outil indispensable. Malgré des parcours variés, tous partagent un socle scientifique solide, principalement en mathématiques, qui les prépare à mobiliser des outils computationnels. Fischbacher poursuit un doctorat en mathématiques ; Epstein sou-

tient en sciences politiques, Hendry, Turocy et Gramain en économie. Pierse, quant à lui, ne termine pas son doctorat, faute de financement, bien qu'il mène une carrière active dans le monde institutionnelle et à la Banque d'Angleterre en tant que chercheur. Ce qui les réunit, c'est que leurs premiers travaux de recherche mobilisent immédiatement l'informatique, que ce soit pour simuler, expérimenter ou automatiser. L'hybridité disciplinaire n'est pas un détour mais un point d'ancrage essentiel de leurs trajectoires : c'est précisément parce qu'ils viennent d'ailleurs qu'ils ont apporté à l'économie des outils et des méthodes computationnelles innovantes.

Un autre trait commun à ces économistes-programmeurs est leur souplesse dans l'usage des langages de programmation. Aucun d'entre eux ne témoigne d'une fidélité durable à un langage unique. Tous adoptent une démarche pragmatique, choisissant leurs outils en fonction des contraintes techniques, des projets en cours, ou des évolutions technologiques. David Hendry a commencé à programmer en Fortran, langage standard des années 1970-80 dans les milieux économétriques. Par la suite, il collabore étroitement avec Jurgen Doornik, qui réécrit une grande partie de leur outil en C, avant de lui déléguer le développement du langage Ox, spécifiquement conçu pour l'économétrie appliquée et pour leurs logiciels. Cette évolution illustre une adaptation constante aux enjeux de performance et de reproductibilité. Theodore Turocy, a commencé à coder très jeune sur un TI-99/4A familial, l'un des premmiers ordinateurs familiaux, sur lequel il a appris le BASIC, puis a utilisé d'autres langages comme le C, Maple, Python et d'autres outils selon les besoins. Il a également contribué au développement de GAMBIT, un environnement dédié à la théorie des jeux computationnelle. Sa trajectoire montre une grande aisance à naviguer entre langages, dans une logique d'expérimentation et d'efficacité. Joshua Epstein, bien que formé initialement à la modélisation mathématique "papier-crayon", s'est rapidement tourné vers des langages comme C++ pour développer Sugarscape au début des années 1990, à une époque où aucun environnement de modélisation agent n'existait encore. Par la suite, il adopte des plateformes comme Repast ou NetLogo, dans un souci de portabilité et de diffusion. Urs Fischbacher a, pour sa part, conçu le logiciel z-Tree en C++, qu'il continue d'utiliser pour des raisons de performance, de portabilité et de stabilité. Il a aussi beaucoup utilisé Pascal. Fischbacher souligne l'importance de pouvoir tout coder lui-même, y compris l'interface, afin d'assurer une compatibilité et une fiabilité maximales dans les laboratoires d'économie expérimentale. Richard Pierse a d'abord appris à programmer en Fortran dans le cadre de son doctorat à Cambridge. Il adopte ensuite GAUSS, lors de son recrutement à Cambridge. Un langage très utilisé dans les années 1980-1990 pour l'économétrie, notamment pour sa souplesse dans la résolution de systèmes non linéaires et l'analyse de séries temporelles. Par la suite, il sera contraint d'apprendre à utiliser C++ pour pouvoir contruireson logiciel, Winsolve. Agnès Gramain fait figure d'exception : formée à SAS pendant sa scolarité à l'ENSAE, elle développe une forte affinité avec ce langage et affirme que ce dernier à struc-

turer la manière dont elle pense, mais nous développperons cet aspect dans le paragraphe suivant. Au-delà des langages eux-mêmes, ces trajectoires illustre une approche pragmatique et artisanale du développement logiciel : ce qui compte, ce n'est pas le langage en soi, mais la logique sous-jacente, ce qu'il permet de faire. Cependant, l'expérience d'Agnès Gramain semble montrer que les habitudes et les philosophies peuvent être chambouler par un changement de langage.

Un autre trait frappant des trajectoires étudiées est la très forte masculinisation du milieu de la programmation en économie. Parmi les six économistes-programmeurs interviewés, une seule est une femme : Agnès Gramain. Agnès Gramain évoque elle-même l'environnement masculin dans lequel elle a été formée : tant à l'ENSAE qu'au sein des laboratoires où elle développe ses compétences en simulation, les femmes sont rares. Elle ne rapporte pas d'obstacles explicites, mais son témoignage suggère que la programmation reste, dans son parcours, un domaine majoritairement investi par des hommes, où il faut trouver sa place sans modèles féminins nombreux. L'absence d'autres femmes parmi les personnes interrogées peut être du à la petite taille de notre échantillon, mais elle reflète probablement une réalité plus large : l'informatique et l'économie sont deux disciplines historiquement masculinisées, et leur point d'intersection, celui de la programmation en économie, l'est a fortiori. La littérature établit déja clairement l'écart entre le nombre de femmes dans la population et le nombre de femmes étudiant l'informatique (Beyer, 2014[4]) faisant de la recherche en informatique (Falkner et al., 2015[9]). Le même constat peut etre fait en économie (Kahn, 1993[15] et Bateman, 2023[3]). Cette sous-représentation manifeste invite à interroger les rapports de genre dans l'accès aux compétences informatiques et aux carrières scientifiques situées à l'interface entre économie et informatique. Plusieurs hypothèses peuvent être avancées : cette situation est-elle le résultat de biais historiques de recrutement, de barrières d'entrée structurelles à l'apprentissage de l'informatique pour les femmes, ou d'une construction genrée des compétences techniques, socialement valorisées chez les hommes et moins encouragées chez les femmes ? Le cas isolé d'Agnès Gramain ne permet pas à lui seul de tirer des conclusions définitives, mais il souligne la nécessité d'une réflexion plus large sur les inégalités de genre dans les pratiques informatiques en économie. Le fait que les économistes-programmeurs soient presque exclusivement des hommes dans cette enquête dit quelque chose des dynamiques d'exclusion ou d'auto-sélection à l'œuvre, encore aujourd'hui, dans l'économie computationnelle.

