International Review of Economics Education 52 (2026) 100347

Contents lists available at ScienceDirect

International Review of Economics Education

journal homepage: www.elsevier.com/locate/iree

Programming and the economics curriculum: Evidence from undergraduate student attitudes☆

Nigar Hashimzade [a,1], Oleg Kirsanov [b,2], Tatiana Kirsanova [b,*]

[a] Brunel University London, United Kingdom
[b] University of Glasgow, United Kingdom

ARTICLE INFO

JEL classification:
A22
I23
J24
O33

Keywords:
Programming skills
Economics education
Undergraduate curriculum
Employability
Survey analysis

ABSTRACT

A significant and growing proportion of graduate economics job advertisements in the UK and other countries mentions programming skills. Do undergraduates see that shift, and do they want their degree programme to keep pace? We answer these questions with the first multi-year survey of student attitudes towards programming, administered to 317 economics majors in Years 2–4 at a UK university. The support for curricular integration is overwhelming: 92% favours adding programming and 55% favour making it mandatory. Further analysis shows that support for a compulsory course is strongest among final-year and international students, while the preference for earlier programming exposure is the highest among those still uncertain about their career plans. Confidence in programming skills remains low even in the final year, and students regard generic computer science modules as poor substitutes for economics-focused instruction. Taken together, the survey results strongly suggest that in undergraduate economics programming should be introduced early on and that it should be embedded in discipline-specific content. Moreover, its teaching should be supported throughout the economics curriculum to meet the students' demand, enhance their employability, and close the skill gaps before they become entrenched.

1. Introduction

The economic profession increasingly relies on computational modelling and big data analysis, and graduate employers routinely list programming skills among their desired competencies. This shift is well documented in employer surveys (Economics Network, 2019), calls for enhanced quantitative skills (NACE, 2024), labour market analyses (Blumenstyk, 2016), and policy reviews (CBI, 2022; IFS, 2024). It is further reflected in educational initiatives, such as CORE Project (2024), and in recent American Economic Association reports (AEA, 2024). A significant and growing proportion of graduate economics job advertisements in the UK mention programming languages, with Python skills being most sought-after, and languages such as R, Stata, SQL, Matlab and Julia also often required by central banks, economic consultancies, and financial services.

☆ The views expressed are those of the authors and do not necessarily reflect those of Brunel University of London and the University of Glasgow. We are grateful to the editor and two anonymous referees for their constructive comments and suggestions, which substantially improved the paper. We also thank colleagues in the Economics group at the University of Glasgow, and in particular Wenya Chen, Paulina Navrouzoglou, and Geethanjali Selvaretnam, for helpful discussions and feedback at an earlier stage of this project.

* Correspondence to: Economics, Adam Smith Business School, University of Glasgow, Glasgow G11 6EY, United Kingdom.
E-mail addresses: nigar.hashimzade@brunel.ac.uk (N. Hashimzade), oleg.kirsanov@glasgow.ac.uk (O. Kirsanov), tatiana.kirsanova@glasgow.ac.uk (T. Kirsanova).
[1] Department of Economics, Finance and Accounting, Brunel University of London, United Kingdom.
[2] Economics, Adam Smith Business School, University of Glasgow, United Kingdom.

https://doi.org/10.1016/j.iree.2026.100347
Received 12 November 2025; Received in revised form 12 April 2026; Accepted 16 April 2026
Available online 20 April 2026

Yet, programming appears — if at all — only in optional electives in advanced or specialisation stages of undergraduate study or behind the drop-down menus of econometrics packages in most UK programmes. That fragmented provision lags behind the growing demands of the labour market and the expanding scope of undergraduate dissertations, which frequently require computational skills. The question is no longer whether programming is valuable, but how early and systematically it should be integrated into the curriculum to prepare students for economics-related careers.

Several studies have explored how programming can be incorporated into economics teaching, through dedicated modules, coded examples, and hands-on modelling exercises (e.g., Gorry and Gilbert, 2015; Jenkins, 2022; Solis-Garcia, 2021; Luedtke, 2023; Kuroki, 2021, 2023, 2025). Collectively, this literature establishes that programming can be taught effectively within economics degrees using a range of pedagogical approaches and computational tools. However, the emphasis has largely been on how programming is delivered within individual modules rather than when it should be introduced across the degree programme, or which groups of students may be disadvantaged by delayed or optional exposure. As a result, the issues of sequencing, confidence formation, and equity across student backgrounds remain under-explored. Addressing these gaps requires evidence not only on teaching practices, but also on students' own perceptions of timing, preparedness, and access to programming skills.

A recent contribution by Hashimzade et al. (2025) moves this discussion to the level of curriculum design. They argue that programming should be placed on the same curricular footing as mathematics, statistics, and econometrics in undergraduate economics education. Their case rests on three linked arguments: (i) programming is increasingly integral to contemporary research and policy analysis, (ii) graduate recruiters reward demonstrable programming competence, and (iii) postponing these skills until postgraduate study entrenches inequalities in access to dynamic careers in the modern economy. While Hashimzade et al. (2025) detailed the curriculum design and implementation costs for introducing programming in economics degrees at a typical UK university, the present paper serves a distinct and complementary purpose: it provides the missing empirical evidence on student perceptions—evidence that can inform how and when programming should be taught, based on students' experiences and career expectations.

Specifically, this paper differs from Hashimzade et al. (2025) in three fundamental ways. First, it adopts a different perspective. While Hashimzade et al. (2025) develop a faculty- and institution-side case for curriculum reform, with the primary focus on feasibility, sequencing, and pedagogical design, the present study brings in the student voice, documenting students' attitudes, expectations, and perceived constraints. Second, it offers distinct substantive insights. Rather than inferring optimal sequencing from curriculum logic, we directly elicit students' preferences over the timing of programming instruction, uncover persistent confidence gaps even among final-year students, and identify systematic differences across groups, including international students, academically strong students, and those with initially uncertain career plans. These patterns speak directly to issues of equity and access that cannot be addressed through curriculum design alone. Third, the paper makes a methodological contribution by using a multi-cohort undergraduate survey combined with both descriptive and regression analyses, allowing us to separate prior exposure to programming from preferences over curricular timing and to examine which student characteristics predict support for early and compulsory integration.

Taken together, the shift in perspective, the focus on timing and equity, and the use of student-level evidence clarify why the student perspective is essential for curriculum reform. Research on curriculum co-design shows that incorporating student views can improve engagement, satisfaction, and alignment with learning needs (Bovill and Bulley, 2011; Healey et al., 2014). Related work in higher education shows that early exposure and confidence formation are central to students' engagement with quantitative subjects. Evidence from mathematics and engineering education demonstrates that students' self-confidence and early experiences with technical material strongly shape persistence, progression, and perceived preparedness, particularly for students entering with diverse educational backgrounds (Parsons et al., 2009; Bamforth et al., 2007). In the context of programming as a potential core requirement, student attitudes are not merely a matter of acceptance or resistance, but provide concrete information about sequencing, perceived difficulty, confidence formation, and unequal access to skills across groups. Understanding how students experience programming and when they believe it should be introduced is, therefore, critical for designing curricula that are both pedagogically effective and equitable.

This paper addresses these gaps by incorporating student perspectives into the curriculum design process. Using survey data from 317 economics undergraduates across three cohorts at a UK university, we examine two primary questions: (i) How strong is student support for embedding programming in the Economics curriculum? (ii) Do students prefer earlier exposure to programming? In addition, we explore whether support for programming differs across students with different characteristics, and what factors drive preference for earlier exposure.

In what follows, "programming" does not refer to software engineering in the computer-science sense. Rather, we use the term in the narrower economics-applied sense: writing and modifying code to handle data, clean and reshape datasets, implement econometric or statistical estimation, produce visualisations, solve and simulate economic models, and evaluate counterfactual or policy scenarios. The focus of the paper is therefore on programming as a practical tool for producing and interpreting economic evidence within the Economics curriculum.

In this survey, 92% of the respondents favour including programming in the Economics curriculum and 55% would make it compulsory. Support is the strongest among final-year and international students. Further analysis confirms that both the year of study and the international status significantly predict backing for a mandatory course. Moreover, students at the earlier stage in the degree programme are more likely to perceive importance of programming for their later career choice. This suggests that curriculum co-design ensuring earlier exposure to programming will enhance student experience and improve their confidence in the professional direction.

