The new computing curriculum in English schools: a statistical analysis of student participation

Peter E J Kemp, University of Roehampton; Miles G Berry, University of Roehampton; Billy Wong, University of Reading
peter.kemp@roehampton.ac.uk; m.berry@roehampton.ac.uk; b.wong@reading.ac.uk

The framework for English education at school level is largely determined by three inter-related but independent components: the National Curriculum, which describes what should be taught in local authority maintained schools between ages 5 and 16 and is published by the government's Department for Education; the specifications of GCSE (16+) and A-level (18+) qualifications, regulated by Ofqual (the Office for Qualifications), and the school inspection regime conducted by Ofsted (the Office for Standards in Education). Ofqual and Ofsted are non-ministerial government departments. UK devolved administrations for Scotland, Wales and Northern Ireland have their own curricula, qualifications and inspection arrangements.

In the data presented here, the focus is on the uptake of and achievement in the GCSE and A-level qualifications in the 2016 summer exams, two years after computing was introduced as a national curriculum subject, but before the withdrawal of ICT as GCSE and A Level qualifications. There are disparities between regions and local authorities, and between genders, ethnicity and socio-economic status.

Key Dates

2010 A new coalition government is elected. Their first acts in education include the abolition of the Qualifications and Curriculum Development Authority and the British Educational Communication and Technology Agency. They reject the previous government's proposals for a revised primary National Curriculum and announce a review of the whole National Curriculum, to be chaired by Tim Oates of Cambridge Assessment.

The OCR exam board announce the pilot of a new GCSE qualification in computing, to sit alongside the existing qualifications in ICT (information and communication technology), which had become devalued by many students, parents and employers due to a focus on relatively mundane, user-level skills.

2012 After persuasive advocacy from The Royal Society, the video games and visual effects industries, Google and the grassroots subject association Computing At School, and concerns expressed by Ofsted over the quality of the ICT national curriculum and qualifications, Education Secretary Michael Gove announced that the ICT curriculum would be withdrawn and that computer science would be taught as a subject in all schools. His stated rationale was that it was advised that a panel, chaired by the British Computer Society and the Royal Academy of Engineering would be convened to provide expert advice on the content of a new ICT curriculum, which would include programming and other aspects of computer science.

2013 ICT is replaced by Computing in the National Curriculum. The subject emphasises computational thinking and creativity, with elements of computer science, information technology and digital literacy (foundations, applications and implications) for all students from 5 to 16 in local authority supported schools.

The Department for Education accepts a BCS recommendation that GCSE Computing be counted as a science for the English Baccalaureate (EBacc) school performance measure.

2014 The new national curriculum takes effect. The programmes of study for computing, as with other subjects, apply to all years simultaneously, allowing no gradual phasing in of the new requirements.

Alongside reform of other A-level qualifications, new subject content requirements are produced for A-level computer science. Three of the four exam boards subsequently develop specifications in line with these requirements. Of note are an open-ended coursework project, and a requirement to know about a variety of programming paradigms.

2015 New requirements for GCSE computer science are announced by the Department for Education, and all four awarding organisations subsequently develop specifications subsequently approved by Ofqual. These increase the academic rigour of the subject and place more emphasis on theoretical computer science. The specifications include 20% credit for a non-exam assessment (NEA) of practical programming, requiring 20 hours of independent work under closely monitored conditions.

Subsequently, the department for Education announces that GCSE and A-level ICT will not be renewed and will thus cease to be available, citing too much overlap with the new computer science qualifications.

2017 After reports of widespread malpractice in GCSE computer science NEAs, Ofqual announce their intention to remove this component from the assessed work for GCSE computer science; thus pupils will still be required to undertake the NEA but this work will no longer count toward their exam grade.

2018 Students will sit GCSE and A-level exams in ICT for the last time. The evidence we present here indicates that as a result there will be fewer female, BAME, socio-economically disadvantaged and lower ability students taking qualifications in the computing space, thus the subject area becomes less inclusive and less diverse overall.

The content for computer science GCSEs

Introduction

1 The GCSE subject content sets out the knowledge, understanding and skills underpinning the subject to be taught and learnt. It also sets out assessment objectives it provides the framework within which the awarding organisations create the detail of their specifications, so ensuring progression from key stage 3 national curriculum requirements and the possibilities for development into A level.

Subject aims and learning outcomes

2 All specifications in computer science must build on the knowledge, understanding and skills established through the computer science elements of the national curriculum at key stage 3 and must enable students to develop an interest in elements of computer science at key stage 4 and enable students to progress into further learning and/or employment.