Bien que tous les économistes-programmeurs interrogés valorisent la puissance de l'informatique dans leurs recherches, leurs trajectoires sont aussi marquées par des moments de frustration, de blocage ou de découragement face aux défis techniques, matériels ou institutionnels liés à l'usage de la programmation. David Hendry évoque les difficultés de ses débuts, au moment de la transition entre les ordinateurs centraux (mainframes) et les pre-

miers ordinateurs personnels. Il relate les complications liées au langage Fortran, aux limites matérielles des machines de l'époque, ainsi qu'à l'absence d'environnement de travail ergonomique. La lenteur des calculs ou les erreurs de compilation complexes à diagnostiquer ont constitué des obstacles importants pour ses premiers travaux économétriques. Joshua Epstein, de son côté, insiste sur la rudesse des débuts de la modélisation agent-based, à une époque où aucun environnement logiciel dédié n'existait encore. Lors de la conception de Sugarscape, il travaille avec Rob Axtell sans bibliothèque[1] et avec très peu de documentation. Il décrit cette période comme exaltante mais techniquement aride, marquée par une forme de bricolage permanent et de navigation à vue dans un champ scientifique encore inexistant. Urs Fischbacher, qui développe z-Tree presque seul, souligne les difficultés techniques liées au développement et au maintien du logiciel : il doit tout concevoir, y compris l'interface graphique et les permettant l'interaction en temps réel entre les joueurs. Richard Pierse mentionne également des problèmes techniques concrets dans son travail à la Banque d'Angleterre. Lors de son second passage au sein de cet institution, il réalise notamment que trois versions de son programme NIModels ont évolué dans des directions les rendant désormais incompatibles. Il témoigne aussi de difficulté lors de l'utilisation de bases de données économiques volumineuses, ou dans la mise en œuvre de simulations complexes nécessitant une puissance de calcul importante. Ces défis sont amplifiés par la nécessité de produire des résultats exploitables dans des délais courts, au service de la politique monétaire. Agnès Gramain, elle, vit très mal le fait d'être contrainte d'utiliser GAUSS pendant sa thèse, pour des questions de performances, alors qu'elle avait appris à maitriser SAS pendant sa formation universitaire. Elle essaye maintenant d'apprendre à utiliser R, un logiciel gratuit, libre et donc plus accessible à ses étudiants. Mais témoigne retourner vers SAS lorsqu'elle doit faire de la recherche :

> *"Mais c'est comme une langue maternelle. Je pense qu'une fois que tu as appris une langue, ça structure ta manière de penser quand même."*
>
> Gramain, 2024[11]

Enfin, Theodore Turocy revient sur les obstacles rencontrés pour faire reconnaître et financer GAMBIT : pendant près de vingt ans, le logiciel, pourtant utilisé, ne bénéficie d'aucun soutien institutionnel. Ce manque de reconnaissance et de moyens techniques freine considérablement son développement, malgré son utilité démontrée dans la communauté des théoriciens des jeux. Ces témoignages montrent que le travail informatique est souvent invisible, chronophage, et peu valorisé dans les circuits académiques. Les défis rencontrés ne sont pas seulement techniques, mais aussi institutionnels et symboliques, renforçant parfois un sentiment d'isolement ou de sous-valorisation du travail accompli.

1. ensemble de fonctions ou routines pré-écrites, réutilisables pour faciliter le développement d'applications informatiques.