Our results also show that students distinguish between principal uses of programming in economics: interest in data-analytic applications (visualisation and econometrics) is higher in the earlier years, while interest in model-based applications (solving and simulating economic models) rises later in the degree programme.

Taken together, these empirical findings provide support for the case made by Hashimzade et al. (2025) regarding the curriculum: integrating programming early and teaching it through economics-specific tasks, such as handling and analysing data, modelling dynamic systems, and conducting policy simulations, would align with strong students' demand, support their skill development, and prepare them for economics-related careers.

The paper proceeds as follows. Section 2 describes the survey's methodology and data collection process. Section 3 presents descriptive findings, focusing on employability beliefs, attitudes towards curriculum reform, and the effect of timing of exposure. Section 4 complements this by using regression analysis to examine how students' responses relate to their academic background and experiences. Section 5 draws together the main insights and considers implications for curriculum design. Section 6 concludes.

2. Data and survey design

The data used in this study come from a survey conducted among undergraduate economics students at the University of Glasgow during the Spring term of the 2024–25 academic year.[3] The survey was designed to capture students' views on programming, including their exposure to programming languages, confidence in programming skills, perceived relevance to employability, and preferences for curriculum integration. Students were also asked about their background, career intentions, and self-assessed competitiveness in the job market.

The survey was conducted at a Russell Group[4] UK university operating under the four-year Scottish degree structure. The student body draws from a broad applicant pool, and the Economics curriculum follows a standard pattern comparable to that of other Russell Group universities in the UK. The achieved sample is representative of the full undergraduate Economics cohort in terms of gender and nationality, allowing cautious generalisation of the findings to similar UK institutions.

In Scotland, undergraduate degrees typically span four years, with the first two years often referred to as "pre-Honours" and the final two as "Honours years", when students specialise in their chosen discipline. This structure differs from the rest of the UK, where Economics degrees typically span three years without such a formal distinction. In line with the Scottish degree structure, this study classifies students by their second, third, and fourth years of their undergraduate degree. Our second-year students broadly correspond to the first-year students in a three-year UK degree, while our third- and fourth-year students correspond to second- and third-year students, respectively. While the Scottish four-year structure offers an additional pre-Honours year, this may in fact sharpen students' views about the timing of skill acquisition, as decisions about specialisation and dissertation preparation are made earlier and more explicitly. Importantly, the central distinction examined in this paper: early versus late exposure to programming, — maps directly onto the sequencing choices faced by three-year UK programmes and US undergraduate curricula, where similar decisions arise within a more compressed timeframe. For international readers, particularly in the US context, we note that this four-year structure is broadly comparable to the standard four-year undergraduate degree, which similarly progresses from a broad-based study in the early years to more specialised coursework in the final stages.

First-year students were not surveyed. Second-year students completed a paper version of the questionnaire during a scheduled lecture session at the beginning of the Spring semester, in early January 2025. The session formed part of a mandatory core course for all second-year students who take Economics as one of three subjects. Attendance was typical for a large undergraduate lecture, with approximately 160–180 out of 280 enrolled students present. At that point, students had completed three semesters of coursework, typically in Economics and two other subjects from the College of Social Sciences, where technical training is less common. Economics (and Finance) were thus relative outliers in terms of quantitative content. Since survey participation depended primarily on lecture attendance rather than prior interest in programming, self-selection bias in this subgroup is unlikely to be related to the topic of the survey.

Third- and fourth-year students completed the survey electronically later in the semester. The survey opened shortly before the dissertation submission deadline and remained accessible through the Easter vacation, closing before the start of the Summer Examination Period. By that stage, third-year students had completed two semesters of specialised Economics coursework, while fourth-year students were nearing the end of a six-month dissertation project and typically had clearer career or postgraduate plans.

Participation was voluntary across all year groups, but modes of administration differed, leading to differences in the response rates. Among second-years, the survey was distributed in paper form during a compulsory lecture session; while completing it remained voluntary, compliance was very high among students present. Among third- and fourth-years, the survey was conducted electronically, and response rates were lower, potentially favouring students with greater interest in technical skills. However, the combined sample broadly mirrors the programme's overall gender and nationality composition (see Table B.1 in Appendix B). A chi-squared test indicates that the only group slightly under-represented is international second-year students.

In total, we received 317 responses: 154 from second-year students, 84 from third-year students, and 79 from fourth-year students. The second-year sample comprises roughly 54% of the enrolled cohort (estimated class size: 286), while the response rate among Year 3 and Year 4 students was approximately 30% and 38%, respectively (class sizes: 281 and 210).[5]

[3] The survey and its procedures received ethical approval from the University of Glasgow College of Social Sciences Research Ethics Committee. Participation was voluntary and anonymous, and all respondents provided informed consent before submitting the questionnaire.

[4] The Russell Group represents 24 research-intensive universities located in every region and nation of the UK. See https://www.russellgroup.ac.uk/ for details.

Table 1
Summary of the survey sample.

	Year 2	Year 3	Year 4
Respondents (N)	154	84	79
Female (%)	42.9	42.9	36.7
International (%)	37.7	40.5	29.1
Response rate (%)	53.8	29.9	37.6

Notes: An extended comparison with administrative cohort composition is reported in Appendix B, Table B.1.

Table 1 provides a compact summary of the sample used in the analysis. The gender composition is broadly balanced across cohorts, and the home/international split is also close to the underlying programme composition, with the only notable deviation being a slight under-representation of international students in Year 2. An extended comparison with administrative cohort composition is reported in Appendix B.

Although the paper and online versions of the survey were closely aligned in content, small differences in wording and structure existed across versions. These are documented in Appendix A, and flagged in the analysis where relevant. Additional background on cohort structure, administrative definitions, and sample classification can be found in Appendix B.

The merged dataset allows us to examine variation by year of study, home vs international status, confidence levels, exposure types, and career expectations. The next section presents the main descriptive patterns in the data.

Unless otherwise stated, descriptive statistics use all available responses for the relevant survey item, so the underlying number of observations may vary slightly across panels because of occasional item non-response. By contrast, the regression models in Section 4 use a common estimation sample excluding observations with missing data either on the dependent variable or any of the included covariates. We report the resulting regression sample sizes explicitly in each table and explain the exclusions in the corresponding notes.

3. Descriptive and thematic evidence

This section presents evidence from four composite figures based on our student survey, covering second-, third-, and fourth-year undergraduate economics students. The figures were designed to isolate key dimensions of student views on programming: its relationship to employability, patterns of exposure to programming languages, interest in specific applications of programming, and the perceived adequacy of current provision. The figures also allow for comparisons by year of study and home/international status. We distinguish throughout between the UK and international students not only because these groups differ in prior exposure and confidence, but also because they may face different labour-market constraints, information sets, and incentives to acquire transferable technical skills. This makes the comparison substantively relevant for curriculum design rather than merely descriptive.

3.1. Programming and employability

Fig. 1 summarises students' responses to three questions capturing how programming is perceived to affect employability and career preparation. Together, these items provide insight into whether students view programming as a broadly relevant skill and whether its timing within the Economics degree matters for career outcomes. For clarity, the three panels correspond to distinct survey items: `competitive` asks whether programming improves students' competitiveness in the labour market; `skills_lack` asks whether a lack of programming may limit career opportunities; and `skills_early` asks whether earlier programming exposure would have influenced career preparation or career choices. Full wording of all survey questions is reported in Appendix A.

Responses to `competitive` show that a clear majority of students believe programming enhances their competitiveness in the labour market. This perception is particularly strong among international students and becomes more pronounced in later years of study, suggesting that students increasingly recognise programming as a differentiating skill as they approach graduation.

The panel `skills_lack` indicates that many students believe a lack of programming skills could limit their opportunities in economics-related careers. This concern is again more prevalent among international students, consistent with the view that programming serves as both a productive skill and a signal of employability in competitive and international labour markets.

Finally, `skills_early` directly addresses the importance of timing. A substantial share of students, especially international students and those in the final year, report that learning programming earlier in their Economics studies would have influenced their career preparation or choices. This pattern highlights that perceived employability benefits are not only linked to acquiring programming skills per se, but also to when those skills are developed during the degree.

Taken together, the evidence suggests that students view programming as a core employability skill whose value depends critically on early and structured integration within the Economics curriculum, rather than as an optional or late-stage technical add-on.