3 GCSE specifications in computer science should enable students to:

- understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation
- analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs
- think creatively, innovatively, analytically, logically and critically
- understand the components that make up digital systems, and how they communicate with one another and with other systems
- understand the impacts of digital technology to the individual and to wider society
- apply mathematical skills relevant to computer science

Subject content

Knowledge and understanding

4 GCSE specifications must require students to develop a knowledge and understanding of the fundamentals of computer science and programming including:

- standard algorithms, including binary search and merge sort
- following and writing algorithms to solve problems including
  - sequence, selection and iteration
  - input, processing and output
- how particular programs and algorithms work
- the concept of data type, including integer, Boolean, real, character and string, and the ways in which computers store and process different types of data
- representation of numbers in binary and hexadecimal, conversion between those and decimal, binary addition and shifts
- representation of text, sound, and graphics inside computers
- Boolean logic using AND, OR and NOT, combinations of these, and the ability to construct and simplify appropriate truth tables to solve problems
- the purpose and functionality of systems software, including the operating system and utility software
- characteristics of systems architecture, including
  - CPU architecture, including Von Neumann and the role of the components of the CPU in the fetch-execute cycle
  - main and contemporary secondary storage and ways of storing data on these devices, including their advantages and disadvantages
  - data capacity and calculation of data capacity requirements
  - hardware components and embedded systems
- networks and the importance of
  - connectivity, both wired and wireless
  - types of network
  - common network topologies
  - network security
  - the concept of networking protocols, including Ethernet, Wi-Fi, TCP/IP, HTTP, HTTS, FTP and email protocols
  - layers
- cyber security (forms of attack (based on technical weaknesses and behaviour), methods of identifying vulnerabilities, and ways to protect software systems (during design, coding, testing, and use)
  - the ethical, legal and environmental impacts of digital technology on wider society, including issues of privacy and cyber security
- characteristics and purpose of different levels of programming language, including low-level language

Skills

5 GCSE specifications must require students to develop the following skills:

- take a systematic approach to problem solving including the use of decomposition and abstraction, and make use of conventions including pseudocode and flowcharts
- design, write, test and refine programs, using one or more high-level programming languages with a textual program definition, either to a specification or of their own design
- use appropriate security techniques, including validation and authentication
- evaluate the fitness for purpose of algorithms in meeting requirements efficiently using logical reasoning and test data
- use abstraction effectively
  - to model selected aspects of the external world in a program
  - to appropriately structure programs into modular parts with clear, well-documented interfaces
- apply computing-related mathematics
  - connecting, both wired and wireless
  - types of network
  - common network topologies
  - network security
  - the concept of networking protocols, including Ethernet, Wi-Fi, TCP/IP, HTTP, HTTS, FTP and email protocols
  - layers

* Formal compression of algorithms efficiency also not required

GCSE

Public examination at the end of a 2 year course, typically studied by 14 to 16 year olds.

Problem: Cross and Nots

Cross and Nots is a growing game where the player tries to guess a number that has been randomly generated by the computer. The player's guesses are then compared against the generated number for duplicates. An example of a valid randomly generated number is 1234. An example of an invalid randomly generated number is 1123 because it contains two 1s.

The player makes a number to try to beat the computer. The computer then checks the player's guess against the secret number. Each individual digit entered by the player is compared to each digit within the randomly generated number. If a digit in the player's guess matches a digit in the randomly generated number, a cross is made in the player's number that lines up in that "cell". The digit is in the randomly generated number but in a different position than the digit in cell.

If a digit in the player's guess matches a digit in the randomly generated number, it is not in the randomly generated number. In record a tally of the number of guesses it took to guess the correct number.

The program must work in the following way:

1. To welcome generated the digit number is chosen. The number must have exactly four digits between 0 and 9. All four of the digits must be different to each other.
2. The program should then ask the player to enter a number to try to beat the computer.
   - If the player entered said that the randomly generated four digit number is shown and the program ends.
   - If the player entered a number with fewer than four digits then the user is asked whether intended of exactly four digits. An appropriate message must be displayed if the user enters a number with more than four digits.
   - If the four digit number entered contains repeated digits then an appropriate message should be displayed.
   - Step 2 should be repeated until a valid input is received.
3. The program should check the player's number against the randomly generated number
   - If a digit in the player's number matches a digit in the randomly generated number and is in the same position then cow but is counted.
   - If a digit in the player's number matches a digit in the randomly generated number but is in a different position then one cow is counted.
   - The number of cows and bulls should be displayed to the number of bulls and cows.