1.2 Les différences ou aspects à questionner

Malgré ces traits communs, plusieurs aspects différencient ces économistes programmeurs ou méritent d'être questionnés plus précisément. Premièrement, ces économistes appartiennent à des générations différentes, qui ont connu des stades variés de l'évolution des technologies informatiques. Ces différences générationnelles ont structuré leur accès aux machines, aux langages, et aux pratiques de programmation. David Hendry représente la génération des pionniers : il débute sa carrière dans les années 1970, à l'époque des mainframes IBM et des cartes perforées. L'informatique est alors centralisée, peu accessible, et la programmation se fait en Fortran, sur des machines coûteuses et lentes. Theodore Turocy, formé dans les années 1990 à Caltech, appartient à une génération qui a grandi avec l'ordinateur personnel à domicile. Il commence à programmer vers huit ans sur un micro-ordinateur familial, en BASIC, et bénéficie très tôt d'un environnement académique intensément informatisé. À l'université, il combine des cours de sciences appliquées, d'informatique et d'économie, et participe dès ses années de licence à des projets logiciels (comme GAMBIT). Il symbolise une intégration précoce et fluide de l'informatique dans la formation des économistes. Joshua Epstein, Urs Fischbacher et Richard Pierse se situe dans une position intermédiaire. La formation d'Epstein commence avant la diffusion massive des PC, mais il adopte très tôt les outils computationnels dans ses travaux. Lors du développement de Sugarscape au début des années 1990, aucun environnement de modélisation agent n'existe encore : il doit programmer « à la main », sans interface dédiée. Il appartient à une génération charnière, qui passe de la modélisation mathématique aux simulations informatiques, dans un contexte encore expérimental. De son côté, Fischbacher développe z-Tree au milieu des années 1990, dans un moment où les PC sont déjà largement disponibles, et où l'usage d'ordinateurs en laboratoire devient central dans l'économie expérimentale. Il bénéficie des environnements de développement modernes avec la version orienté objet de PASCAL. Enfin, Agnès Gramain, a été formée dans un environnement proche des grandes structures informatiques universitaires françaises, où l'usage de SAS était encore dominant. Elle témoigne avoir du faire tourner ses programmes en "batch" pendant la nuit, sur les ordinateurs de l'Insee. Cependant, dès son doctorat, l'usage d'ordinateurs personnels est très développé. Ainsi, les trajectoires de ces économistes reflètent bien l'évolution des infrastructures informatiques : des mainframes centralisés aux PC personnels, des langages compilés aux environnements interactifs. Chaque génération s'approprie l'informatique dans des conditions techniques, pédagogiques et institutionnelles spécifiques.

Les parcours des économistes-programmeurs reflètent également l'évolution des langages de programmation utilisés en économie au fil des décennies. Si tous ont commencé avec des outils plus complexes à maitrisés ou spécialisés – comme FORTRAN, C++, GAUSS

ou Maple – les standards contemporains semblent désormais converger vers Python et R. En effet, ces langages n'ont pas été conçut en premier lieu pour l'économie. Cependant, leurs caractères accessibles (syntaxe claire, apprentissage facilité), open source (ils sont accessibles gratuitement, tout le monde peut en consulter l'architecture interne et le fonctionnement) et les écosystème puissants qui se sont créer autour d'eux (des communautés d'utilisateurs actifs développent et enrichissent continuellement des bibliothèques spécialisés) leurs ont permis de s'imposés comme des outils de référence pour l'analyse statistique, la modélisation économique et le traitement de données. L'histoire des trajectoires de cees économistes est aussi celle de l'évolution de leurs outils de travail. Le langage C++, quoique toujours pertinent, paraît davantage réservé à des applications spécifiques ou à des besoins computationnels particuliers. Fischbacher l'utilisant largement pour créer zTree. Utilisé à l'origine par Epstein et d'autres, il tend à être délaissé dans les usages courants, au profit de langages plus haut niveau et plus lisibles. Sa complexité, ainsi que la courbe d'apprentissage qu'il impose, en font aujourd'hui un outil de niche, réservé à des contextes exigeants.Theodore Turocy, utilise aujourd'hui Python dans ses activités de formation et de recherche, et le considère comme un outil adapté aux besoins modernes de la théorie des jeux computationnelle. Joshua Epstein, sans mentionner un langage unique, cite également Python parmi les environnements désormais incontournables dans la modélisation agent-based, aux côtés de plateformes comme NetLogo ou Repast. Il souligne combien l'arrivée de ces outils a facilité l'accessibilité de la simulation sociale, par rapport aux débuts plus artisanaux en C++. Agnès Gramain, pour sa part, a été formée à SAS à l'ENSAE, le logiciel statistique privilégié par l'INSEE à l'époque. Aujourd'hui, dans le cadre de son enseignement, elle cherche à se tourner vers R, un langage libre, gratuit et plus facilement accessible à ses étudiants. Elle rapporte cependant que cette transition est difficile : les logiques syntaxiques et conceptuelles de R lui semblent très éloignées de celles de SAS, qu'elle maîtrise parfaitement. Ce témoignage met en lumière les obstacles cognitifs et pratiques que peuvent rencontrer les chercheurs confrontés à des évolutions techniques rapides, même lorsqu'ils en saisissent les enjeux pédagogiques. En somme, l'évolution des langages utilisés par ces économistes reflète une transition structurelle dans les outils de travail de la discipline : d'environnements spécialisés, souvent payants et complexes, vers des langages plus ouverts, communautaires et pédagogiquement adaptés, comme Python et R. Mais cette transition reste inégalement vécue, selon les trajectoires, les générations et les usages.