[5] Final response rates were calculated using administrative enrolment data from the UG Economics programme. The relatively low response rate among third-year students likely reflects the substantial number of students on a year abroad (estimated at over 10% of the cohort) who may not have engaged with non-essential communication from their home university.

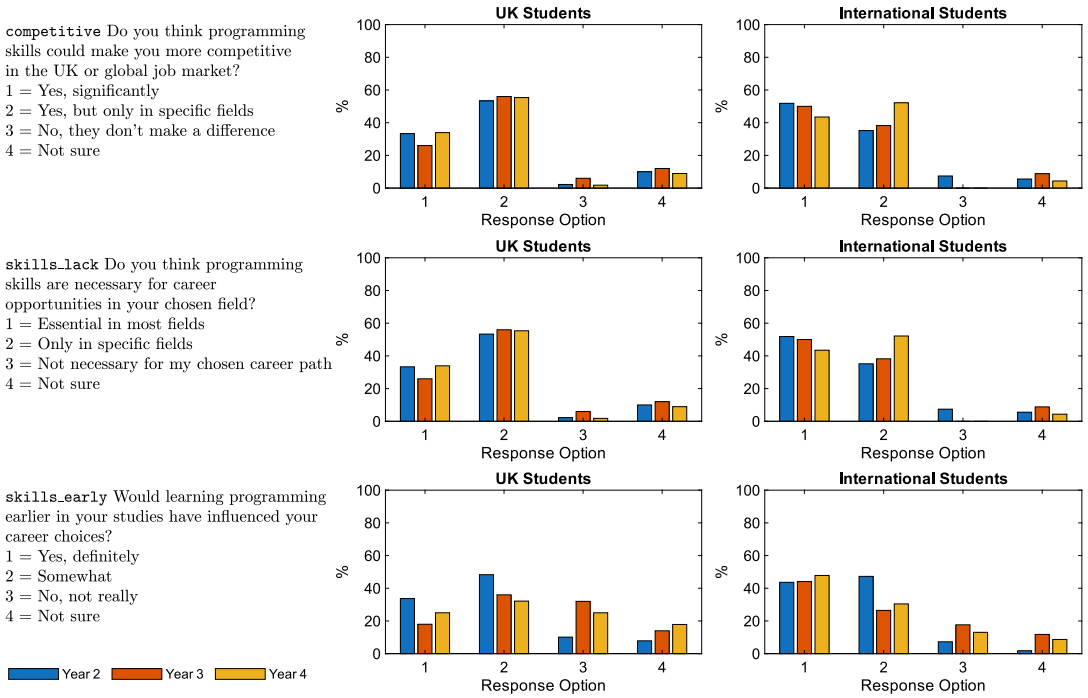

Fig. 1. Programming and employability.

3.2. Exposure to programming languages

This subsection reports students' *actual prior exposure* to programming languages, based on the multi-response survey item `lang`: "Which programming languages have you been exposed to during your studies or on your own? Mark all that apply". Fig. 2 summarises these responses across year groups and by home/international status.

The figure presents six pie charts: three for UK students (Years 2, 3, and 4) and three for international students. In each case, responses are grouped into three categories: exposure to at least one "core" programming language, selection of "Other" *without any core language*, or "None". We use "core" to denote languages that are standard and recognisable across the survey versions as general-purpose or widely used computational tools in economics: Python, R, and MATLAB in all versions, with Java and C++ additionally listed in the Year 3 and Year 4 online survey, which was designed at a later stage of the project. Since the Year 2 could choose only from among three core languages (Python, R, and MATLAB), in addition to "None" or "Other", where the latter did not ask for additional information, some of these students who chose "Other" may have had experience of independent learning of programming languages such as Java or C++, or were exposed to statistical software such as SPSS or Stata, which are scriptable tools, or domain-specific languages, rather than full programming languages. The dominant response, however, is clear: 56% of UK students and 44% of international students reported no exposure to either a core language, or "Other".

In contrast, the Year 3 and 4 surveys included five core languages (adding Java and C++) and allowed students to specify "Other". Among those who selected "Other" *only*, only one student listed Mathematica as an additional programming language. All others named Stata, SPSS, SQL, or HTML. Since these responses were given without selecting a core language, they likely reflect a broader, and, possibly inaccurate, understanding of what programming language is. This contrasts with students who selected both a core language and an "Other" tool, which could be read as a desire to signal additional competencies.

When comparing across year groups, a clear trend emerges. Among Year 2 students, a majority report no exposure to any of the core programming languages, though international students show slightly higher rates of exposure than their UK peers. By Year 3, the share reporting no exposure falls noticeably, and by Year 4, most students have encountered at least one core language.[6]

International students appear to be roughly one year ahead: their Year-2 distribution resembles UK Year-3, and by Year-4 very few select "None" and or name Stata as programming language. Early exposure appears both broader and clearer for this group.

[6] About 5 percent of fourth-year students (both UK and international) report exposure to R only, through an econometrics course taken in their second year. This was a large, mandatory course, and, strictly speaking, all students in this cohort encountered R, but only through illustrative regression and plotting examples in tutorials, without formal instruction in the language itself. The fact that most students do not report R as a standalone language suggests they did not view this exposure as meaningful. In almost all other cases, R is reported alongside other languages. Even excluding the small group of "R-only" students from the Exposed to Core 5" category, the remaining proportion remains substantially greater than the corresponding segment in Year 3.

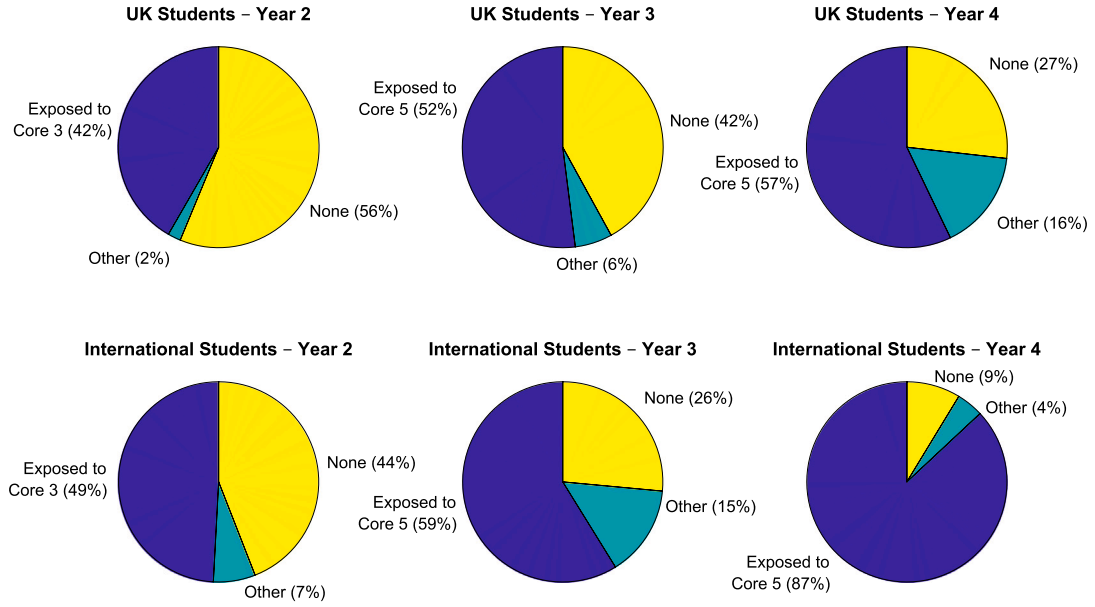

Fig. 2. Exposure to programming languages.

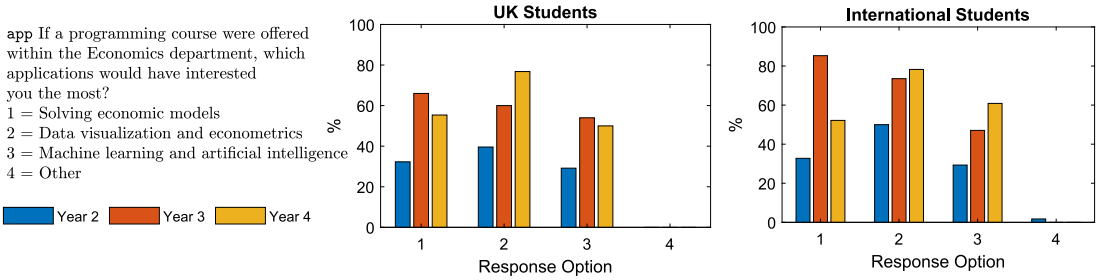

Fig. 3. Interest in programming applications.

This pattern may partly reflect greater pursuit of programming experience outside the economics curriculum, as further discussed in Section 3.4.

These trajectories reinforce the regression analysis results presented in Section 4: students, and especially the internationals, who encounter programming early are more supportive of mandatory curriculum provision and less likely to believe that earlier exposure would have influenced their preparation for economics-related careers.

3.3. Interest in applications

Fig. 3 summarises responses to a multiple-choice question asking students which applications of programming would interest them most if a programming course were offered within Economics (app). The response options distinguish three broad uses: solving economic models, data visualisation and econometrics, and machine learning/AI. The question therefore captures not whether students have already done these activities, but which kinds of economics-related programming they would most value learning.

Interest in using programming to solve economic models increases in Year 3, particularly among international students. This pattern likely reflects a shift in exposure: third-year students encounter more technically demanding material, including dynamic models and simulation-based analysis, often for the first time. By Year 4, interest in model-based applications remains high but stabilises, while interest in machine learning and AI increases modestly, peaking in the final year. These trends suggest that students' preferences evolve with both experience and a more realistic understanding of the value of tools and applications.

Overall, students want programming for concrete, economics-focused tasks: predominantly for data work, followed by formal modelling, rather than as a generic technical skill. This emphasis on analytics-oriented applications resonates with recent developments in economics education. As documented in Hashimzade et al. (2025), a growing number of UK economics programmes have introduced computational and data-oriented courses at the honours level, often centred on applied econometrics, data analysis, and simulation-based methods. At the same time, the expansion of postgraduate programmes in data analytics and data analytics for

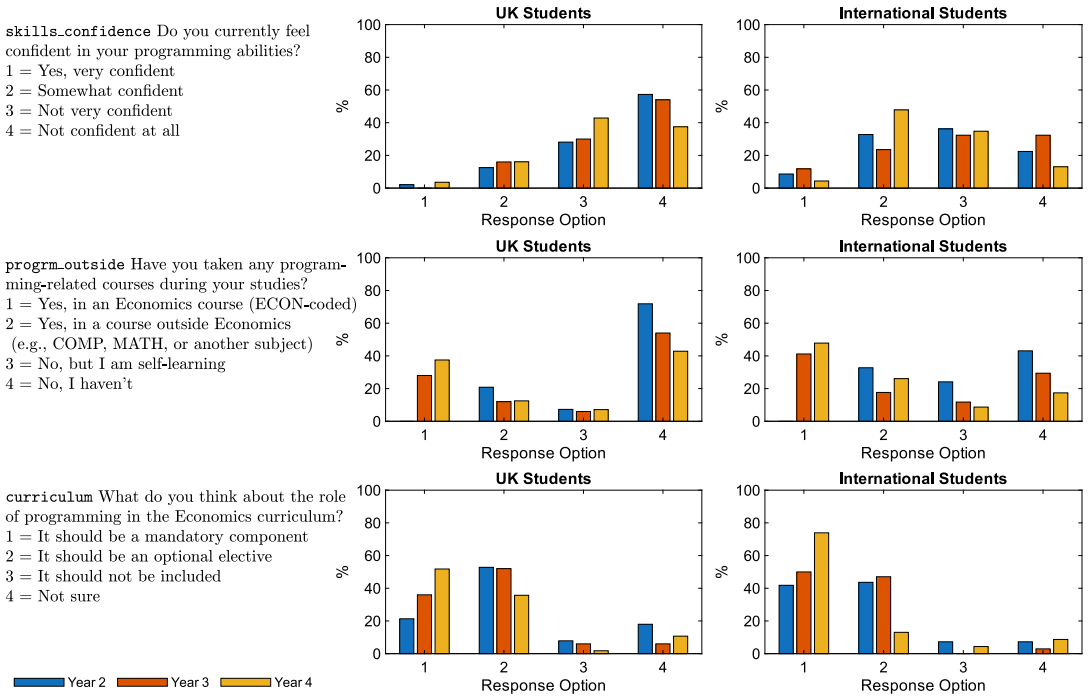

Fig. 4. Building programming skills.

economics and finance reflects sustained demand for data literacy in economics-related careers. Taken together, these developments suggest that students' interest in programming for data-analytic and applied economic tasks aligns with both current curriculum design choices and external labour-market incentives.

3.4. Confidence, provision, and curriculum reform

Fig. 4 presents three survey items capturing students' confidence in programming, their exposure to programming outside Economics, and their support for curricular integration.

Responses to `skills_confidence` reveal persistently low levels of confidence in programming across all years of study. Even among final-year students, fewer than one-third describe themselves as even moderately confident. This pattern suggests that limited or late exposure to programming does not translate into a strong sense of preparedness by graduation.

The panel `progrm_outside` shows that international students are substantially more likely than UK students to have pursued programming courses outside the Economics curriculum, particularly in Years 3 and 4. However, as later regression analysis demonstrates, such external exposure does not significantly reduce students' preference for earlier programming instruction within Economics, indicating that incidental or non-discipline-specific training may not substitute for structured curricular integration.

Support for curriculum reform is consistently high. Approximately 90% of students favour embedding programming within the Economics curriculum (`curriculum`), with a substantial proportion advocating mandatory inclusion. Support is strongest among international students and among students in their final year, reflecting both accumulated experience and heightened awareness of labour market requirements.

Overall, these responses point to a clear mismatch between students' demand for programming skills and the current modes of provision. Students do not view external courses or self-directed learning as adequate substitutes for early, structured, and economics-specific programming instruction embedded within the degree.

3.5. Robustness checks: Sceptics and typical career aspirations

To test whether support for programming is limited to enthusiasts, we first identified a group of sceptical students as those who believe programming is useful only in specific fields. A student is classified as `sceptical` if they selected response category 2 ("only in specific fields") on any of three survey items: `employability`, `skills_lack`, or `competitive`. Under this broad definition, 235 students, or about two-thirds of the sample, are classified as sceptical.

To complement this check, we perform a parallel analysis focusing on students who plan to enter Finance and Banking, the most common career destination reported in our survey. This group is identified using the variable `work_path`, which classifies intended employment sectors into five categories: Finance and Banking, Accounting and Professional Services, Policy and Government,

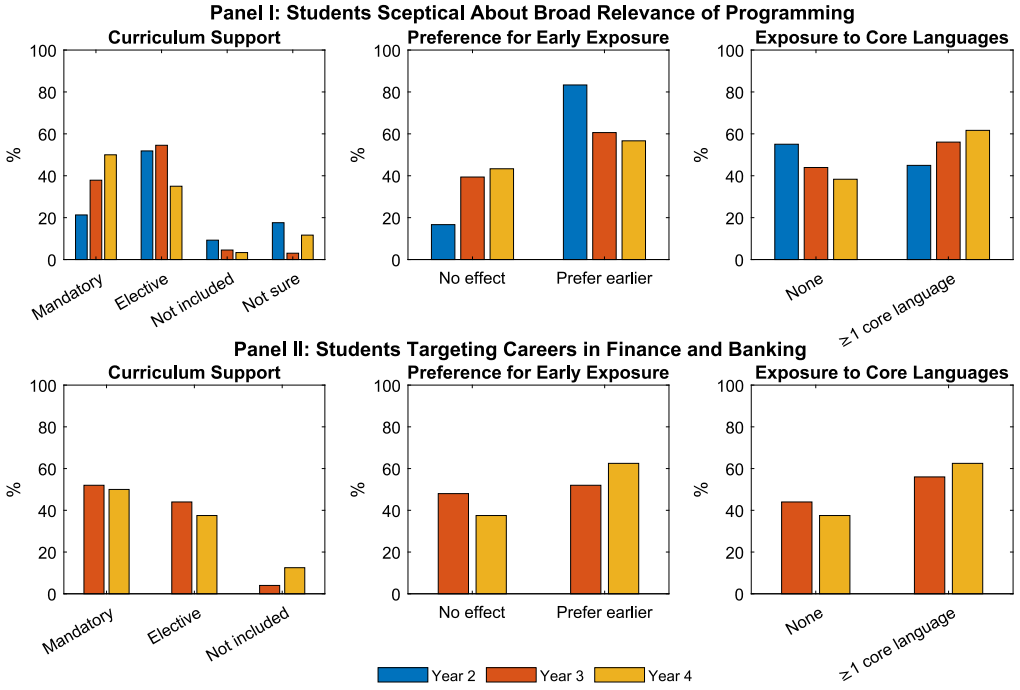

Fig. 5. Subsample evidence for two groups: students who view programming as relevant only in specific fields (top row) and students intending to work in Finance and Banking (bottom row). Within each subgroup, panels report support for curriculum integration (left), preference for earlier exposure (centre), and exposure to at least one core programming language (right), by year group.

Consulting, and Technology and Data Analytics. Examining this group allows us to assess whether students targeting typical graduate roles, often perceived as less technical, though increasingly valuing data literacy and quantitative analysis, show weaker demand for programming skills.

Fig. 5 reports outcomes on three variables later used in the regression analysis: support for curriculum integration (`curriculum`); preference for earlier exposure to programming, based on responses to `skills_early` (`early_timing`); and exposure to core programming languages, derived from the multi-response item `lang` (`lang_any`). The variable `lang_any` equals one if a student reported exposure to at least one core programming language (Python, R, MATLAB, Java, or C++), and zero otherwise. The Finance-and-Banking group shown in the lower panels is identified using the variable `work_path`, which records students' intended employment sector when they plan to start work immediately after graduation.

The top row shows that most sceptical students still express support for embedding programming in Economics, with many favouring making it mandatory. Over half prefer earlier exposure, especially in Year 2, and the majority report experience with at least one core language by Year 4.[7]

The bottom row reveals very similar patterns among students aiming for careers in Finance and Banking. Despite pursuing roles that may not explicitly require programming, these students also show strong support for curriculum integration, a preference for earlier exposure, and substantial programming experience by their final year.

These patterns are nearly identical to those in overall population and reinforce the conclusion that support for programming is broad-based, extending beyond enthusiasts or students in data-intensive fields. Even those with limited initial expectations or more traditional career aspirations often come to support reform and engage with programming when given the opportunity.

3.6. Key messages

Taken together, the evidence on employability beliefs, language exposure, intended applications, and views on provision reveals a consistent set of patterns. First, students overwhelmingly view programming as a core component of economics education rather than an optional or specialist skill. Over 90% support its inclusion in the curriculum, citing direct relevance to employability and postgraduate study.

Second, timing emerges as a central concern. Students, particularly those in final years of the degree, tend to believe that learning programming earlier would have influenced their career choices, while second-year students often indicate that integrating

[7] Results are similar under narrower definitions of scepticism, with nearly identical response patterns across all three questions. Alternative figures using these definitions are reported in Appendix D.

programming now would help inform their future career decisions. This view, coupled with persistently low confidence in programming skills even among final-year students, suggests that introducing programming only at the final years of the degree limits its impact on academic and career trajectories. Students want these skills embedded earlier, when they can still shape degree choices.

Third, students strongly prefer discipline-specific training. They are sceptical of generic computer science modules or incidental programming components, and instead favour instruction grounded in economics. This suggests that instruction should be based on the applied problems, models, and data drawn from the field.

Notably, across these dimensions, international students are consistently ahead. They report earlier exposure, greater confidence, and stronger support for mandatory provision. The responses of the UK students follow similar trajectories, on average with a lag of one year in the degree programme.

These patterns point to a clear policy implication: programming should be introduced early, taught through economics content, and supported in a way that allows students to build confidence well before entering the labour market.

4. Econometric analysis

To complement our descriptive analysis, we now provide econometric evidence that systematically investigates the determinants of student attitudes towards programming skills within the undergraduate economics curriculum. Specifically, we focus on two central outcomes derived from our survey data: support for incorporating programming into the economics curriculum (Section 4.1) and the importance of early exposure to programming (Section 4.2).

Because most survey items are themselves attitudes, we run a minimal specification that uses only exogenous characteristics to see whether support for curriculum reform is simply universal or varies systematically by objective traits. Our aim here is to isolate factors independent of students' subjective attitudes by using variables that are objective or indicative of concrete behavioural choices. While attitudes towards programming are inevitably subjective, we mitigate concerns over purely attitudinal bias by including explanatory variables capturing observable demographics, prior exposure, and concrete educational or career intentions.

In this section we briefly explain key constructed variables used in the analysis, directing interested readers to Appendix C for further details. The dependent variable for the curriculum analysis (`curriculum`) captures students' views on whether programming should be part of the economics curriculum. It is coded as an ordered categorical variable with four values: (1) "mandatory component", (2) "optional elective", (3) "not sure", and (4) "should not be included". Higher numerical values therefore indicate progressively weaker support for integrating programming into the economics curriculum. Given the ordered nature of responses, we use ordered logit regression in Section 4.1.

The dependent variable for the analysis of importance of the early exposure (`early_timing`) captures whether students believe learning programming earlier in their Economics studies would have influenced their academic or career decisions. This binary variable equals 1 for students indicating that earlier exposure would have influenced their decisions (either "yes, definitely" or "somewhat") and 0 otherwise ("no, not really" or "not sure"). Given its binary structure, we employ logistic regression analysis in Section 4.2.

The `early_timing` variable is derived from the `skills_early` item, whose wording differs slightly across cohorts. In Year 2, students were asked whether integrating programming into their Economics studies *now* would help them make a more informed career decision later. In Years 3–4, students were asked whether learning programming earlier in their studies would have influenced their career choices. Despite this wording difference, both versions capture the same underlying idea: whether students view the *timing* of programming exposure within the degree as important for academic or career preparation. We therefore interpret affirmative responses ("yes, definitely" or "somewhat") as evidence that earlier programming exposure is perceived to matter.

Both analyses draw on a common set of explanatory variables, facilitating consistent interpretation and comparison of findings across the two regressions. These explanatory variables include demographic characteristics such as gender (`gender`), year of study (`year_group`), and home versus international student status (`home`). In addition, we incorporate measures capturing students' prior programming exposure (`progrm_outside`), indicating if students learned programming outside the economics department, exposure to programming languages (`lang_any`), and their future academic or career plans (`plans_grp`). Complete definitions, programming details, and further explanation for all these variables are provided in Appendices A and C.

In the following subsection, we present regression results for student attitudes towards curricular integration of programming. All reported marginal effects and significance levels refer to detailed estimates available in Appendix E.

4.1. Support for programming in the curriculum

Table 2 reports results from an ordered logistic regression estimating student support for including programming in the Economics curriculum. The model uses responses from 305 undergraduate economics students.[8] As is typical for attitudinal survey data, the overall model fit is modest (pseudo-$R^2 = 0.067$, likelihood-ratio chi-square = 45.45, $p < 0.001$), but the directional effects and marginal effects remain stable across specifications.[9]

[8] Of the 317 valid survey responses, one observation was excluded because the respondent selected "Prefer not to say" for gender. An additional eleven observations were omitted from the regression analysis due to missing responses on key covariates (curriculum preference, programming exposure, year group, gender, home/international status, outside programming training, or future career plans). This yields an analytical sample of 305 students.

[9] Approximate likelihood-ratio test of proportionality of odds failed to reject the null hypothesis ($\chi^2(5) = 6.56$, $p = 0.26$), confirming that the proportional odds assumption holds.

Table 2
Ordered logit regression for support for programming in the Economics curriculum. Dependent variable: `curriculum`. Sample size: 305.