Rules 2 and 3 are repeated until the randomly generated four digit number has been correctly guessed by the player. If the guess is successful an appropriate "success" message is displayed. A count of the number of guesses the player has taken should be displayed to the player once the game is complete.

The algorithm in **Figure 4** is the binary search algorithm designed to search for a value (item) an array.

Figure 4

```
• Line numbers are included but are not part of the algorithm.
• Line 30 contains a deliberate error (bug) of 1.
  10 low = 0
  20 high = N - 1
  30 mid = (low + high) DIV 2
  40 WHILE (low <= high) AND (mid <> item)
  50   IF mid < item THEN
  60     low = mid + 1
  70   ELSE
  80     high = mid - 1
  90   mid = (low + high) DIV 2
  100 ENDWHILE
  110 IF mid = item THEN
  120   PRINT "Found"
  130 ELSE
  140   PRINT "Not found"
  150 ENDIF
```

Complete the trace table for the algorithm in **Figure 4** (you may not need to use all of the lines in the table). The first value of mid is entered for you.

(3 marks)

mid	left	right	mid	mid (result)

Why would the binary search algorithm shown in **Figure 4** not work when the array contains 1000 elements? (1 mark)

(1 mark)

Reproduced under Fair Use

1 Selecting and approaching a project

1.1 Selecting a project

The project will take a significant amount of time to complete and contributes 30% to the overall grade. It is important that the student chooses a suitable project topic for the purpose of:

- an in-depth final report and coding skills; and
- is suitably challenging so that it will enable them to fulfil their learning potential
- is feasible to complete in the time allowed
- can be supported and assessed by the teacher.

It is the student who must decide upon the project topic, but it is expected that the teacher will guide the student to a suitable project.

Suitable for a solution:

The student selects a problem and develops a solution to solve it. Typically, the solution will involve the student writing a program. The problem must be suitable for a solution to be found from having once or twice to be useful. Examples of the type of project include:

- a crossword, sign of a crossword, or scientific nature, or a word-search problem such as that found in a newspaper.
  https://en.wikipedia.org/wiki/Crossword#Dry_Valley_Cryptic
- a game such as a card game, board game, or a game involving a physical representation.
  - the ability to develop problems in computational terms through practical experience of solving such problems, including writing programs to do so
  - the capacity for thinking creatively, innovatively, analytically, logically and critically
  - the capacity to see relationships between different aspects of computer science
  - mathematical skills as set out in the attainment target
  - the ability to articulate the individual (moral), social (ethical), legal and cultural opportunities and risks of digital technology

Subject content

Knowledge and understanding

- KS and A level qualifications must require students to develop a knowledge and understanding of the fundamentals of computer science and programming including:
  - fundamental of programming
  - the concept of data type, including primitive data types and complex data structures
  - data representation
  - following and writing algorithms
  - methods of planning, refining, exchanging and managing data to produce information in a political purpose
  - the tools for and functions of systems software
  - characteristics of contemporary systems architecture, including processors, storage, memory and communication
  - characteristics of networks and the importance of networking protocols and standards
  - the individual (moral), social (ethical), legal and cultural opportunities and risks of digital technology
- In addition, A level specifications must require students to develop a knowledge and understanding of:
  - the Turing machine
  - the Church-Turing thesis
  - the concept of computational complexity, including the concepts of algorithmic efficiency, the same task can be completed
  - standard algorithms
  - the use of databases to store, retrieve and manipulate data, including relational databases and the concept of normalisation
  - the need for and characteristics of a variety of programming paradigms

Skills

- KS and A level specifications must require students to develop the following skills:
- take a systematic approach to problem solving
- design, write and test programs to either a specification or to solve a problem
- articulate how a program works, arguing for its correctness and efficiency using logical reasoning, test data, and user feedback
- use abstraction effectively
  - to model selected aspects of the external world in a program
  - apply computing-related mathematics
- In addition, A level specifications must require students to
  - know and understand how to write specifications for a programming solution

Annex: mathematical skills

Computer science uses mathematics in exercise for computational laws and processes.

Any accredited specification in computer science must contain a minimum of 10% mathematics appropriate to the level of the specification. Awarding organisations should ensure that the mathematics may be seen to demonstrate their knowledge and understanding and use of computational processes and problem solving in both theoretical and practical ways. The following is a list of the mathematical skills students should be expected to demonstrate:

For each topic below, while the concepts are used in (through out) appropriate GCSE mathematics specifications, students will, however, be expected to apply the skills they acquire in a wider 2 context.