Troisièmement, la reconnaissance académique du travail logiciel demeure problématique. C'est un constat partagé largement dans la littérature (Merow et al. 2023[17], Howison et al 2011[13]). Tous ont développé des outils informatiques cruciaux pour leurs recherches – voire pour celles d'une communauté entière – mais la reconnaissance institutionnelle de ces contributions demeure variable, souvent limitée et indirecte. Turocy est explicite sur ce point : le logiciel GAMBIT, auquel il a consacré plusieurs années, est fréquemment ab-

sent ou relégué dans les publications finales. Alors même que les résultats sont obtenus à l'aide du logiciel, les articles publiés n'en disent presque rien. Cette invisibilisation du travail computationnel témoigne, selon lui, d'une tension entre l'importance pratique du logiciel et sa faible valeur perçue dans les critères académiques classiques. Hendry évoque quant à lui les résistances qu'il a rencontrées dans les années 1980, notamment face à l'idée d'automatiser l'économétrie. Certains collègues voyaient dans les approches computationnelles une menace à l'autonomie intellectuelle, ou une perte des savoir-faire analytiques traditionnels. Malgré le succès de ses méthodes, cette réception initiale montre que le travail logiciel a longtemps été vu comme secondaire ou mécanique, par rapport à la « vraie » théorie. Epstein évoque moins directement ce problème, mais souligne l'importance d'établir une épistémologie claire pour valoriser l'approche computationnelle. Son travail sur l'« explication générative » vise précisément à établir cette légitimité. Pierse, quant à lui, souligne que la valeur académique d'un outil ne réside pas nécessairement dans sa visibilité directe, mais dans la qualité des résultats empiriques qu'il permet d'obtenir. Ce sont les simulations robustes et les prévisions fiables qui assurent, selon lui, la reconnaissance du travail logiciel – même si celui-ci reste en arrière-plan dans les publications. Fischbacher, enfin, a adopté une stratégie différente, plus explicite et volontariste. Lorsqu'il diffuse z-Tree, son logiciel d'expérimentation, il demande systématiquement aux utilisateurs de citer son article méthodologique. Ce modèle de "citeware", qu'il assume pleinement, a largement fonctionné : l'article de présentation de z-Tree est devenu sa publication la plus citée, avec 12 951 citations en mars 2025. Fischbacher considère cette citation obligatoire non seulement comme un levier de reconnaissance personnelle, mais aussi comme un mécanisme de diffusion efficace du logiciel. Ces trajectoires révèlent une ligne de fracture persistante dans la science économique : les contributions informatiques, même essentielles à la production de résultats, peinent à être reconnues comme telles. Qu'il s'agisse d'outils invisibilisés, de résistances disciplinaires ou de stratégies de légitimation par la citation, le travail logiciel reste souvent en tension avec les normes dominantes de valorisation académique.

Un autre point marquant réside dans la diversité des ancrages institutionnels des économistes - programmeurs interrogés. Leurs parcours s'inscrivent dans des espaces professionnels variés, sans qu'un modèle unique ne se dégage. Cette hétérogénéité illustre la plasticité des profils en économie computationnelle, à la croisée de plusieurs champs ainsi que les usages multiples de leurs outils. Cependant, cela reflète aussi une certaine marginalité dans la discipline : beaucoup naviguent entre champs, sans appartenir pleinement à un sous-domaine. Richard Pierse a mené l'essentiel de sa carrière à la Banque d'Angleterre, où il a contribué à l'intégration des outils économétriques dans la modélisation macroéconomique des politiques monétaires. Son travail illustre la place des économistes-programmeurs dans les institutions publiques, au service de la décision économique. Joshua Epstein, de son côté, a travaillé dans des think tanks (Brookings Institution), avant d'enseigner dans des univer-

sités de premier plan (Johns Hopkins, NYU). Son parcours est marqué par des passages entre sciences sociales, santé publique et modélisation computationnelle, souvent dans des centres de recherche interdisciplinaire, à la frontière entre politique, épidémiologie et économie. Urs Fischbacher a construit sa carrière en deux étapes. D'abord en tant qu'ingénieur logiciel dans des entreprises privé et dans des organismes publiques. Puis dans une seconde partie, à l'université de Zurich puis de Constance, dans des départements de recherche appliquée en économie expérimentale. Il a aussi conçu z-Tree de manière à le rendre utilisable en dehors du milieu académique, notamment dans des contextes de test comportemental, ce qui a pu conduire à des usages hors du champ strictement universitaire. David Hendry a été une figure centrale du département d'économie de l'université d'Oxford, avec une forte influence dans les cercles académiques et dans les institutions de politique économique au Royaume-Uni. Il incarne une position plus classique dans la recherche universitaire, mais avec une attention constante aux applications pratiques de l'économétrie. Theodore Turocy a alterné entre institutions américaines (Caltech, Northwestern, Texas AM) et britanniques (University of East Anglia). Il occupe actuellement une position hybride entre l'université et l'Alan Turing Institute, où il participe à des projets d'intelligence artificielle fondés sur la théorie des jeux computationnelle. Son profil illustre l'ouverture récente de la recherche académique à des collaborations avec l'informatique avancée et l'IA. Agnès Gramain, enfin, a mené une carrière académique plus classique, à Dauphine et à l'université de Lorraine, au sein de laboratoires d'économie théorique et appliquée, en mobilisant beaucoup la modélisation micro économétrique. En somme, les économistes-programmeurs interrogés naviguent entre universités, think tanks, institutions publiques et centres de recherche interdisciplinaire. Cette diversité d'ancrages confirme l'absence de "profil type" et reflète la fluidité professionnelle propre à un champ encore en construction, situé à la croisée de plusieurs disciplines et pratiques