Variable	Coef.	SE	z	p-value	95% CI	AME
Gender `gender` *(ref: female)*						
Male	−0.026	0.234	−0.11	0.913	[−0.485, 0.434]	0.005
Year Group `year_group` *(ref: Year 2)*						
Year 3	−0.263	0.288	−0.91	0.362	[−0.828, 0.302]	0.056
Year 4	−0.693	0.323	−2.14	0.032	[−1.326, −0.059]	0.153
Home Status `home` *(ref: UK)*						
International	−0.523	0.267	−1.96	0.050	[−1.047, 0.000]	0.113
Exposure to Programming Languages `lang_any` *(ref: No Exposure)*						
≥ 1 Core Language	−0.319	0.249	−1.28	0.199	[−0.806, 0.168]	0.068
Programming Exposure `progrm_outside` *(ref: in ECON)*						
Outside ECON	0.986	0.397	2.48	0.013	[0.207, 1.765]	−0.221
Self-learning	1.036	0.480	2.16	0.031	[0.094, 1.977]	−0.232
No exposure	0.860	0.356	2.41	0.016	[0.161, 1.558]	−0.195
Post-Graduate Plans `plans_grp` *(ref: Work)*						
Master Econ	−0.144	0.305	−0.47	0.637	[−0.742, 0.454]	0.032
Master DA/PhD	−0.779	0.579	−1.34	0.179	[−1.915, 0.356]	0.176
Undecided	0.450	0.272	1.65	0.098	[−0.084, 0.984]	−0.094

Notes: The dependent variable `curriculum` is coded as 1 = mandatory component, 2 = optional elective, 3 = not sure, 4 = should not be included; lower values therefore indicate stronger support for programming in the curriculum. Reported AMEs are average marginal effects on the probability of choosing "mandatory component." Positive AMEs indicate a higher probability of supporting mandatory programming. Statistical significance is reported in the *p*-value column.

Table 2 reports the coefficients from the baseline ordered logit model together with average marginal effects (AMEs) for the probability of choosing "mandatory component", shown in the last column. For readability, we interpret these unadjusted AMEs below. The corresponding inverse-probability-weighted AMEs, as well as additional robustness checks, are reported in Appendix E and are qualitatively very similar.[10]

Students' support for mandatory inclusion of programming is significantly related to their stage in the degree programme. Fourth-year students are about 15 percentage points more likely to advocate mandatory programming compared to second-year students (AME = 0.148, $p = 0.036$). This pattern suggests increased awareness of programming skills value closer to labour market entry. Third-year students show a similar, though statistically insignificant, trend, with a marginal effect of around 6 percentage points.

Home status also plays an important role. International students are about 12 percentage points more likely to support mandatory inclusion compared to the UK students (AME = 0.122, $p = 0.034$). This suggests the greater emphasis international students place on programming skills for competitiveness in global job markets.

Students who acquired programming skills outside the Economics department, whether through other departments, self-study, or external sources, were 19 to 24 percentage points less likely to favour mandatory integration (AMEs from −0.189 to −0.236, all $p < 0.05$). These students may feel less need for structured programming courses, having already addressed the skills gap independently.

Exposure to at least one core programming language (Python, R, MATLAB, Java, C++) was associated with an 8 percentage point lower likelihood of supporting mandatory programming, but this effect was not statistically significant ($p = 0.118$). Thus, the simple fact of exposure, without regard to timing or context, does not appear to shape curricular preferences meaningfully.

Regarding post-graduation intentions, students who were undecided about their future plans were around 10 percentage points less likely to favour mandatory programming compared to those planning to enter the workforce (AME = −0.095, $p = 0.092$, marginal significance). In contrast, students planning further study did not differ significantly from those planning immediate employment.

Gender differences in attitudes towards programming integration were negligible. Male students were no more or less likely to support mandatory inclusion than female students (AME = 0.006, $p = 0.904$), suggesting gender-neutral preferences once other characteristics are controlled for.

Overall, support for embedding programming into the Economics curriculum is strongest among students nearing graduation, international students, and those who have not already acquired programming experience externally. These results suggest that structured, early integration of programming into economics training would effectively serve a broad range of students and reduce reliance on self-directed learning outside the discipline.

As a robustness check, we also estimate a binary logit model in which the dependent variable equals one if the student supports making programming a mandatory part of the Economics curriculum, and zero otherwise. The results, reported in Table G.1, are qualitatively very similar to those from the ordered logit specification.

[10] As a robustness check, all models were also estimated with standard errors clustered by year group and home status. The results remained substantively unchanged. Interaction terms between year group and home status were additionally tested, revealing borderline significance (p-values between 0.055 and 0.059). However, the core findings and marginal effects for other covariates were highly stable. Full results are available upon request.

Table 3
Logistic regression: Factors associated with perceived importance of early exposure to programming. Dependent variable: `early_timing` = 1 if the student reports that earlier programming exposure would have mattered ("yes, definitely" or "somewhat"), and 0 otherwise. Sample size: 305.

Variable	OR	Std. Err.	z	p-value	95% CI	AME
Gender `gender` (ref: Female)						
Male	1.406	0.406	1.18	0.239	[0.798, 2.477]	0.060
Year Group `year_group` (ref: Year 2)						
Year 3	0.274	0.099	−3.57	0.000	[0.134, 0.558]	−0.236
Year 4	0.384	0.149	−2.47	0.014	[0.180, 0.821]	−0.163
Home Status `home` (ref: UK)						
International	2.389	0.821	2.53	0.011	[1.218, 4.686]	0.146
Exposure to Programming Languages `lang_any` (ref: No Exposure)						
≥ 1 Core Language	1.116	0.342	0.36	0.722	[0.611, 2.036]	0.019
Programming Exposure `progrm_outside` (ref: in ECON)						
Outside ECON	1.653	0.768	1.08	0.280	[0.665, 4.109]	0.087
Self-learning	1.037	0.581	0.06	0.948	[0.346, 3.111]	0.007
No exposure	1.325	0.514	0.73	0.468	[0.620, 2.835]	0.051
Postgrad Plans and Certainty `plans_combo` (ref: Work – Certain)						
Work – Uncertain	2.345	1.114	1.79	0.073	[0.924, 5.950]	0.165
Postgrad – Certain	2.800	2.024	1.42	0.154	[0.679, 11.543]	0.194
Postgrad – Uncertain	1.697	0.839	1.07	0.285	[0.644, 4.473]	0.107
Undecided	2.539	1.195	1.98	0.048	[1.009, 6.387]	0.178
Constant						
_cons	1.198	0.785	0.28	0.782	[0.332, 4.328]	–

Notes: Odds ratios from a binary logistic regression estimating associations with students' perceived importance of early exposure to programming in Economics studies. OR > 1 indicates a higher likelihood of perceived importance; OR < 1 indicates a lower likelihood. AMEs report average marginal effects on the probability that `early_timing`=1. For factor variables, AMEs are discrete changes relative to the reference category. Reference categories are noted in parentheses.

4.2. Importance of early exposure to programming

Table 3 presents logistic regression results for students' perceived importance of early exposure to programming in their Economics studies for shaping of their career preparation. The model uses responses from 305 undergraduate economics students, with strong model significance (likelihood-ratio chi-square = 38.63, $p < 0.001$) and typical explanatory power for attitudinal outcomes (pseudo-$R^2 = 0.109$).[11]

The regression results reveal several important insights. First, the year group is strongly associated with the perceived importance of early exposure. Compared to second-year students, third-year students are substantially less likely to believe that earlier exposure would have influenced their career preparation (OR = 0.274, $p = 0.000$), with an average marginal effect of −0.236. Fourth-year students show a similar pattern (OR = 0.384, $p = 0.014$), with an average marginal effect of −0.163. In probability terms, this means that, relative to Year 2 students, students in Years 3 and 4 are about 24 and 16 percentage points less likely, respectively, to report that earlier programming exposure would have mattered. This finding suggests that perceived importance is primarily associated with delayed rather than absent exposure: once students receive some structured instruction, the perceived need for early timing diminishes significantly.

International students are significantly more likely to view early exposure as important for career choice. Relative to UK students, they have approximately 2.4 times the odds of believing that earlier exposure would have influenced their career preparation (OR = 2.389, $p = 0.011$), corresponding to an average marginal effect of 0.146. In other words, international students are about 15 percentage points more likely than UK students to report that earlier programming exposure would have mattered. This aligns with descriptive evidence suggesting that international students are particularly sensitive to programming skill gaps, possibly reflecting their desire to remain competitive in a global labour market. Their greater emphasis on programming likely arises from both direct experience and perceived employers' expectations abroad.

In contrast, measures of prior programming exposure, whether through external courses, self-learning, or exposure to at least one core language, are not significantly associated with the perceived importance of early exposure once other factors are controlled for ($p > 0.2$ for all exposure variables, and the corresponding AMEs are small). This result is consistent with the descriptive evidence and reflects an important distinction made by students between incidental exposure and structured curricular integration. Informal or externally acquired programming skills do not substitute for early, economics-specific instruction, and therefore do not attenuate concerns about delayed exposure. If anything, such experience may heighten awareness of the limitations of non-integrated training, reinforcing rather than weakening the perceived importance of timely, discipline-embedded programming. This interpretation aligns

[11] For the logistic model associated with early exposure, the overall classification accuracy was 75%, with a sensitivity of 95%.

Table 4
Logistic regression: Factors associated with preference for earlier programming exposure (3rd and 4th Year Subsample).
Dependent variable is `early_timing`. Sample size: 161.

Variable	OR	SE	z	p	95% CI	AME
Gender `gender` (ref: Female)						
Male	2.104	0.778	2.01	0.044	[1.019, 4.344]	0.159
Year Group `year_group` (ref: Year 3)						
Year 4	1.399	0.514	0.92	0.360	[0.682, 2.873]	0.070
Home Status `home` (ref: UK)						
International	3.221	1.471	2.56	0.010	[1.316, 7.884]	0.236
Exposure to Programming Languages `lang_any` (ref: No Exposure)						
≥ 1 Core Language	1.612	0.651	1.18	0.237	[0.731, 3.558]	0.101
Programming Exposure `progrm_outside` (ref: In ECON)						
Outside ECON	0.978	0.536	−0.04	0.968	[0.334, 2.865]	−0.005
Self-learning	1.194	0.889	0.24	0.812	[0.277, 5.141]	0.038
No exposure	1.782	0.773	1.33	0.183	[0.761, 4.171]	0.119
Postgrad Plans and Certainty `plans_combo` (ref: Work – Certain)						
Work – Uncertain	3.215	1.834	2.05	0.041	[1.051, 9.834]	0.249
Postgrad – Certain	2.031	1.794	0.80	0.422	[0.360, 11.472]	0.155
Postgrad – Uncertain	1.544	0.915	0.73	0.463	[0.484, 4.930]	0.096
Undecided	3.221	1.835	2.05	0.040	[1.054, 9.840]	0.250
Academic Performance `academic_simpler` (ref: Top marks)						
Other	0.391	0.182	−2.02	0.044	[0.157, 0.974]	−0.184
Constant						
_cons	0.341	0.251	−1.46	0.143	[0.081, 1.440]	–

Notes: Odds ratios from a binary logistic regression estimating associations with students' preference for earlier programming exposure in Economics studies. OR > 1 indicates a higher likelihood of preferring earlier exposure; OR < 1 implies lower likelihood. AMEs report average marginal effects on the probability that `early_timing`=1. For factor variables, AMEs are discrete changes relative to the reference category. Reference categories are noted in parentheses.

with the descriptive evidence showing that students value programming most when it is directly connected to economic data analysis and modelling tasks, rather than treated as a generic technical skill.

Career certainty also plays a notable role. Students who are uncertain about their future employment (“Work—Uncertain”) or undecided about their career path (“Undecided”) have substantially higher odds of viewing early exposure to programming as a determinant of career choice (OR = 2.345, $p = 0.073$ and OR = 2.539, $p = 0.048$, respectively) compared to students with certain work plans. The corresponding AMEs are 0.165 and 0.178, implying increases of about 17 and 18 percentage points in the probability of reporting that earlier exposure would have mattered. This suggests that for students navigating uncertain or competitive job markets, programming skills are viewed as a valuable source of flexibility and employability in economics-related fields. One plausible interpretation is that such students are effectively hedging: when future pathways are less clearly defined, broader technical exposure becomes more attractive because it preserves access to a wider range of academic and career options.

Finally, gender differences in the perceived importance of early exposure are small and statistically insignificant (OR = 1.406, p = 0.239, AME = 0.060), consistent with the findings on curriculum support. Thus, the perception appears broadly similar between male and female students.

Taken together, these results reinforce the importance of early, discipline-specific programming instruction within Economics curricula. Students' perceptions are shaped far more by the timing and structure of exposure than by the mere presence of technical skills. Structured programming courses introduced early in undergraduate education could significantly reduce the perceived need for early exposure and better support academic and career preparedness in economics.

To further explore the equity implications of these findings, we examine whether preferences for earlier exposure vary by academic performance. We restrict this additional exercise to the timing-preference model because academic-performance information is available only for third- and fourth-year students, and in our data it was not informative for support for curriculum integration. We therefore treat this specification as a targeted robustness check on heterogeneity in timing preferences rather than as a parallel extension of all baseline models. We re-estimate the timing-preference model for a subsample of 161 third- and fourth-year students who reported their marks, introducing a simplified indicator of academic performance that distinguishes students with top results (first-class average) from all others. The results are reported in Table 4.

The estimates show that academically stronger students are significantly more likely to prefer earlier exposure to programming, even after controlling for year group, home status, career plans, and prior programming exposure. Students outside the top-performing group have markedly lower odds of reporting that earlier exposure would have mattered (OR = 0.391, $p = 0.044$), corresponding to an average marginal effect of −0.184. In probability terms, this implies that they are about 18 percentage points less likely than top-performing students to express a preference for earlier exposure. This pattern suggests that high-performing students may be particularly aware of missed opportunities for skill development or better able to recognise the long-term value of early programming instruction for advanced study and economics-related careers. Importantly, this does not imply indifference

among other students: as shown in both the descriptive evidence and the main regressions, support for programming integration is widespread across the cohort. Rather, the difference reflects a stronger belief among top-performing students that earlier exposure would have materially influenced their preparation. Taken together, these patterns are difficult to reconcile with an interpretation based purely on perceived difficulty. Preferences for earlier exposure are strongest among academically high-performing students and persist even among those with prior programming experience. This suggests that timing preferences reflect forward-looking considerations about skill accumulation and career preparation, rather than avoidance of technical challenge.

In this performance-conditioned subsample, gender also becomes statistically significant, with male students expressing a greater preference for earlier exposure once academic achievement is accounted for (OR = 2.104, $p = 0.044$; AME = 0.159). This implies an increase of about 16 percentage points in the probability of preferring earlier exposure. This result suggests that gender differences may interact with academic confidence or progression rather than reflecting broad-based differences in attitudes towards programming.

The other strongest marginal effects in this subsample are associated with international status (AME = 0.236) and uncertain or undecided career plans (AMEs of about 0.25), reinforcing the view that preferences for earlier exposure are especially strong among students facing greater uncertainty about future pathways.

Finally, to assess the robustness of these results to minor sample imbalances by year group, gender, and home status, we re-estimated the main timing-preference model using inverse probability weighting. The weighted estimates, reported in Appendix F (Table F.1), are highly consistent with the baseline results, preserving both the magnitude and significance of the key coefficients. This confirms that the central findings are not driven by differential response patterns across student groups.

5. Implications for curriculum co-design

The empirical findings in this paper provide direct student-side support for the staged curriculum framework proposed in Hashimzade et al. (2025). While that study develops a faculty- and institution-led case for introducing programming as a core component of undergraduate Economics degrees, the evidence presented here demonstrates that students' preferences over timing, sequencing, and accessibility are closely aligned with such a scaffolded approach.

More broadly, the findings point to clear priorities for curriculum reform in undergraduate Economics education, particularly with respect to the timing of programming exposure and the distributional consequences of delaying technical skill acquisition.

Descriptive evidence from Section 3 reveals near-universal endorsement of programming integration: almost all students, across year groups, genders, and home statuses, support including programming either as a required or elective component. Very few believe that programming is unnecessary. This widespread consensus supports a strong mandate for reform.

Regression analysis in Section 4 sharpens this picture. Two key implications for Economics curriculum co-design emerge from our findings.

First, the timing of programming exposure is critical, providing direct support for a staged introduction of programming within the Economics curriculum. Students who encounter programming only in the later stages of their Economics degree are significantly more likely to report that earlier exposure would have influenced their career preparation, as evidenced by stronger preferences among second-year students compared to third- and fourth-year students. Delaying programming exposure until the final years of the degree heightens demand for earlier exposure, particularly among students with limited prior experience, and is associated with low confidence in programming skills even among final-year students. Departments aiming to enhance both student satisfaction and employability outcomes should therefore consider introducing programming instruction early in undergraduate Economics curricula.

Second, targeted curriculum reform could yield particular benefits for specific student groups. International students and those uncertain about their career trajectories express the strongest preference for earlier programming exposure in their Economics studies and express the greatest support for curricular change. These groups are likely to be disadvantaged at the labour market for Economics graduates: the international students because of their weaker knowledge of local conditions and opportunities and lack of local networks, and the undecided students because of their lack of focussed preparation for particular jobs or careers in a specific sector. Equity considerations strengthen this argument further. Students from underrepresented or lower-SES backgrounds are less likely to have had access to computing at school or to the resources needed to self-teach coding, and they may be unable to afford postgraduate study where programming is commonly introduced. Integrating programming early in undergraduate economics therefore helps to level the playing field, ensuring that essential technical skills are developed within the degree rather than left to those with prior advantage or additional means. Such early exposure can also improve students' ability to make informed career choices and to prepare for the demands of the modern labour market for Economics graduates.

Taken together, the evidence suggests that curriculum reform would not be a divisive intervention, but rather a response to broad-based and well-articulated student preferences. Early and structured exposure to programming, embedded within economics-focused content, would likely be widely welcomed. Tailored supports, such as scaffolding learning, economic model-based applications, and optional supplementary sessions, could further expand accessibility, particularly for students with low initial confidence.

Thus, this study reinforces the value of curriculum co-design. By incorporating student perspectives directly into the reform process, institutions can better align programming instruction with learner needs, reduce mismatches between curricular content and employer expectations, and promote equitable access to technical skills that increasingly define the economics profession.

Consistent with this evidence, a practical and policy-relevant response is the staged approach outlined in Hashimzade et al. (2025). The proposed two-course framework, comprised of-*Programming Principles for Economists* followed by *Applied Programming for Economists*, introduces programming logic and problem-solving early in the degree and extends these skills to econometric and macroeconomic applications at the honours level. As demonstrated in that paper, implementation costs are modest: the courses can

be fitted within existing teaching allocations and use widely available software and infrastructure. This structure is therefore both pedagogically coherent and financially feasible, providing a clear route from foundational programming literacy to discipline-specific application.

Adding new or adapting the existing material will, of course, require certain efficient adjustments elsewhere. As discussed in Hashimzade et al. (2025), implementation would require departments to streamline or reallocate existing provision, for example by reducing duplication across methods teaching or embedding computational work into topics currently taught as pure theory or using qualitative description.

A further implication concerns generative AI. Recent developments in AI may reduce the value of memorising syntax or writing routine code from scratch, but they do not eliminate the need for foundational programming knowledge. If anything, they increase the importance of understanding data structures, model logic, debugging, interpretation, and the economic meaning of computational output. Students who lack such foundations are less able to evaluate, adapt, or safely use AI-generated code in applied economics settings. The case for early programming instruction is therefore not weakened by AI: rather, the emphasis shifts towards coding as analytical understanding and disciplined problem-solving, not merely code production.

6. Conclusion

This study provides systematic student-side evidence on how programming should be integrated into undergraduate Economics curricula, moving beyond employer surveys and conceptual curriculum proposals. By analysing students' preferences over timing, confidence formation, and accessibility, the findings show that early, discipline-specific programming instruction is not only widely supported, but also critical for addressing persistent confidence gaps and unequal access to technical skills. These patterns are particularly pronounced among international students, academically strong students, and those with initially uncertain career plans, underscoring the importance of sequencing decisions for both pedagogical effectiveness and equity. While we do not observe graduate outcomes directly, students' perceptions of employability and timing align closely with documented employer demand for programming skills, suggesting that student attitudes provide a valuable complementary perspective to employer-based evidence.

While the analysis focuses on a single institution, the consistency and robustness of the results across cohorts suggest that the conclusions may be relevant beyond the local context. Extending this approach to multi-institutional settings, particularly in three-year degree programmes and in the U.S. context, and linking students' exposure and timing of programming instruction to longitudinal graduate outcomes would provide valuable comparative evidence and strengthen the external validity of these findings.

More broadly, the study highlights the value of bringing student perspectives into the curriculum design process. Co-design not only helps align teaching with learner needs but also supports wider engagement with curriculum reform across diverse institutional contexts. Together with the curricular framework developed in Hashimzade et al. (2025), which reviewed existing practice and implementation costs, this paper provides complementary evidence from the student perspective to guide the timing and structure of programming integration. As programming becomes increasingly central to economics education and practice, involving students in shaping how these skills are taught will be essential for both pedagogical effectiveness and inclusive curriculum development.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.iree.2026.100347.

Data availability

Data will be made available on request.

References

AEA, 2024. Committee on Economic Education Reports. American Economic Association, Available at https://pubs.aeaweb.org/doi/pdfplus/10.1257/pandp.114.811. (Accessed 9 April 2025).

Bamforth, S.E., Robinson, C.L., Croft, T., Crawford, A., 2007. Retention and progression of engineering students with diverse mathematical backgrounds. Teach. Math. Appl.: Int. J. IMA 26 (4), 156–166.

Blumenstyk, G., 2016. Liberal-arts majors have plenty of job prospects, if they have some specific skills, too. The Chronicle of Higher Education, June 9. https://www.chronicle.com/article/liberal-arts-majors-have-plenty-of-job-prospects-ifthey-have-some-specific-skills-too/. (Accessed 20 August 2024).

Bovill, C., Bulley, C.J., 2011. A model of active student participation in curriculum design: Exploring desirability and possibility. In: Rust, C. (Ed.), Improving Student Learning (ISL) 18: Global Theories and Local Practices: Institutional, Disciplinary and Cultural Variations. Oxford: Oxford Brookes University, pp. 176–188.

CBI, 2022. Education and Skills Survey 2022. Confederation of British Industry, Available at https://www.cbi.org.uk/articles/skills-creating-the-conditions-for-investment-cbi-education-and-skills-survey-2022/. (Accessed 9 April 2025).

CORE Project, 2024. Quantitative methods in economics education. Available at https://www.core-econ.org. (Accessed 9 April 2025).

Economics Network, 2019. 2019 employers survey. Available at https://www.economicsnetwork.ac.uk/projects/surveys/employers2019. (Accessed 9 April 2025).

Gorry, D., Gilbert, J., 2015. Numerical simulations of competition in quantities. Int. Rev. Econ. Educ. 18, 49–61. http://dx.doi.org/10.1016/j.iree.2015.01.003.

Hashimzade, N., Kirsanov, O., Kirsanova, T., 2025. Integrating programming into the modern undergraduate economics curriculum. Int. Rev. Econ. Educ. 49, 100310. http://dx.doi.org/10.1016/j.iree.2025.100310.

Healey, M., Flint, A., Harrington, K., 2014. Engagement through Partnership: Students as Partners in Learning and Teaching in Higher Education. Higher Education Academy, York.

IFS, 2024. Graduate Outcomes Series. Institute for Fiscal Studies, Available at https://ifs.org.uk/series/graduate-outcomes. (Accessed 9 April 2025).

Jenkins, B.C., 2022. A Python-based undergraduate course in computational macroeconomics. J. Econ. Educ. 53 (2), 126–140. http://dx.doi.org/10.1080/00220485.2022.2038322.

Kuroki, M., 2021. Using Python and Google Colab to teach undergraduate microeconomic theory. Int. Rev. Econ. Educ. 38, 100225. http://dx.doi.org/10.1016/j.iree.2021.100225.

Kuroki, M., 2023. Integrating data science into an econometrics course with a kaggle competition. J. Econ. Educ. 54 (4), 364–378. http://dx.doi.org/10.1080/00220485.2023.2220695.

Kuroki, M., 2025. Integrating advertising into duopoly simulations for intermediate microeconomics courses using python. J. Econ. Teach. 10 (4), 308–320. http://dx.doi.org/10.58311/jeconteach/73359d691c4b63156842af299030f00ad61ab5af.

Luedtke, A.O., 2023. Teaching nash equilibrium with python. J. Econ. Educ. 54 (2), 177–183. http://dx.doi.org/10.1080/00220485.2023.2168813.

NACE, 2024. What are Employers Looking for When Reviewing College Studentsâ€™ Resumes? National Association of Colleges and Employers, Available at: https://www.naceweb.org/talent-acquisition/candidate-selection/what-are-employers-looking-for-when-reviewing-college-students-resumes/. (Accessed 9 April 2025).

Parsons, S., Croft, T., Harrison, M., 2009. Does studentsâ€™ confidence in their ability in mathematics matter? Teach. Math. Appl.: Int. J. IMA 28 (2), 53–68.

Solis-Garcia, M., 2021. Yes we can! Teaching DSGE models to undergraduate students. J. Econ. Educ. 49 (3), 226–236. http://dx.doi.org/10.1080/00220485.2018.1464987.