Topics

- Boolean algebra
- Comparison of complexity of algorithms*
- Number representations and bases

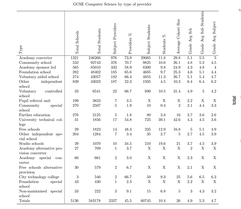
*applicable to A level only, due to its advanced nature.

Published: April 2014

© Crown copyright 2014
Reference: DFE-00508-2014

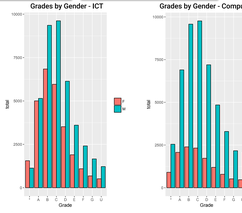
A Level

Public examination at the end of a 2 year course, typically studied by 16 to 18 year olds.



GCSE Computer Science by type of provider

Type	Total Schools	Total Students	Subject Provision	Provision %	Subject Students	Students %	Average Cohort Size	Grade Avg Sch.	Grade Avg Sch Standard	Grade Avg Subject	Grade Avg Subject Standard
Academy converter	1321	240268	976	73.9	39065	11.6	39.9	5.1	5.1	5	5
Academy sponsor led	365	67111	270	74.0	9	0.0	33.3	4.5	4.6	4.5	4.6
Academy sponsor led	365	93010	323	88.5	4089	9.0	25.9	4.5	4.6	5.1	5
Foundation school	292	49402	185	63.4	4045	9.7	25.3	4.6	5.1	4.4	4.4
Foundation school	292	51000	207	70.9	3553	8.5	21.8	4.6	5.1	4.4	4.4
Other independent school	439	63023	147	33.5	1051	1.5	16.5	6.4	6.4	6.3	6.2
Community controlled school	55	6541	22	66.7	690	10.5	31.4	4.9	5	4.2	
Pupil referral unit	190	2623	7	3.5	X	X	X	3.2	X	X	
Community special school	270	2507	5	1.9	10	0.4	2	3.1	4.4	3.0	
Further education	376	2135	5	1.4	40	0.4	16	3.7	3.6	3.6	
University technical college	31	1956	17	54.8	73	38.1	4.3	4.1	3.8	3.6	
Free schools	29	1423	14	48.3	235	13.9	16.8	5	5.1	4.9	
Other independent special school	204	1394	7	3.4	15	1.1	2.1	3.7	4.5	3.9	
Studio schools	29	1070	10	34.5	233	19.6	23	4.7	4.5	3.9	
Academy alternative provision converter	17	799	1	3.7	X	X	X	X	2	X	X
Community special centre	66	661	2	3.0	X	X	X	3.3	X	X	
Free schools alternative provision	30	579	2	6.7	X	X	X	2.1	X	X	
City technology college	3	549	2	66.7	50	9.2	25	5.6	6.3	6.3	
Foundation special school	43	430	1	2.3	X	X	X	3.2	X	X	
Non-maintained special school	10	222	3	9.1	15	6.8	5	3	4.3	3.3	
Total	3130	543176	2337	45.5	60715	10.4	26	4.9	5.3	4.7	



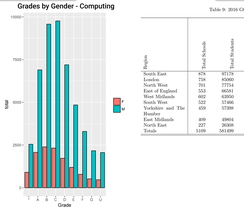
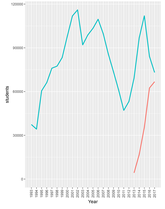


Table 9: 2016 GCSE Computer Science by region

Region	Total Schools	Total Students	Subject Provision	Provision %	Subject Students	Students %	Average Cohort Size	Grade Avg Sch.	Grade Avg Sch Standard	Grade Avg Subject	Grade Avg Subject Standard
South East	678	97176	395	43.0	10299	10.6	26.1	5.1	5.6	4.8	
London	518	93029	342	37.2	8200	8.8	24.0	5.1	5.5	4.8	
North West	791	77754	334	47.4	4032	5.1	25.6	4.9	5.2	4.6	
East of England	553	66501	271	49.7	7420	11.1	27	5	5.2	4.7	
West Midlands	602	63950	256	42.5	6411	10	25	4.9	5.1	4.8	
South West	522	57466	246	47.1	6398	11.1	26	5	5.2	4.6	
Yorkshire and The Humber	459	57399	204	44.4	5053	8.8	24.8	4.8	5.3	4.7	
East Midlands	409	49604	198	48.4	4937	9.9	26.3	4.8	5.1	4.4	
North East	227	26388	105	46.3	2664	10.1	25.4	4.8	5.3	4.8	
Totals	5169	541499	2329	45.6	60614	10.4	26	4.9	5.3	4.7	