Enfin, les trajectoires des économistes-programmeurs révèlent combien le hasard, les opportunités inattendues et le bricolage intellectuel jouent un rôle structurant dans la naissance de nombreux projets logiciels. Loin d'être issus d'une planification rationnelle ou linéaire, ces outils sont souvent le fruit de rencontres fortuites, de besoins pratiques ou de circonstances expérimentales. Joshua Epstein raconte de manière emblématique la genèse de Sugarscape, conçu avec Robert Axtell lors d'une discussion informelle dans la cafétéria du Brookings Institution. Ils dessinent les premiers éléments du modèle sur des serviettes en papier, dans un esprit d'expérimentation libre, sans cadre théorique préalable ni logiciel existant. Il insiste sur le caractère profondément improvisé de ce moment, soulignant que rien n'était prédéfini : ni la structure du modèle, ni son objectif final. C'est dans ce contexte de création spontanée, encouragé par la liberté offerte par l'institution, que naît un modèle aujourd'hui emblématique de la modélisation agent-based. Theodore Turocy souligne également la contingence de son engagement dans le projet GAMBIT. Il découvre le logiciel

lors d'un cours à Caltech, alors qu'il n'était pas initialement destiné à devenir programmeur. C'est un besoin pédagogique – le manque de logiciels adaptés à l'enseignement de la théorie des jeux – qui l'amène à s'impliquer dans son développement. Ce détour imprévu devient progressivement un pilier de sa carrière, même si ce travail reste longtemps sans financement ni reconnaissance académique. Urs Fischbacher explique que son entrée dans l'économie expérimentale n'a rien d'un plan de carrière : il travaille d'abord comme ingénieur logiciel, puis rejoint par hasard un laboratoire d'économie qui cherche quelqu'un pour développer un outil informatique d'expérimentation. Il saisit l'opportunité, apprend les exigences du champ, et finit par concevoir z-Tree, devenu un standard international. Richard Pierse évoque également des choix de trajectoire façonnés par les circonstances. Ce sont ses emplois d'été au NIESR (National Institute of Economic and Social Research) qui l'initient à la modélisation économétrique appliquée. Ce premier contact, motivé au départ par des considérations alimentaires, l'oriente durablement vers le croisement entre économie et programmation. Ces exemples montrent que, bien souvent, les projets logiciels en économie ne naissent ni d'un plan de carrière ni d'un projet de recherche initialement centré sur la programmation. Ils émergent dans les interstices de l'activité scientifique : par nécessité, par chance, ou par expérimentation. Le hasard des rencontres, les contraintes locales, les besoins techniques non satisfaits et les financements opportuns jouent un rôle déterminant dans la structuration de ces trajectoires.

1.3 Aspects à approfondir

Un aspect essentiel, encore peu interrogé, concerne la finalité des outils développés par les économistes-programmeurs : s'agit-il de logiciels créés avant tout pour répondre à leurs propres besoins de recherche, ou bien de ressources conçues délibérément pour la communauté scientifique plus large ? À la lumière des entretiens, la réponse est souvent double : ces outils naissent d'un besoin individuel et local, puis évoluent vers une diffusion plus vaste, parfois planifiée, parfois inattendue. David Hendry illustre bien cette dynamique. Les logiciels PcGive, PcGets et OxMetrics ont d'abord été développés pour faciliter ses propres travaux en économétrie, notamment pour automatiser l'analyse de modèles dans un cadre rigoureux. Mais dès leur conception, il intègre une dimension pédagogique forte : il s'agit de proposer des outils accessibles aux étudiants, fiables, transparents et documentés. Hendry insiste sur l'importance de garantir une intelligibilité des méthodes, même dans des environnements computationnels complexes, un souci constant de transmission méthodologique. Joshua Epstein présente une trajectoire différente : Sugarscape est au départ un projet exploratoire, sans finalité instrumentale explicite. Conçu comme une expérience intellectuelle libre, le modèle devient progressivement un point de référence en modélisation

agent-based, mobilisé bien au-delà de son cadre initial. Epstein souligne que cette diffusion n'était pas anticipée, et que la légitimité scientifique du modèle a nécessité la construction a posteriori d'une épistémologie rigoureuse, fondée sur la notion d'explication générative. Pour lui, l'outil devient une méthode scientifique à part entière, au service de la compréhension des dynamiques sociales et économiques complexes. Theodore Turocy, quant à lui, navigue entre ces deux pôles. Son travail sur GAMBIT commence comme une réponse à un besoin local : permettre aux étudiants et chercheurs de son entourage d'explorer des équilibres dans les jeux à stratégie mixte. Mais il tente ensuite d'en étendre la portée, cherchant des financements pour stabiliser et diffuser le logiciel. Il reconnaît cependant les difficultés à faire reconnaître cet effort comme une contribution académique à part entière. Ses projets récents avec l'Alan Turing Institute, autour de l'intelligence artificielle, témoignent néanmoins d'une volonté d'ouvrir ces outils à des champs d'application plus vastes, au-delà des frontières strictes de l'économie académique. D'autres cas confirment cette dynamique évolutive. Urs Fischbacher conçoit z-Tree pour répondre aux contraintes très concrètes de l'économie expérimentale, dans son propre laboratoire. Mais le logiciel devient très vite un standard international, utilisé dans des centaines de laboratoires. Fischbacher accompagne activement cette diffusion, en demandant que son article de présentation soit cité systématiquement, une manière de formaliser la reconnaissance académique de cette contribution logicielle. Ces parcours révèlent une tension féconde entre usages personnels et diffusion collective. Le logiciel naît souvent d'un besoin immédiat, parfois urgent, lié à un projet spécifique. Mais son potentiel généralise se révèle ensuite, soit par la communauté, soit par l'auteur lui-même. Ce mouvement progressif, du bricolage individuel à l'outil partagé, souligne que la programmation n'est pas seulement un support technique, mais un vecteur de production de connaissances.

Les trajectoires des économistes-programmeurs montrent également que la programmation ne se réduit pas à une compétence technique ou à un savoir-faire instrumental : elle constitue une forme d'engagement intellectuel à part entière dans la discipline économique. En développant leurs propres outils, ces chercheurs ne se contentent pas d'optimiser leurs méthodes : ils ouvrent de nouveaux espaces de questionnement, remettent en cause des routines établies, et proposent d'autres manières de construire des modèles, de produire des données, ou de simuler des dynamiques complexes. L'informatique devient ainsi un levier de transformation méthodologique, qui rend possible l'exploration de phénomènes mal saisis par les approches analytiques classiques. Joshua Epstein, avec Sugarscape, en donne un exemple emblématique : il s'agit de "prendre un regard brut sur le tout", selon les termes de Murray Gell-Mann, en modélisant des sociétés artificielles peuplées d'agents aux comportements simples, mais générant des dynamiques collectives riches. À travers cette approche, Epstein cherche moins à prédire qu'à expliquer, au sens fort, les régularités sociales à partir de mécanismes "générés". David Hendry, de son côté, conçoit ses outils

comme un moyen de renforcer la rigueur scientifique des tests économétriques, en les rendant reproductibles, transparents et automatisables. Son travail est guidé par une préoccupation méthodologique explicite : garantir que les pratiques empiriques de l'économiste reposent sur des fondements solides, et qu'elles soient accessibles à un public élargi, y compris les étudiants. Urs Fischbacher, en créant z-Tree, ne se limite pas à fournir un logiciel d'expérimentation : il contribue à structurer un champ entier, l'économie expérimentale, en rendant possible des pratiques standardisées, reproductibles et comparables entre laboratoires. Son intervention est donc à la fois technique, institutionnelle et épistémique. En somme, pour ces chercheurs, coder, c'est faire de l'économie autrement. La programmation permet de formaliser autrement, de modéliser différemment, de tester empiriquement de manière plus souple ou plus précise. Elle n'est pas simplement un moyen d'exécution, mais un vecteur de renouvellement disciplinaire. Elle offre aux économistes-programmeurs une manière singulière de participer à l'évolution des normes, des méthodes, et des objets de la science économique contemporaine.

1.4 Conclusion

L'exploration des trajectoires d'économistes-programmeurs permet de mieux comprendre la manière dont l'informatique s'est peu à peu enracinée au cœur des pratiques économiques contemporaines. Derrière les outils devenus familiers, logiciels économétriques, plateformes expérimentales, environnements de simulation, se trouvent des chercheurs aux parcours atypiques, souvent marqués par une hybridité disciplinaire, une formation autodidacte à la programmation, une grande souplesse technique et une forte capacité d'adaptation aux évolutions technologiques. Si leurs profils sont hétérogènes, ils partagent une manière spécifique de faire de l'économie : une pratique située à l'intersection de la modélisation, de l'expérimentation et de l'ingénierie logicielle. En retraçant les parcours d'Agnès Gramain, Urs Fischbacher, David Hendry, Richard Pierse, Robert Axtell, Theodore Turocy et Joshua Epstein, on comprend mieux ce que signifie être un économiste-programmeur : ce n'est pas un sous-métier technique, mais une position hybride, souvent en tension, à l'intersection des savoirs, des outils et des institutions. Ces figures partagent une même ambition : créer des instruments adaptés à leurs objets d'étude, quitte à s'écarter des normes établies ou à inventer des méthodes nouvelles pour explorer les phénomènes économiques. Les économistes-programmeurs jouent ainsi un rôle pivot dans la transformation contemporaine de l'économie, en combinant expertise technique et théorique pour concevoir de nouveaux outils analytiques. Leur profil collectif révèle une communauté professionnelle à la fois diverse et cohérente, marquée par une dynamique d'apprentissage autonome, une adaptabilité constante aux mutations technologiques, et un questionnement critique des

routines académiques dominantes. Leurs contributions, pourtant essentielles à la transformation de la discipline, restent encore partiellement invisibilisées. Leurs trajectoires révèlent des obstacles structurels : faible reconnaissance académique, invisibilisation du travail logiciel, fragilité des financements ou encore absence de soutien institutionnel durable. Ces limites coexistent avec une créativité remarquable, une capacité de bricolage conceptuel et technique, et un rôle actif dans l'évolution des pratiques scientifiques. Leurs contributions sont pourtant déterminantes pour comprendre les mutations contemporaines de la discipline, et pour envisager une économie plus ouverte à la simulation, à l'expérimentation, et à l'innovation logicielle. Les obstacles à la valorisation académique, les tensions entre bricolage individuel et diffusion collective, ou encore les inégalités de genre dans l'accès à la programmation, montrent que l'économie computationnelle ne se développe pas hors sol, mais dans un contexte social, institutionnel et épistémique qu'il faut continuer à interroger.

La figure d'Agnès Gramain montre tout de même la limite de ce portrait de l'économiste-programmeur : elle nous rappelle que notre échantillon comporte peu de femmes, qu'il existe des économistes-programmeurs aux trajectoires différentes, formés dans des cadres très structurés — ici celui de l'ENSAE — et orientés vers des carrières administratives, où la programmation est avant tout un outil fonctionnel au service de la modélisation et de la production statistique, plus qu'un espace d'innovation autonome.

Après avoir étudié les figures de l'économiste-programmeur, la prochaine étape de ce travail portera sur les logiciels eux-mêmes : leurs usages, leurs architectures, leurs trajectoires de diffusion. On s'appuiera notamment sur Renfro, 2004[20] Il s'agira d'analyser comment ces outils sont conçus, ce qu'ils permettent (ou empêchent), et comment ils structurent les formes de preuve et les régimes d'autorité dans la discipline. À travers cette enquête sur les instruments, c'est toute une histoire technique et intellectuelle de l'économie contemporaine qui se dessine. Enfin, nous approfondirons la recherche du côté des épistémologies sous-jacentes à chaque famille de méthodes computationnelles, modélisation agent, expérimentation assistée par ordinateur, automatisation des tests économétriques, afin d'en tirer des leçons pour l'avenir de la discipline. Ce détour par les outils et celles et ceux qui les fabriquent n'a rien d'anecdotique : il ouvre sur une réflexion plus large sur les conditions concrètes de production des savoirs économiques aujourd'hui. Finalement, dans une dernière partie, nous nous intéresserons aux enseignements à tirer de l'étude de ces six économistes-programmeurs : quelles implications pour la formation, la reconnaissance ou l'organisation des carrières en économie ? Par exemple, que signifie aujourd'hui le fait d'intégrer des cours de programmation dans les cursus d'économistes ? S'agit-il d'un simple outillage technique ou d'un apprentissage qui transforme profondément les manières de penser, de modéliser et de faire science ?

Bibliographie

[1] Robert Axtell. *Interview with Robert Axtell, Conducted by Romain Plassard and Francesco Sergi on April 9th 2024*. application/pdf,audio/mpeg. Version 2. NAKALA - https ://-nakala.fr (Huma-Num - CNRS), 2025. doi : 10.34847/NKL.AFB92MBJ. url : https://nakala.fr/10.34847/nkl.afb92mbj (visité le 21/04/2025).

[2] Roger E. Backhouse et Béatrice Cherrier. « "It's Computers, Stupid!" The Spread of Computers and the Changing Roles of Theoretical and Applied Economics ». In : *History of Political Economy* 49 (Supplement 2017), p. 103-126. issn : 0018-2702, 1527-1919. doi : 10.1215/00182702-4166287. url : https://read.dukeupress.edu/hope/article/49/Supplement/103/133606/Its-Computers-Stupid-The-Spread-of-Computers-and (visité le 15/10/2024).

[3] Victoria Bateman et Erin Hengel. « The Gender Gap in UK Academic Economics 1996-2018 : Progress, Stagnation and Retreat ». In : *Œconomia. History, Methodology, Philosophy* 13–2 (13–2 1er juin 2023), p. 163-200. issn : 2113-5207. doi : 10.4000/oeconomia.15193. url : https://journals.openedition.org/oeconomia/15193 (visité le 23/04/2025).

[4] Sylvia Beyer. « Why Are Women Underrepresented in Computer Science ? Gender Differences in Stereotypes, Self-Efficacy, Values, and Interests and Predictors of Future CS Course-Taking and Grades ». In : *Computer Science Education* 24.2–3 (3 juill. 2014), p. 153-192. issn : 0899-3408. doi : 10.1080/08993408.2014.963363. url : https://doi.org/10.1080/08993408.2014.963363 (visité le 23/04/2025).

[5] Marcel Boumans et al. « The Computerization of Economics : Three Lessons for Economics ». In : *Œconomia. History, Methodology, Philosophy* 13–3 (13–3 1er sept. 2023), p. 637-655. issn : 2113-5207. doi : 10.4000/oeconomia.15684. url : https://journals.openedition.org/oeconomia/15684 (visité le 15/10/2024).

[6] Béatrice Cherrier, Aurélien Saïdi et Francesco Sergi. « "Write Your Model Almost as You Would on Paper and Dynare Will Take Care of the Rest !" A History of the Dynare Software ». In : *Œconomia. History, Methodology, Philosophy* 13–3 (13–3 1er sept.

2023), p. 801-848. ISSN : 2113-5207. DOI : 10.4000/oeconomia.16123. URL : https://journals.openedition.org/oeconomia/16123 (visité le 15/10/2024).

[7] *Computer Ownership up Sharply in the 1990s*. Bureau of Labor Statistics. URL : https://www.bls.gov/opub/ted/1999/apr/wk1/art01.htm (visité le 25/04/2025).

[8] Joshua Epstein. *Interview with Joshua Epstein, Conducted by Romain Plassard and Francesco Sergi on April 11th, 2024*. application/pdf,audio/mpeg. Version 2. NAKALA - https ://nakala.fr (Huma-Num - CNRS), 2024. DOI : 10.34847/NKL.034BBFRN. URL : https://nakala.fr/10.34847/nkl.034bbfrn (visité le 20/01/2025).

[9] Katrina Falkner et al. « Gender Gap in Academia : Perceptions of Female Computer Science Academics ». In : *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ITICSE '15 : Innovation and Technology in Computer Science Education Conference 2015. Vilnius Lithuania : ACM, 22 juin 2015, p. 111-116. ISBN : 978-1-4503-3440-2. DOI : 10.1145/2729094.2742595. URL : https://dl.acm.org/doi/10.1145/2729094.2742595 (visité le 23/04/2025).

[10] Urs Fischbacher. *Interview with Urs Fischbacher, Conducted by Dorian Jullien and Thomas Delcey on November 17th 2023. Oral Histories of Economics*. application/pdf,audio/mpeg,image/p Version 2. NAKALA - https ://nakala.fr (Huma-Num - CNRS), 2024. DOI : 10.34847/NKL.B03EW75I. URL : https://nakala.fr/10.34847/nkl.b03ew75i (visité le 04/04/2025).

[11] Agnès Gramain. *Interview with Agnès Gramain, conducted by Francesco Sergi and Pierrick Dechaux on January 16th, 2023. Oral Histories of Economics*. audio/mpeg,application/pdf. Version 2. NAKALA - https ://nakala.fr (Huma-Num - CNRS), 2024. DOI : 10.34847/NKL.387FU51R. URL : https://nakala.fr/10.34847/nkl.387fu51r (visité le 03/12/2024).

[12] David Hendry. *Interview with David Hendry, Conducted by Romain Plassard and Francesco Sergi on March 19th and April 26th 2024. Oral Histories of Economics*. application/pdf,video/mp4. Version 2. NAKALA - https ://nakala.fr (Huma-Num - CNRS), 2024. DOI : 10.34847/NKL.86CFW03E. URL : https://nakala.fr/10.34847/nkl.86cfw03e (visité le 20/01/2025).

[13] James Howison et James D. Herbsleb. « Scientific Software Production : Incentives and Collaboration ». In : *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*. CSCW '11. New York, NY, USA : Association for Computing Machinery, 19 mars 2011, p. 513-522. ISBN : 978-1-4503-0556-3. DOI : 10.1145/1958824.1958904. URL : https://doi.org/10.1145/1958824.1958904 (visité le 24/04/2025).

[14] Dorian Jullien. *Interviews : Some Methodological and Historiographical Issues of Oral Sources*. 25 jan. 2018. doi : 10.2139/ssrn.3116213. url : https://papers.ssrn.com/abstract=3116213 (visité le 13/01/2025). Prépubl.

[15] Shulamit Kahn. « Gender Differences in Academic Career Paths of Economists ». In : *The American Economic Review* 83 (2, 1993), p. 52-56. JSTOR : 2117639. url : http://www.jstor.org/stable/2117639.

[16] John A. N. Lee. « History of Computing in Education : An Overview ». In : *History of Computing in Education*. Sous la dir. de John Impagliazzo et John A. N. Lee. T. 145. New York, NY : Springer US, 2004, p. 1-16. isbn : 978-1-4020-8135-4 978-1-4020-8136-1. doi : 10.1007/1-4020-8136-7_1. url : http://link.springer.com/10.1007/1-4020-8136-7_1 (visité le 22/04/2025).

[17] Cory Merow et al. « Better Incentives Are Needed to Reward Academic Software Development ». In : *Nature Ecology & Evolution* (27 fév. 2023). doi : 10.1038/s41559-023-02008-w.

[18] Mary S Morgan. « The World in the Model ». In : ().

[19] Richard Pierse. *Interview with Richard Pierse, Conducted by Francesco Sergi, Pierrick Dechaux, and Thomas Delcey on March 11th 2024. Oral Histories of Economics*. application/pdf,audio/mpeg. Version 2. NAKALA - https ://nakala.fr (Huma-Num - CNRS), 2024. doi : 10.34847/NKL.2317P125. url : https://nakala.fr/10.34847/nkl.2317p125 (visité le 20/01/2025).

[20] Charles G. Renfro. « A Compendium of Existing Econometric Software Packages ». In : *Journal of Economic and Social Measurement* 29.1–3 (20 juill. 2004). Sous la dir. de Charles G. Renfro, p. 359-409. issn : 18758932, 07479662. doi : 10.3233/JEM-2004-0225. url : https://journals.sagepub.com/doi/full/10.3233/JEM-2004-0225 (visité le 17/12/2024).

[21] John Schmitt et Jonathan Wadsworth. *Give PC's a Change : Personal Computer Ownership and the Digital Divide in the United States and Great Britain*. Discussion Paper 526. London : Centre for Economic Performance, 2002. 42 p. isbn : 978-0-7530-1924-5.

[22] Theodore Turocy. *Interview with Theodore Turocy, Conducted by Dorian Jullien, Thomas Delcey, and Francesco Sergi on March 8th 2024. Oral Histories of Economics*. audio/mpeg,application/pdf. Version 3. NAKALA - https ://nakala.fr (Huma-Num - CNRS), 2024. doi : 10.34847/NKL.50A7980K. url : https://nakala.fr/10.34847/nkl.50a7980k (visité le 20/01/2025).