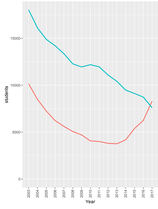
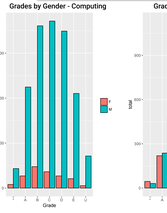
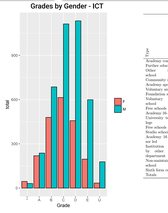
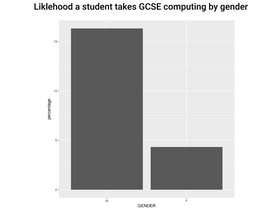


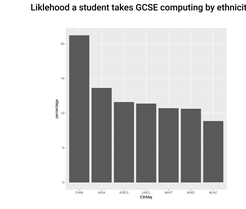
Table 10: 2016 Alevel Computer Science by region

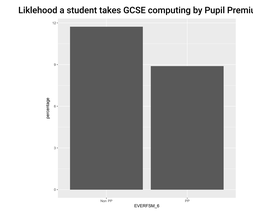
Region	Total Schools	Total Students	Subject Provision	Provision %	Subject Students	Students %	Average Cohort Size	Grade Avg Sch.	Grade Avg Sch Standard	Grade Avg Subject	Grade Avg Subject Standard
South East	543	32330	158	38.1	1341	8.9	8.5	203.4	213.8	207.7	217
London	518	45029	142	27.4	820	1.8	5.8	223.5	213.7	209.4	
North West	368	33605	87	24.2	660	2.1	9	221	213.4	212.2	
East of England	342	31518	112	32.7	661	2.1	5.9	222.4	211.9	205.3	
West Midlands	327	29604	102	31.2	545	2	5.3	220.9	211.6	207.2	
South West	297	26026	84	24.9	383	1.5	4.7	219	208.9	206.3	
Yorkshire and The Humber	246	22920	44	17.9	266	1.6	6.7	214.5	208	199	
East Midlands	248	20366	77	31.0	430	2.1	5.7	218.4	209.6	205	
North East	167	9766	29	27.1	192	2	6.6	216.2	208.9	203.1	
Totals	2923	267228	834	28.5	5341	2	6.4	222	213.6	207.2	

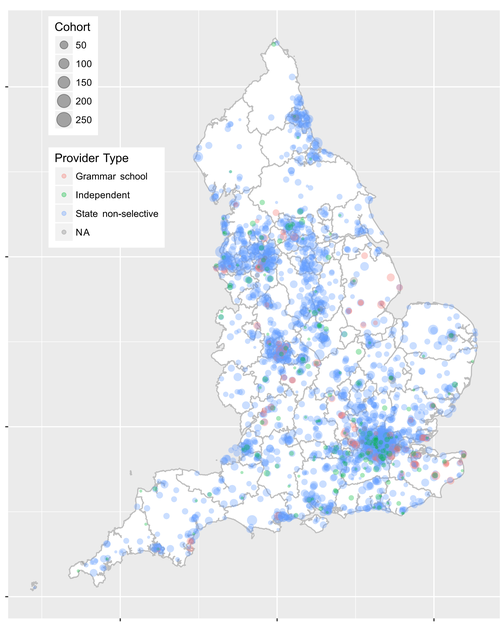


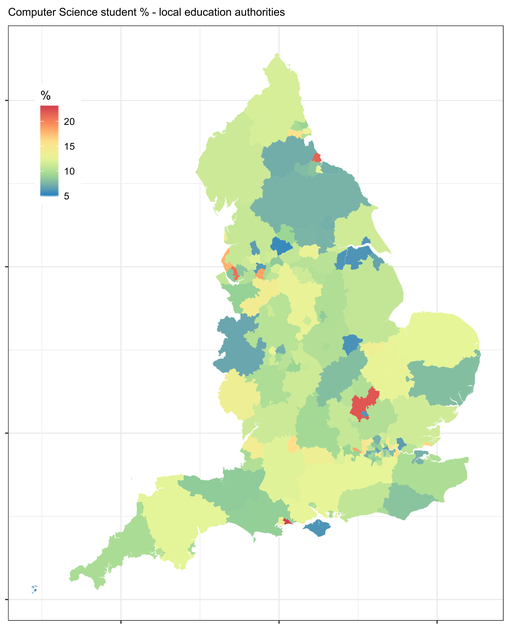


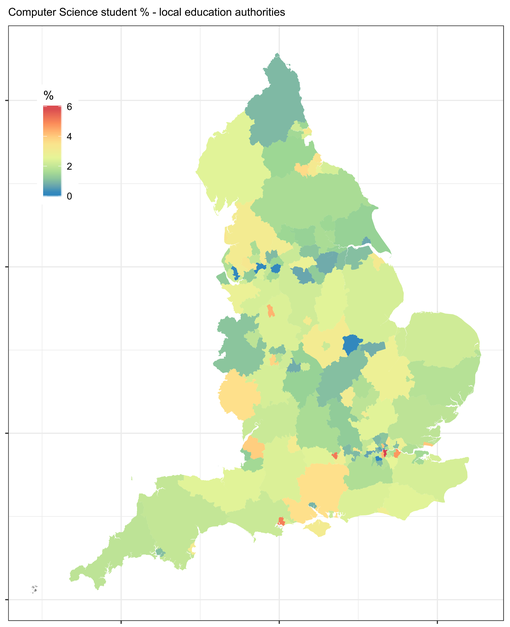


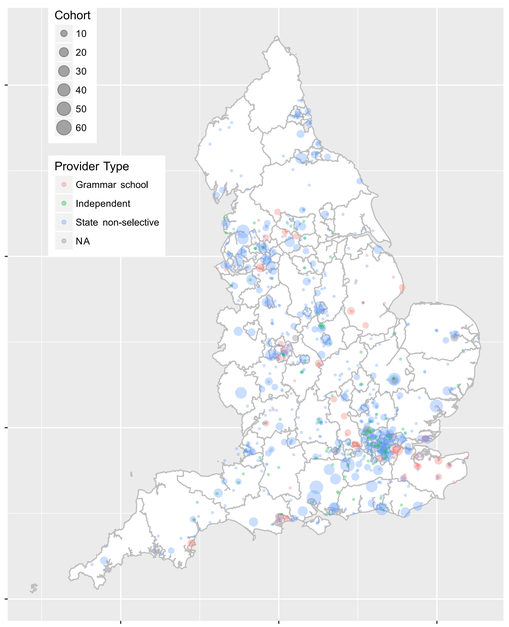


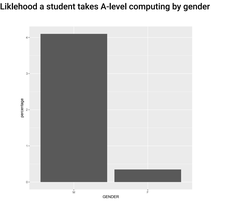


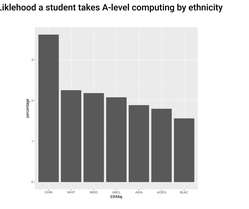


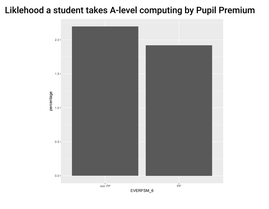


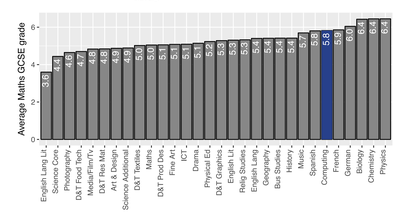


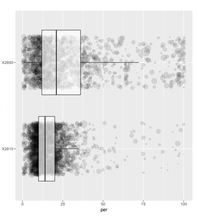


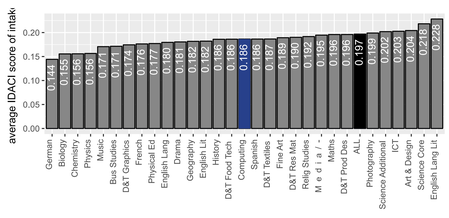
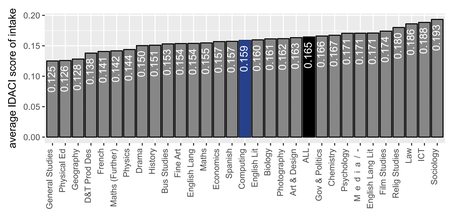
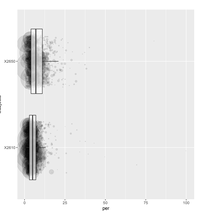
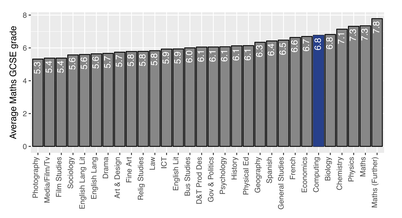












For more about computing education in the UK see: