Assessing SQ-EQ as a Programmer Aptitude Test

Killian Carty

21332673

Department of Computer Science and Information Systems

Faculty of Science and Engineering

University of Limerick

Submitted to the University of Limerick for the degree of

BSc. in Computer Systems academic year 2024/25

Supervisor: Prof. Jim Buckley

University of Limerick

Ireland

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other Irish or foreign examination board.

The thesis work was produced under the supervision of Prof. Jim Buckley at University of Limerick.

Limerick, 2025

Ethics Declaration

I herewith declare that my project involves human participants and that I have received approval from the Science and Engineering Ethics Committee prior to undertaking this part of the project. The application number for this project is: 2024_11_22_S&E

Limerick, 2025

Table of Contents

Introduction:

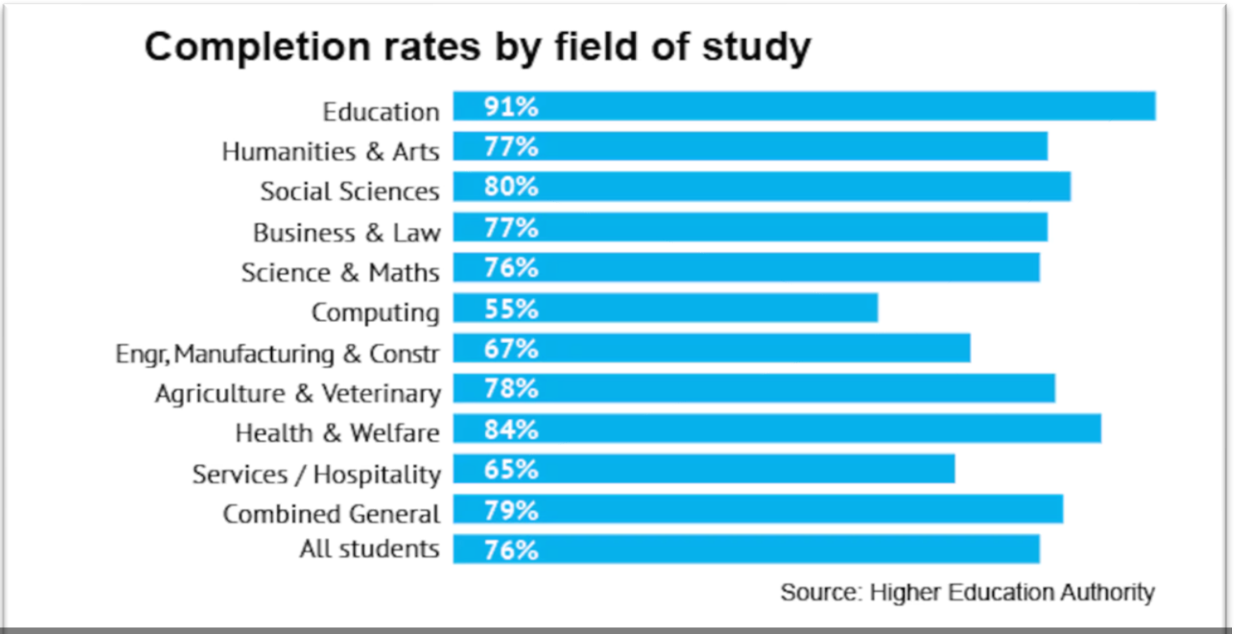
The title of my final year project is “Assessing SQ-EQ as a Programmer Aptitude Test”. The goal of the project is to determine if there is a link between SQ (Systemizing Quotient) and EQ (Empathizing Quotient) scores and programming aptitude, which previous research suggests. The project aims to replicate a previous study carried out by Stuart Wray, which concluded there was a link, but since then, other research has cast doubt on that. SQ and EQ are quotients used to assess a person’s empathising and systemizing cognitive styles (Groen et al 2015, abstract). I want to examine if a person’s SQ and EQ scores have any correlation with programming aptitude, which refers to an individual's natural ability to understand and solve coding problems, using logical reasoning, mathematical skills, and algorithmic abilities (TestGorilla, n.d.). It essentially measures your potential and natural ability to learn and perform well in programming tasks and problems. There are a number of proposed methods to measure programming ability that examine things like a person’s skills, traits and abilities to assess programming capacity. Some popular examples are using the likes of problem-solving ability, mathematical skills pattern recognition and many more which will be examined and discussed in more detail in the literature review section. The overall goal of the project is to examine if there is a link between SQ and EQ scores and programming aptitude and if this link is found develop a tool that can be used to measure a person’s programming aptitude. Being able to accurately predict programming aptitude would be an extremely powerful tool and could solve some problems within the realm of programming education.

Justification:

In this section I will outline the justifications for completing the project. Accurate aptitude prediction could result in the following:

Dropout Reduction:

One area in the fields of programming where we could see improvement with reliable aptitude tests is course dropout rates. A tool that can accurately predict a person’s aptitude for programming could significantly reduce the number of dropouts of various courses, from college degrees with a heavy coding syllabus to online programming courses. If an option to test your programming aptitude was offered before enrolment in a course, it could help people make a more educated decision on if they should pursue a future in programming. An article from the Irish Times (McGuire, P. (2019)) discussed a study that was carried out by the Higher Education Authority (HEA). This study tracked students who had enrolled in college courses over a 10-year period, monitoring if they completed their selected course or not (McGuire, P. (2019)). The HEA produced this graph displaying their findings.



Completion rates by field of study

Source: Higher Education Authority

The findings of a study by the Higher Education Authority (HEA) which has tracked more than 34,000 students who started third level over a 10-year period (McGuire, P. (2019))

As shown in the Graph the study looked at 13 fields of study and their completion rates. The average completion rate was 72% with education being the highest at 91%. However, looking at the graph there is one clear field of study where students are struggling and that is computing. With a completion rate of just 55% which is significantly lower than the average, students are struggling in computing courses. I believe that this could be due to students struggling with the programming aspects of computing. A research paper by Chin Shoon Cheah states that one reason students' struggle with learning how to program is "difficulties in understanding abstract programming concepts such as control structures and creating an algorithm to solve concrete problems" (Cheah, C.S. (2020)). Cheah paper and other research seems to suggest that students are struggling to learn and grasp the concepts of programming. Accurate programming aptitude tests could reduce this issue and improve the completion for computing courses by ensuring people who are likely to perform poorly in programming tasks are informed of this before enrolment. Aptitude tools could also allow colleges and universities to set a minimum programming aptitude score for course enrolment. This would work similarly to how some courses require certain grades in secondary school to be eligible for enrolment. For example, the University of Limerick website states that to enrol in the course "LM121 Computer Science Common Entry" Students must have Two H5 grades and Four O6 grades or four H7 grades in the leaving cert and their subjects must include Mathematics, Irish or another language, and English. Students must also score at least an O2 or H6 in maths (University of Limerick, n.d.). Universities and colleges could simply add a minimum programming aptitude score requirement for courses with a heavy programming syllabus to ensure the students who join the course are more likely to succeed in the coding aspects of the module, assuming this test is accurate of course.

Saving Resources:

Another positive result of accurate programming aptitude tests is resource saving. This is a direct result of aptitude tests causing a reduction in dropout rates and inadequate students enrolling in courses that are not suited for them. Ensuring only people with the potential to succeed are enrolled in these courses, resources are saved for both the course provider and the students considering enrolment. For the course provider such as colleges and universities programming aptitude tests ensure that teaching resources are being prioritized for students who are likely to perform well as it prevents students with poor programming potential taking spots in courses away from students who are likely to succeed. In terms of the students and the people enrolling in the courses themselves their time and money are saved. Accurate reliable programming tests can prevent people from wasting their time and money in pursuing a career in a field that simply is not for them, which instead allows them to explore other fields of study where they might excel. The tests also reassures students that have received a high programming aptitude score that their time and money is being well spent as they are likely to perform well in the field they are studying.

Exposing more people to programming:

So far, we have only discussed the positives programming aptitude tests will have by pushing people who aren't suited to study programming away from programming courses. However reliable aptitude tests could also pull people towards programming courses and encourage them to study coding. Making programming aptitude tests that are accurate, easy to complete and easily available for people could expose new people to the field of programming. It could result in someone who initially had no interest in the area of programming or presumed they would be no good at it, taking a programming aptitude test, scoring high in said test and deciding to study and pursue a career in the realm of programming and excelling. Programming aptitude test could help find and produce new elite programmers who may have never had studied the topic in the first place, if it weren't for the aptitude test.

Overall, the examples identified above portray a clear justification as to why this project should be carried out. The successful completion of this project will add to the research already in the realm of programmer aptitude and, if a link between SQ and EQ and programming aptitude is identified, potentially result in developing a tool that can accurately measure programming ability, addressing the problems highlighted above.

Motivation:

When selecting my final year project, I wanted to be sure to select something that would motivate me throughout the course of the project ensuring that it was something I would enjoy and something that would have real world implications. I have identified both personal and external motivations from this project that will push me to succeed and further justify the reasoning of completing this project.

Personal Motivations:

There is a number of personal aspects to this project that serve as a source of motivation for this project. Given that I am currently studying computer science the topic of programming aptitude and being able to accurately measure it is very interesting to me. When I personally had to select my college course after completing secondary school, I had an interest in computer science but was unsure about pursuing this interest as I had very little coding experience and exposure. A tool to give me an idea about how I could expect to perform in the coding aspects of the course would have been extremely helpful and allowed me to make a more informed decision on whether or not to select computer science as my college course. Thankfully I have stayed in the course for the 3 and a half years I have been studying it, but the same can't be said for my course mates. Over the course of nearly 4 years of studying computer science, I have seen my class size shrink every year as students dropped out or changed courses. I feel as though some of these dropouts could have been prevented if students had knowledge of their programming aptitude making them think twice about studying computer science.

I'm also motivated to complete this project as it consists of specifically studying SQ and EQ to measure programming aptitude, which is quite a niche approach with very little research in the area and conflicting results. Other methods of predicting programming aptitude, which will be mentioned in more detail in the literature review, such as using mathematical ability or problem-solving skills to predict aptitude have a lot more research and studies conducted compared to SQ and EQ. The lack of research in the area motivates me to conduct my own study to see if there is a link between SQ and EQ and programming aptitude.

External Motivations:

In terms of external motivations addressing the problems mentioned above related to high dropout rates, better resource management and programming exposure are all key motivators for this project. This also extends outside beyond traditional education in universities and college, it also applies to online programming courses and boot camps such as Codeacademy, which is an online platform for learning programming with approximately 40 million users worldwide using it to pursue career opportunities in technology (Business Wire (2021)). With platforms like coding academy growing in popularity the ability to learn to code is available to lots of people who before did not have access. People who don't have the time or money to go to college or people who have already completed a degree in a different field can now teach themselves how to code with these online tools. Programming aptitude tests can help push and pull these people in the right directions in terms of pursuing a future in programming.

The mentioned motivations above are derived from both personal experiences and external factors that justify this project and its goals.

Overview of Methodology and Goals:

This section will briefly discuss the methodology and approach that was taken to complete this project as well as identifying the various goals of the project. Both methodology and goals will be discussed in greater detail later in the report.

Methodology:

The following approach was taken to ensure that this project was completed to a high standard. This approach ensured that all the necessary deliverables for the project were produced, and the project abided by all ethics standards and University of Limerick practices. Below is a table of the components and activities for the project as well as the approximate time frame for completion.

Activity:	**Duration:**
Meet With Supervisor to Discuss project	Entire course of the project
Research and Literature Review	9 weeks
Presentation	2 weeks

Design Survey	2 weeks
Ethical Approval	10 weeks
Interim Report	2 weeks
Conduct Survey	3 weeks
Data Analysis	2 weeks
Aptitude Tool Website	2 weeks
Demo Day Poster and Video	2 weeks
Final Report	3 weeks

Deliverables:

By following the approach outlined above the following deliverables were obtained throughout the course of the project

- Project Presentation
- Literature Review
- Survey
- Consent Form
- Information Sheet
- Ethical Approval
- Interim Report
- Data Analysis
- A simple website to measure programming aptitude
- Demo Video
- Demo Poster
- Final Report

Goals:

There are two main overall goals for this project. The first is to conduct a survey on programming students, to try and determine if there is any correlation between SQ and EQ scores and programming aptitude. The second goal (if there is a correlation between SQ/EQ and programming aptitude) is to develop a simple website to act as a tool for students to test their programming aptitude by filling out questions to get their SQ and EQ scores. The aim is to have a website that anyone can use to assess their programming ability. The usability of the second goal, however, is entirely dependent on the result of the first goal finding a link between programming aptitude and SQ and EQ scores.

Literature Review:

Introduction:

The overall goal of the literature review is to introduce and explain the key terms of the project and review previous research and studies in the field of programming aptitude. The review will explore the range of techniques and strategies used to measure programming aptitude. It will examine the approach and methodologies used in these studies as well as their findings to aid in the development of our own study. By comparing and contrasting previous research in the field the aim is to identify their strengths and weakness and uses these findings to build our own programming aptitude study.

The review will begin by discussing a general overview of the various techniques and strategies that have previously been used to measure programming aptitude, such as mental model consistency, cognitive abilities, academic abilities, personal beliefs, natural language ability and more. By analysing this range of programming aptitude tests, we can examine what they have in common and use these finding in our own project. We can also compare results of various studies to get an overall view of the accuracy of current methods used to measure programming aptitude.

After getting a broader understanding of various programming aptitude methods we will shift the focus to studies that specifically investigate the relationship between SQ and EQ scores and programming aptitude. Firstly, defining what exactly SQ and EQ are and how they are measured as this will be crucial for understanding the theory behind SQ and EQ as programming predictors. Once SQ and EQ have been defined, Stuart Wray's paper will be examined in great detail as this is the paper the project is based on. The review of the Wray paper will identify the strengths and weakness of his study and discuss his results. We will then compare the methodology and approach of other SQ and EQ studies with each other and the Wray paper examining if there are consistent results when using SQ and EQ as programming aptitude predictors. This section of the review should be extremely beneficial for the development of the project and create a clear foundation for our own SQ and EQ based programming aptitude study.

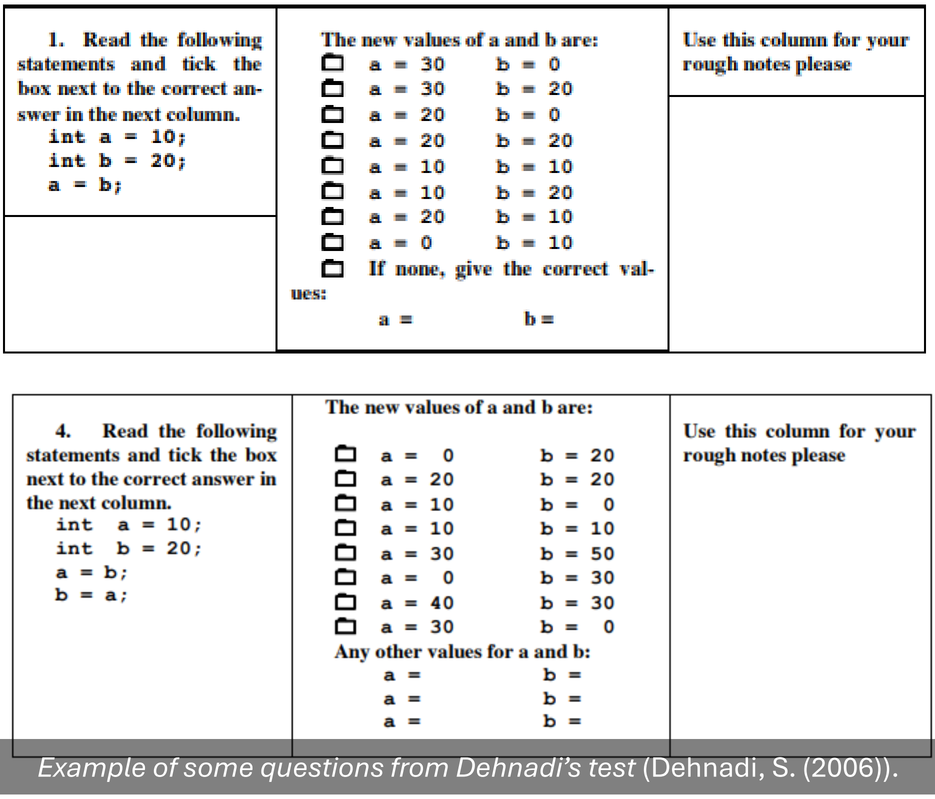
Before beginning the review, we must first define what exactly programming aptitude is. To do this we will analyse each word individually. Programming can be defined as the process of writing code to instruct computers how to perform in a desired way. Aptitude can be defined as an individual's capacity to acquire competence, skill and knowledge through training in a particular area (American Psychological Association (2018)). By combining these two definitions we can define programming aptitude as a person's capacity to acquire skills and knowledge in the field of computer programming. Essentially their potential to learn programming and perform well in the field of coding.

Overview of General Methods for Measuring Programming Aptitude:

Consistent mental models:

The first method for measuring programming aptitude we will examine is measuring the consistent use of mental models. This was a method used by Saeed Dehnadi in his 2006 paper "Testing Programming Aptitude", this is a very popular paper in the field of programming aptitude and is referenced by several other papers in this literature review. Dehnadi purposed that the consistency of which people applied mental models could be a good predictor of programming aptitude. He began with the hypothesis that you could identify a group to represent novice programmers by examining their problem-solving methods and observing the consistent mental models used (Dehnadi, S. (2006)). A study was designed to test this theory.

The study consisted of 60 subjects enrolled in an introductory programming course at Middlesex university. They were asked basic assignment questions in Java before the course began and their answers were recorded. Below are examples of the questions the participants were asked:



1. Read the following statements and tick the box next to the correct answer in the next column.
`int a = 10;`
`int b = 20;`
`a = b;`
	The new values of a and b are:
	□ a = 30 b = 0
	□ a = 30 b = 20
	□ a = 20 b = 0
	□ a = 20 b = 20
	□ a = 10 b = 10
	□ a = 10 b = 20
	□ a = 20 b = 10
	□ a = 0 b = 10
	□ If none, give the correct values:
	a = b =
		Use this column for your rough notes please

4. Read the following statements and tick the box next to the correct answer in the next column.
`int  a = 10;`
`int  b = 20;`
`a = b;`
`b = a;`
	The new values of a and b are:
	□ a = 0 b = 20
	□ a = 20 b = 20
	□ a = 10 b = 0
	□ a = 10 b = 10
	□ a = 30 b = 50
	□ a = 0 b = 30
	□ a = 40 b = 30
	□ a = 30 b = 0
	Any other values for a and b:
	a = b =
	a = b =
	a = b =
		Use this column for your rough notes please

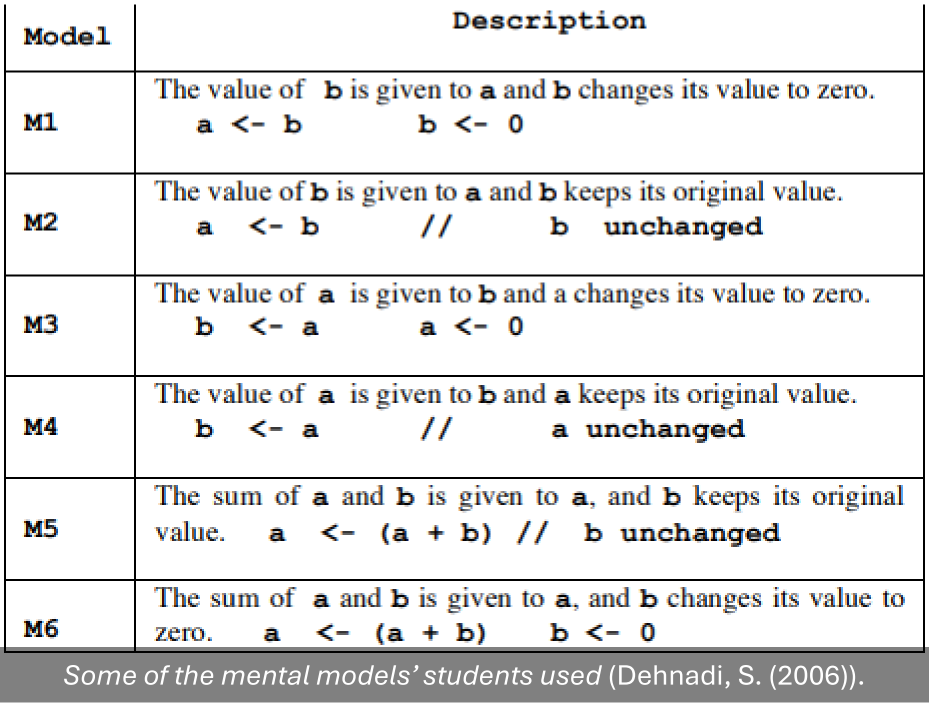
Example of some questions from Dehnadi's test (Dehnadi, S. (2006)).

An example of one of the approaches taken for question 1 was overriding the value of a to contain the value of b and changing the value of b to 0, resulting in a = 20 and b = 0. Another method used was taking the two values of both a and b were swapped, resulting in a = 20 and b = 10. The traditional "Java" approach would result in the value of b being assigned to a, with b keeping its original value, resulting in a = 20 and b = 20. However, the goal of the test was not to see if students correctly used the Java mental model, it instead examined the consistency of the students regardless of the model they used.

Based on the answers the participants gave, Dehnadi noted that the students were split into distinctive groups, “it became clear that they divided into three clear groups ... consistent group ... inconsistent group and the blank group” (Dehnadi, S. (2006)). The characteristics defining the 3 groups that emerged was the consistency of their answers.

- Consistent Users: This group contained about 50% of the total subjects. Students in this group applied the same mental model across all tasks.
- Inconsistent Users: Made up a third of the total subjects. These students shifted between various models throughout the assessment.
- Blank Group: This group accounted for approximately a sixth of the group. This group refused to answer all or almost all the questions.

Dehnadi accounted for 8 possible models to answer the questions, but 3 more models were discovered from the student’s responses. Here are some examples of the models used to answer the assignment questions:



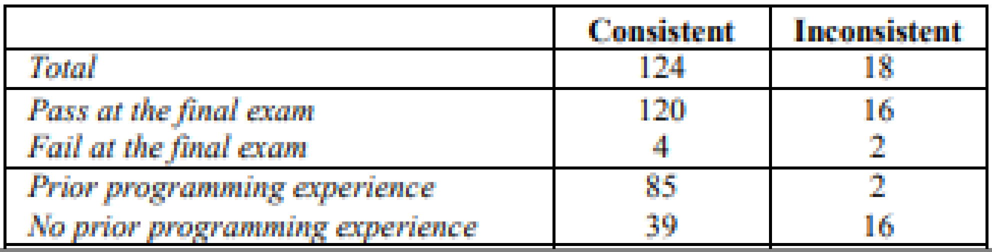
Model	Description
M1	The value of **b** is given to **a** and **b** changes its value to zero. `a <- b        b <- 0`
M2	The value of **b** is given to **a** and **b** keeps its original value. `a  <- b      //      b  unchanged`
M3	The value of **a** is given to **b** and a changes its value to zero. `b  <- a        a <- 0`
M4	The value of **a** is given to **b** and **a** keeps its original value. `b  <- a      //      a unchanged`
M5	The sum of **a** and **b** is given to **a**, and **b** keeps its original value. `a  <-  (a + b)  //  b unchanged`
M6	The sum of **a** and **b** is given to **a**, and **b** changes its value to zero. `a  <-  (a + b)    b <- 0`

Some of the mental models’ students used (Dehnadi, S. (2006)).

The results from the Java assignment test and the official programming course results were compared. The study concluded that ““success in the first stage of an introductory programming course may be predictable by examining the way that students approach a basic programming problem even before they have had any contact with programming notation” (Dehnadi, S. (2006)). Dehnadi’s findings seem to suggest a correlation between consistent model use and programming performance, showing that student who used consistent models in the Java assignment test performed better in the programming course, even if the model they used was different to the traditional Java method, all that mattered was the consistency not the model used.

Following Dehnadi’s paper claiming that mental models could predict programming aptitude the study was replicated by Michael Caspersen, Jens Bennedsen and Kasper Larsen in a paper titled "Mental Models and Programming Aptitude”. In reference to Dehnadi’s test they stated, “We have repeated their test but … we could not show that the test predicts students’ success in our introductory programming course” (Caspersen et al. (2007)).

Caspersen et al. (2007) aimed to replicate the study as follows. They subjects would be 142 students from the University of Aarhus enrolled in an introductory, object orientated programming course. The students were administered the same questionnaire, regarding Java assignment statements, created by Dehnadi in the first week of the programming course. The students were split into 5 groups based on their consistency in their answers from C0 to C4 with C0 being the most consistent and C4 being least consistent. Students in C0 were considered consistent and students in any other category were labelled inconsistent. The students result in the programming aptitude test was compared to their performance in the final exam producing this table:



	Consistent	Inconsistent
Total	124	18
Pass at the final exam	120	16
Fail at the final exam	4	2
Prior programming experience	85	2
No prior programming experience	39	16

Results of the final exam compared to consistency of answers (Caspersen et al. (2007)).

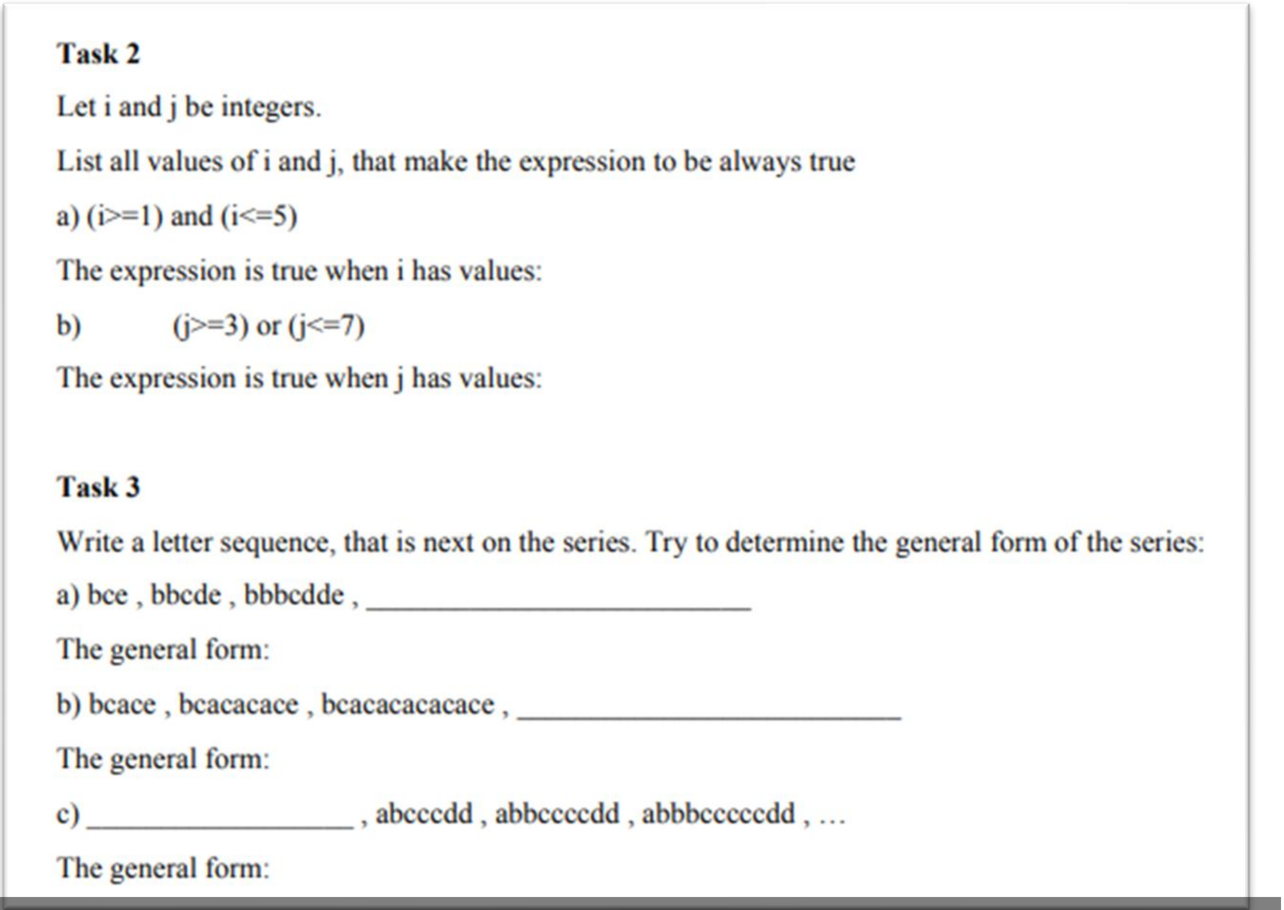
They study found no correlation between mental model consistency and exam performance, producing a contradictory result to the one seen in Dehnadi’s study. To try and explain the differences in results the student who answered inconsistently in Dehnadi’s test but still passed the courses final examination were interviewed, it was found that they all started off with some sort of mental model for completing the questions. They concluded that the problem for these students was that “the model they started out with failed at some point before the end of the test. Not knowing about the purpose of the test, and not considering it important, none of the students cared to backtrack to find a viable model. They simply altered their model and went on from there” (Caspersen et al. (2007)). Essentially the students changed their model during the test after finding the initial model ineffective. This was identified as an error in Dehnadi’s test claiming the only thing the test measured was students, “guessing capabilities” (Caspersen et al. (2007)). Caspersen concluded that after testing the hypothesis of using mental models to predict programming ability his results provided “an unequivocal rejection of the hypothesis” (Caspersen et al. (2007)).

The contradicting results from Dehnadi’s and Caspersen’s studies show the issues with measuring mental models as a predictor for programming success. Dehnadi’s findings seem to suggest strong evidence of a correlation between the two, but Caspersen’s replication has contrasting results. This highlights the difficulty with finding accurate ways to measure programming aptitude in general as just because a method showed positive results in a single study does not guarantee it is a reliable predictor. Caspersen’s replication study does not completely disprove Dehnadi’s hypothesis in the same way that Dehnadi’s results do not prove that mental models is a foolproof way of predicting programming aptitude. The two contrasting results simply suggest that more research is required.

Cognitive and Scholastic Abilities and personality traits:

Another popular research point for programming aptitude is measuring cognitive and scholastic abilities. Cognitive abilities referring to skills associated with perception, learning, memory, understanding, awareness reasoning, judgement, intuition and language (American Psychological Association (2018)) and scholastic meaning “relating to school or education” (Cambridge Dictionary (n.d.)). This section examines 3 studies testing the relationship between these abilities and measuring programming aptitude.

The paper titled "Programming aptitude testing as a prediction of learning to program" by Markku Tukiainen and Eero Mönkkönen (Tukiainen, M. and Mönkkönen, E. (2002)) investigates if aptitude tests for scholastic aptitude and programming aptitude could predict success in a student’s first programming course. The study involved 33 first year business information students. The following data was collected from the students: Results of a scholastic aptitude test which measure logical, verbal and learning aptitude, results of a programming aptitude test and results of the final exam for the programming course. The scholastic and programming aptitude tests were carried out before the course began. The scholastic test was carried out by an 3rd party external company and the programming aptitude tests the students took was the “Houman programming aptitude test”.



Task 2

Let i and j be integers.

List all values of i and j, that make the expression to be always true

a) (i>=1) and (i<=5)

The expression is true when i has values:

b) (j>=3) or (j<=7)

The expression is true when j has values:

Task 3

Write a letter sequence, that is next on the series. Try to determine the general form of the series:

a) bce , bbcde , bbbcdde , ___________________________

The general form:

b) bcace , bcacacace , bcacacacacace , ___________________________

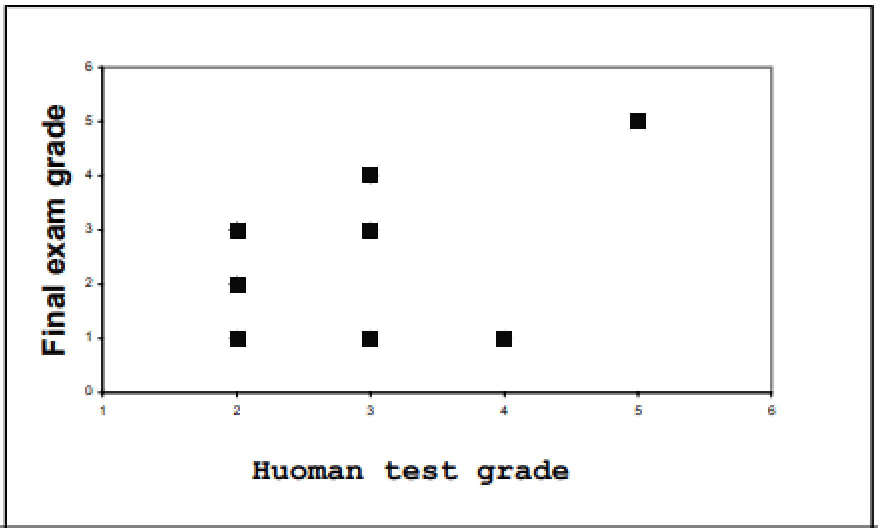
The general form:

c) ___________________ , abcccdd , abbccccdd , abbbcccccdd , …

The general form:

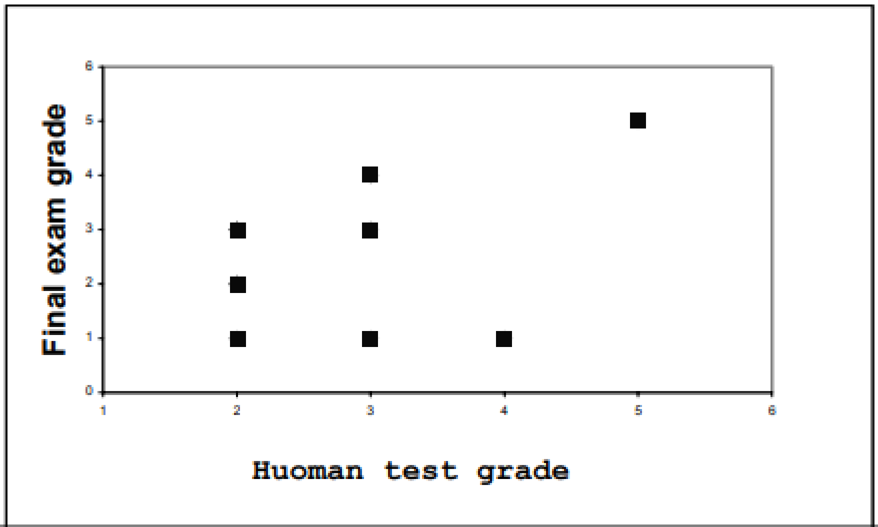
Examples of some of the tasks from the Houman programming aptitude test Mönkkönen (Tukiainen, M. and Mönkkönen, E. (2002))

The data collected was compared to the students results in their final exam and the following results were found. There was no significant link found between the scholastic aptitude test and final exam score. Upon first investigation the programming aptitude test had a very significant correlation with final exam results. However when the students were divided into 2 groups based on prior programming experience, it was found that the Houman test gave little indication of success for the group with no programming experience, they noted that “The only noticeable trend could be that if a student is very successful in the Huoman test (grades 4 to 5), then she will be successful in the programming test as well” (Tukiainen, M. and Mönkkönen, E. (2002)).



Houman test and final exam results comparison for group with no prior programming experience (Tukiainen, M. and Mönkkönen, E. (2002)).

There seemed to be slightly more promising results for the Houman test regarding the group of students who had prior coding experience, as the test gave a “floor limit” to the success in the programming test.

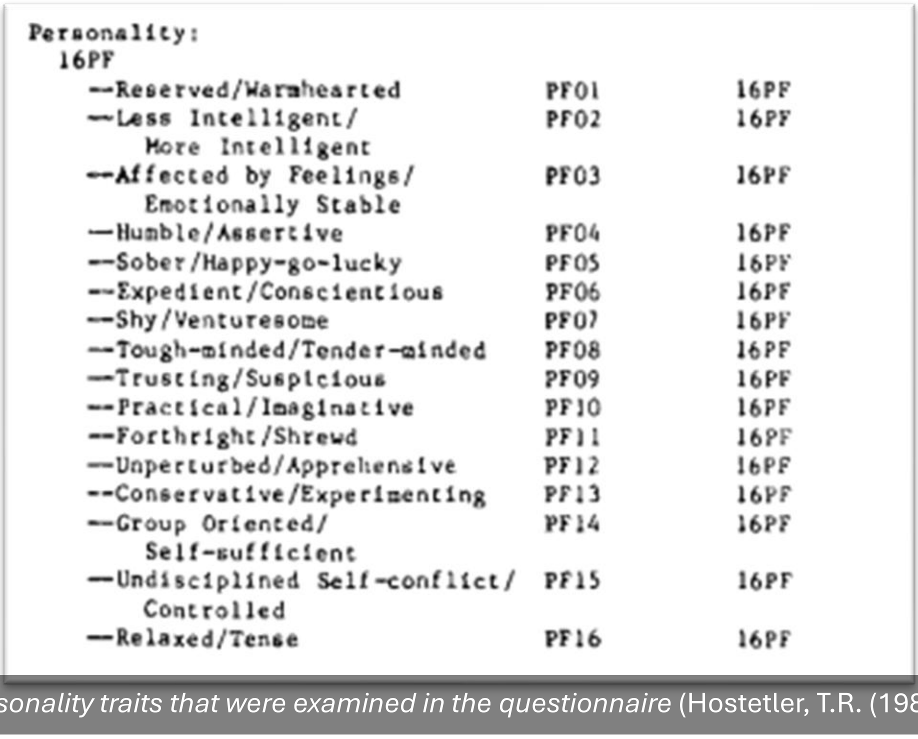


Houman test and final exam results comparison for group with prior programming experience (Tukiainen, M. and Mönkkönen, E. (2002)).

The paper "Predicting Student Success in an Introductory Programming Course" by Terry R. Hostetler examines cognitive skills, personality traits and past academic performance as predictors for programming success (Hostetler, T.R. (1983)). His study consisted of 79 students in the University of Illinois studying the course titled “Introduction to computers and their application to business and commerce”. The course used the following tools for assessment: programming assessments, two one-hour exams and a final three-hour exam. The data for the proposed predictors were collected as follows.

Cognitive skills: Two tests from the computer programmer aptitude battery (CPAB), an aptitude test created by Jean M. Parlormo (Hostetler, T.R. (1983)), were selected. The reasoning test described as “a test of ability to translate ideas and operations from word problems into mathematical notations” and the diagramming test described as “a test of ability to analyse a problem and order the tests for solution in a logical sequence” (Palormo, J.M. (1967)).

Personality traits: Collected using a questionnaire measuring 16 personality traits.



Personality:
16PF

--Reserved/Warmhearted	PF01	16PF
—Less Intelligent/ More Intelligent	PF02	16PF
--Affected by Feelings/ Emotionally Stable	PF03	16PF
—Humble/Assertive	PF04	16PF
--Sober/Happy-go-lucky	PF05	16PF
--Expedient/Conscientious	PF06	16PF
--Shy/Venturesome	PF07	16PF
—Tough-minded/Tender-minded	PF08	16PF
—Trusting/Suspicious	PF09	16PF
--Practical/Imaginative	PF10	16PF
—Forthright/Shrewd	PF11	16PF
—Unperturbed/Apprehensive	PF12	16PF
--Conservative/Experimenting	PF13	16PF
—Group Oriented/ Self-sufficient	PF14	16PF
—Undisciplined Self-conflict/ Controlled	PF15	16PF
—Relaxed/Tense	PF16	16PF

Personality traits that were examined in the questionnaire (Hostetler, T.R. (1983)),

Academic performance: Measured using students GPA and mathematics background.

The cognitive and personality tests were administered during the first 2 weeks of the semester. GPA was collected from the registrar at the beginning of the semester and math background was measure as an integer based on the most difficult math class completed with a higher integer meaning a more advanced class. The student’s overall performance in the course was an accumulation of their performance in both the assignments and examinations. The results found that the diagramming test was the most significant predictor of success with GPA and reasoning test also showing correlation. It was found that no individual personality trait showed significant correlation. A model containing both CPAB tests, GPA, mathematics and personality factor 5 (Happy-go-lucky) was developed and correctly classified 77.2% of the students into high and low aptitude groups.

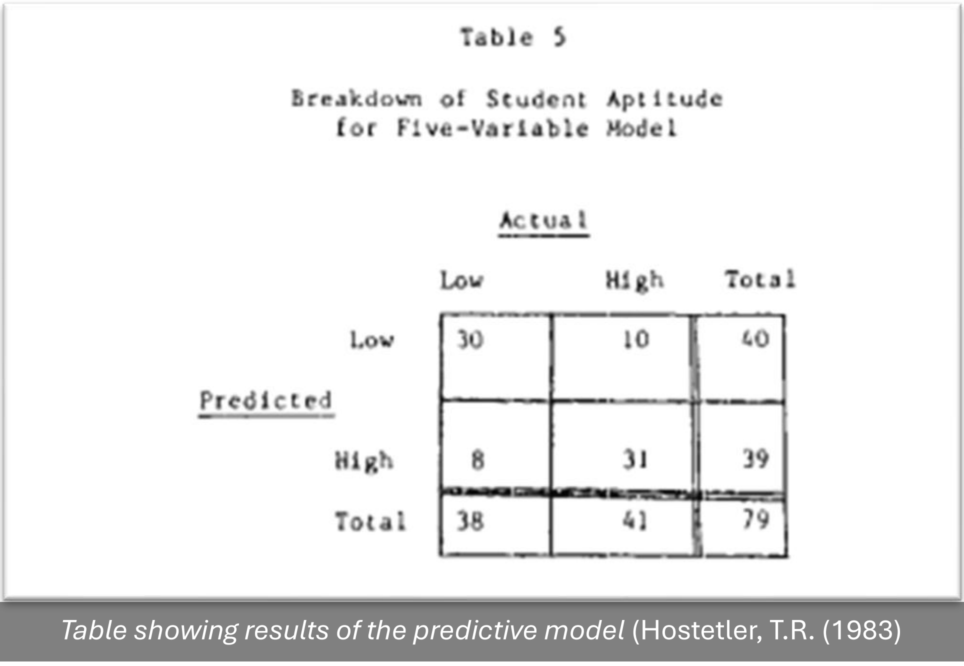


Table 5

Breakdown of Student Aptitude for Five-Variable Model

		Actual		
		Low	High	Total
Predicted	Low	30	10	40
	High	8	31	39
	Total	38	41	79

Table showing results of the predictive model (Hostetler, T.R. (1983)

The study developed a model that seemed to predict programming aptitude with very good accuracy, however Hostetler did state that most of the variance remain unexplainable, "This study's model explained approximately 43% of the variance in the final numerical scores of students, leaving a majority of the variance unaccounted for." (Hostetler, T.R. (1983), suggesting more research is required.

The paper "Identification of Some Components of Computer Programming Aptitude" by Carol Ann Alspaugh also examines personality traits as programming aptitude predictions, as well as mathematical background and existing aptitude tests. The goal of the study was to examine if the following could predict programming ability.

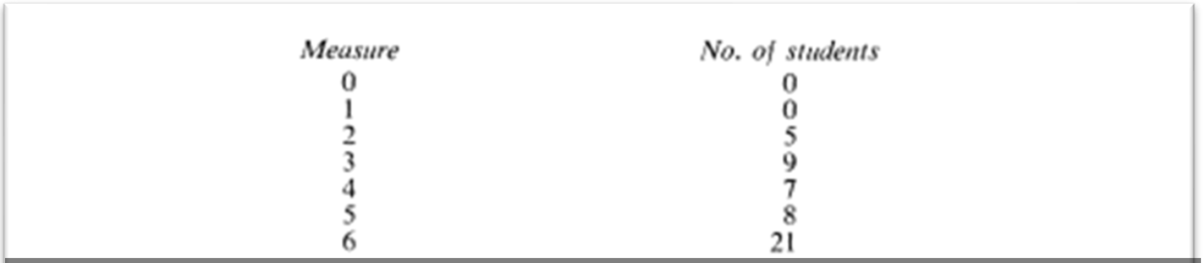
- Existing tests in the form of the Thurstone Temperament Schedule (that measures personality traits), IBM Programmer Aptitude Test (measuring programming aptitude), Watson-Glaser Critical Thinking Appraisal (measuring thinking skills), and SCAT Verbal and Quantitative subtests (measuring verbal and quantitative reasoning).
- Mathematical ability.

Alspaugh also wated to assess if the type of programming language was a factor, so she developed a study to measure to what extent these factors "correlate with a measure of proficiency in symbolic language programming and in algebraic language programming" (Alspaugh, C.A. (1972)).

The study involved 50 students enrolled in a programming course in the University of Missouri-Columbia. This was an ideal course for the study as students would learn a symbolic language in the form of assembly during the first half of the semester and an algebraic language in the second half of the semester in the form of FORTRAN. The aptitude and personality test were administered in the first week of the semester and the SCAT Quantitative and Verbal test scores were obtained from the university records as all students had taken this test before enrolment. Mathematical ability was measure using the following scale:

- 0-no mathematics courses
- 1-one or two units of high school mathematics
- 2-three or four units of high school mathematics or College Algebra
- 3-College Algebra plus a non-calculus course
- 4-Calculus I
- 5-Calculus II
- 6-above Calculus II

Producing the following results:

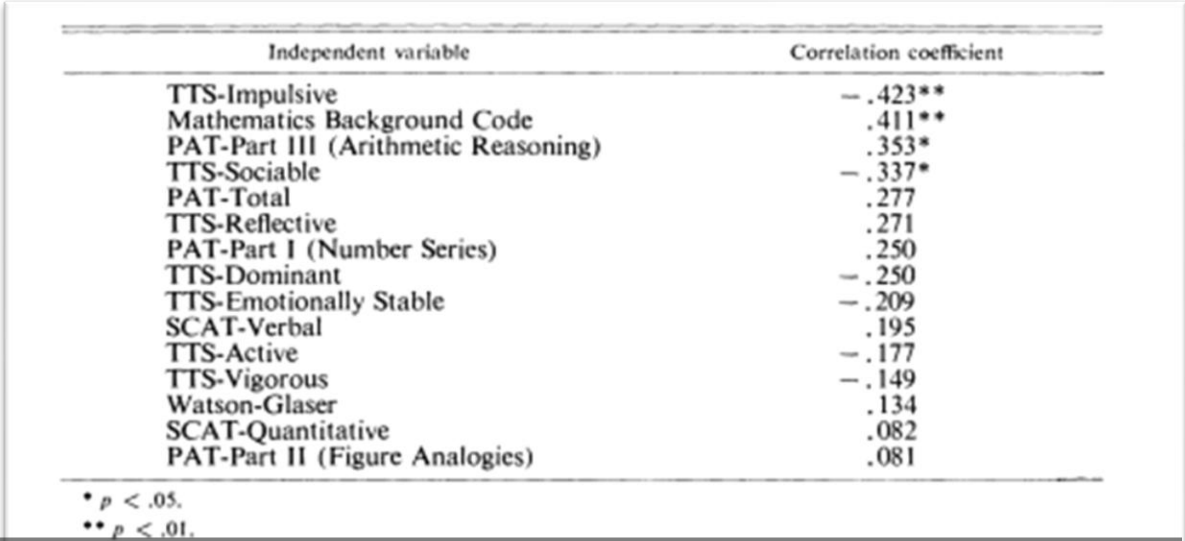


Measure	*No. of students*
0	0
1	0
2	5
3	9
4	7
5	8
6	21

Distribution of mathematical ability based on scale used (Alspaugh, C.A. (1972))

Programming ability was measure in the form of 3 variables. Performance in both programming languages individually in the form of examinations, and a combined overall score in the course.

The study produced very interesting results. When examined individually both Impulsiveness (measure by the Thunderstone temperament schedule) and mathematical background were among the strongest predictors of programming performance across both languages, finding that lower impulsiveness and higher mathematical ability led to better performance in both languages. Unsurprisingly these two factors were the strongest predictions for the combined overall results also, suggesting that similar aptitude requirements were needed for the 2 languages. Another interesting result was that low social abilities also showed strong correlation with success.



Independent variable	Correlation coefficient
TTS-Impulsive	−.423**
Mathematics Background Code	.411**
PAT-Part III (Arithmetic Reasoning)	.353*
TTS-Sociable	−.337*
PAT-Total	.277
TTS-Reflective	.271
PAT-Part I (Number Series)	.250
TTS-Dominant	−.250
TTS-Emotionally Stable	−.209
SCAT-Verbal	.195
TTS-Active	−.177
TTS-Vigorous	−.149
Watson-Glaser	.134
SCAT-Quantitative	.082
PAT-Part II (Figure Analogies)	.081

* $p < .05$.
** $p < .01$.

The independent variables and their correlation with students' performance in the course overall (Alspaugh, C.A. (1972))

The overall findings suggest that mathematical ability is a very important requirement for programming success. They also suggest that low impulsiveness and social ability play a role in programming success. In terms of the other aptitude tests examined it was found that the IBM programming aptitude test showed correlation with programming performance, but the other aptitude tests had little predictive abilities. The findings also show that the programming language was not a factor and consistent aptitude characteristics were seen across both programming languages.

To conclude the three papers, reveal interesting results in the realm of cognitive and scholastic abilities and personality traits in terms of their programming success prediction abilities. Across all studies mathematical ability stood out as a clear predictor of success in terms of coding suggesting a strong link between mathematics and programming. However, the papers also showed some contrasting findings. Alspaugh found that personality traits such as low impulse and social abilities could perhaps predict programming aptitude but Hostetler measure 16 personality traits as predictors for programming success and found very little correlation between single personality traits and programming aptitude. The contrasting and conflicting finding between the papers show again the difficulty in accurately measuring factors and their abilities to predict programming aptitude.

Mindsets and Thinking Styles:

This section will examine previous research in area of using thinking styles and mindsets to predict programming aptitude. It discusses studies that looks at people's way of thinking and their overall mindset towards programming to try and identify ways to predict programming ability. The paper "Programming: Factors that influence success" by Susan Bergin and Ronan Reilly (Bergin, S. and Reilly, R. (2005)) examines how both thinking styles and mindset, as well as other factors could predict programming ability. The study took place in Maynooth University involving 96 students enrolled in a first-year programming course. 15 factors were measure in total, but we will only be focussing on the factors relating to students' mindsets and thinking styles. A questionnaire and test were used to obtain the following information from the students that would be used for the study.

- Comfort level in the module.
- Perceived understanding in the module.
- Work style preference (alone or group).
- Encouragement received to study computer science.
- Numerical and letter sequencing.
- Arithmetic reasoning.
- Problem translation skills.
- Logical ability.

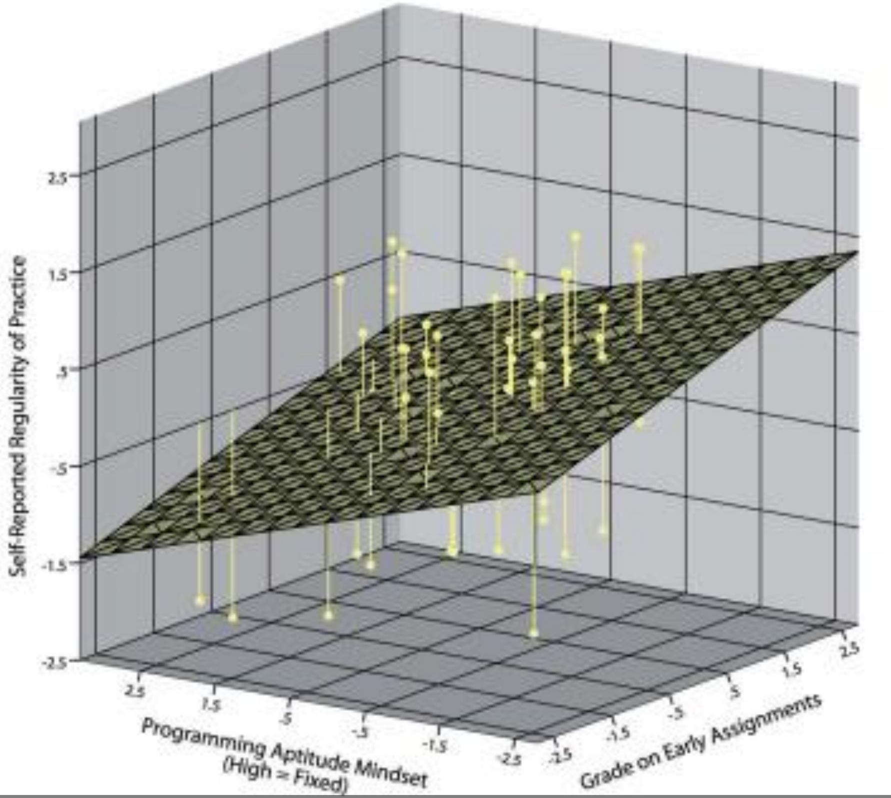
Performance in the course was measure based on the overall grade in the course which consisted of 30% continuous assessment and 70% final examination. Of the 96 students enrolled 80 completed the questionnaire that collected personal information from the students and 30 completed the test that assess students thinking styles. When the final grades had been calculated, they were compared to the students results from the questionnaire and test. The results found a weak overall correlation between the test measuring thinking styles and the overall grade in the module. It was found that individual factors such as arithmetic reasoning were more predictive than other suggesting that further tests could be developed to investigate this further "a number of items in the test were highly correlated with programming performance. We anticipate that a redesign of the test could result in more significant findings in the future.” (Bergin, S. and Reilly, R. (2005)). Variables such as work style preference and encouragement received also showed very weak correlation with success. However, surprisingly out of the 15 factors measured, the 2 strongest predictors of performance were comfort level and self-perceived understanding of the module, with the authors noting “A very strong correlation between a student’s perception of their understanding of the module and programming performance" (Bergin, S. and Reilly, R. (2005)). These findings are very interesting highlighting that the students’ mindsets were played a crucial role in their success in the module suggesting that more research needs to be done in this area of programming ability prediction. This study highlights the many different ways that can be used to measure programming aptitude outside the traditional factors such as mathematical ability and problem-solving skills.

Another paper that analysis a person’s mindset as a factor for predicting programming aptitude is "On the Domain-Specificity of Mindsets: The Relationship Between Aptitude Beliefs and Programming Practice" by Michael J. Scott and Gheorghita Ghinea (Scott, M.J. and Ghinea, G. (2014)). They examine how students’ personal beliefs about their programming aptitude and their mindset towards it influences programming ability and performance. The paper aims to address the following “research questions”:

- “RQ1) Can students have a mindset for programming aptitude that is substantially different to their mindset for general intelligence?
- RQ2) Does the mindset for programming aptitude have more utility for predicting programming practice compared to the mindset for general intelligence?
- RQ3) Does the mindset for programming aptitude change differently to the mindset for general intelligence over a period of programming instruction?”

(Scott, M.J. and Ghinea, G. (2014))

The study was carried out on first- and second-year students enrolled in programming courses. The students took 2 surveys, one in the 8th week of their course and another in the 16th. These surveys collected data around the student’s mindset in relation to their overall intelligence and programming aptitude, the surveys also examined the student’s practices and behaviours in relation to programming “providing an indication of frequency of practice and the typical duration of each practice session.” (Scott, M.J. and Ghinea, G. (2014)). The students result in their first 3 assessments were also measure as a way to measure programming performance. Out of the 296 students enrolled in both the first- and second-year courses 63 students responded to both surveys, pointing out one of the limitations of this study only represents a small proportion of the 2 courses. The results found that the students had sperate beliefs about programming aptitude and general intelligence aptitude with students having a growth mindset in terms of intelligence but a fixed mindset for programming aptitude. This can be seen as 31% of students said programming aptitude was fixed, but only 28.7% seeing intelligence as fixed. This seems to suggest that some students feel there is a difference between overall intelligence and programming ability and that programming ability is something that you cannot improve. These results when compared to the students programming practices and grades showed interesting results. It was found that with a growth mindset towards programming aptitude practiced more regularly suggesting that their perception of programming aptitude being something that could be improved encouraged them to practice more. The opposite effect was seen for students with a fixed mindset, they practiced less, especially of their grades in assessments were low. In terms of the grades, it was found that students with higher grades were less influenced by their mindset towards programming aptitude.



Influence of programming aptitude mindset and performance on early assignments on students' programming practice. (Scott, M.J. and Ghinea, G. (2014))

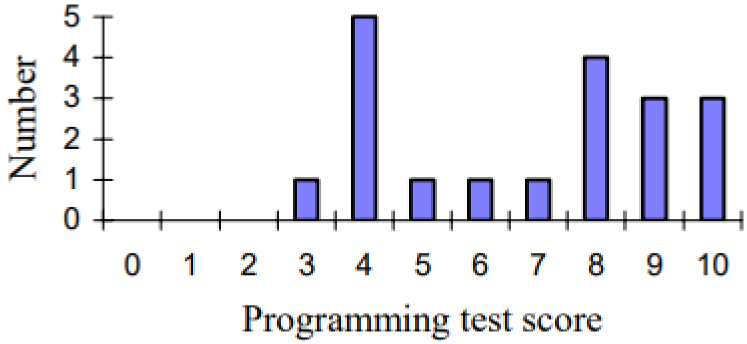
Overall, the paper seems to suggest that some students perceive overall intelligence and programming intelligence and two distinct categories of intelligence. Students who believed that their programming aptitude was something that they could grow seemed to practice more frequently and as a result performed better overall. In contrast, students who felt their aptitude in programming was fixed were less likely to practice especially once receiving poor grades suggesting that the assessment results caused them to lose confidence and intertest in programming.

To conclude both studies seems to show that a person's mindset in terms of programming can be a decisive factor in their programming ability. Bergin and Reilly examined 15 factors and determined that a person's mindset in terms of comfort and perceived understanding of programming as the 2 strongest predictors for programming ability suggesting that a person's mindset plays a crucial role in terms of their performance in a coding environment. Scott and Ghinea's research examine if a person's belief that their aptitude is fixed or has the potential to grow has an effect on their overall ability, with their research suggesting that those who feel they can improve their programming aptitude, are more likely to practice and in turn are more likely to perform well in programming tasks.

SQ and EQ as programming aptitude predictors:

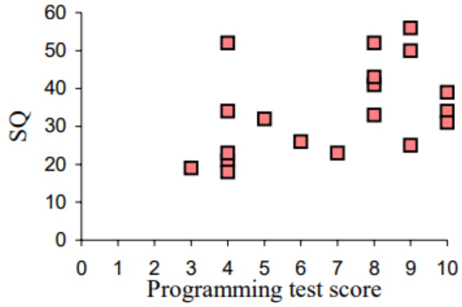
This section of the literature review will focus on programming aptitude research regarding systemizing quotient (SQ) and empathy quotient (EQ) as ways to predict programming potential. SQ or systemizing quotient, refers to “the drive to analyse and explore a system, to extract underlying rules that govern the behaviour of a system; and the drive to construct systems” (Baron-Cohen, S. (2003)) (The Guardian, (n.d.)). It essentially measures a person’s interest in systems and how they work. Empathy quotient (EQ) is intended to “measure how easily you pick up on other people's feelings and how strongly you are affected by other people's feelings” (American College of Surgeons (n.d.)). These two quotients have been identified as possible ways to predict programming aptitude.

The paper “SQ Minus EQ can Predict Programming Aptitude” by Stuart Wray seems to suggest that SQ and EQ is a viable method to measure programming potential (Wray, S. (2007)). This is the study that the project is built on as we will be recreating Wray’s study and examining if we can reproduce his results. Wray’s study involved 19 students enrolled in a telecommunications system engineering program. Students were administered a 10-question programming test to assess programming ability in the form of a python test examining understanding of object orientated programming. Students were also giving SQ and EQ tests from the University of Cambridge, to measure their scores for the respective quotients. The programming test was conducted in June 2006 and the EQ and EQ test were performed 5 months later in November. The following graph shows the results of the programming test:



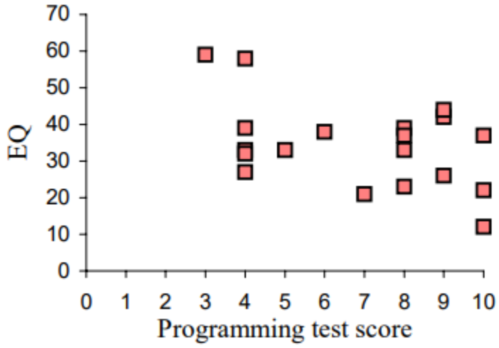
Graph showing the programming results (Wray, S. (2007))

As you can see a lot of students performed quite well with 10 students score 8 out of 10 or higher, it was noted in the paper that “this test should perhaps have been harder, since there is a cluster of marks at the high end” (Wray, S. (2007)). SQ and EQ scores were compared with the programming scores first individually, then as a combined measure in the form of SQ minus EQ. SQ as an independent variable showed moderate correlation with performance in the programming test.



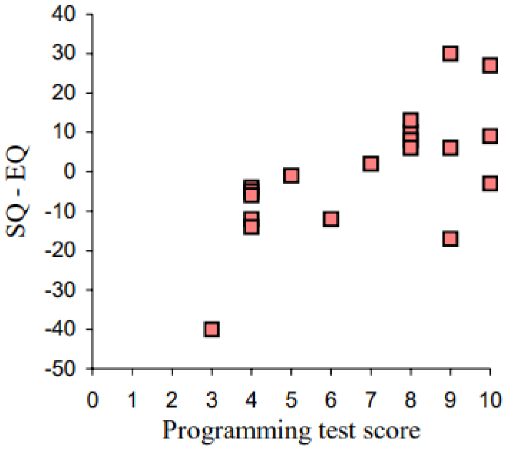
SQ scores compared with test scores (Wray, S. (2007))

EQ showed moderate negative correlation with students’ scores in the programming test.



EQ scores compared with test scores (Wray, S. (2007))

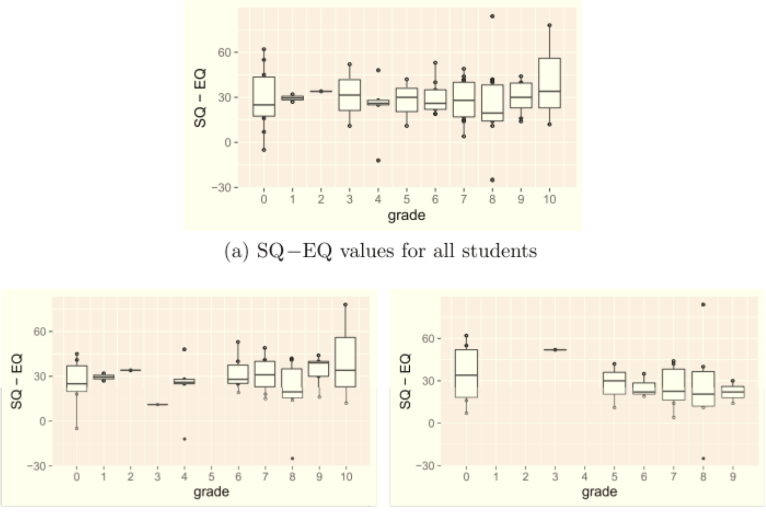
However, when the two variables were combined as SQ – EQ it showed a strong high correlation.



SQ-EQ compared with test scores (Wray, S. (2007))

The results from Wray's study suggest that the difference between SQ and EQ is a strong predictor of programming aptitude as it concluded that students with a high SQ score, and a low EQ score were more likely to perform better in the programming assessment. However, there are some limitations to Wray's study. The small sample size of only 19 students could be seen as an issue in the study as it is not a representation of the general population. Also, all these students were male and studying the same course. The performance of the students in the programming assessment also suggest that this test was perhaps too easy, something that was noted by Wray in his paper. The claim from Wray that "SQ minus EQ can predict programming ability" combined with some of the issues highlighted is one of the reasons why I would like to perform the same study and analyse the results. Wray's study has already been reproduced by Juris Borzovs in the paper "Can SQ and EQ Values and Their Difference Indicate Programming Aptitude to Reduce Dropout Rate?" (Borzovs et al., (2017)).

Borzovs paper aimed to replicate Wray's study and examine if SQ and EQ values could truly predict programming aptitude. The study was conducted on 73 first year computing students, 29 females and 44 males, at the University of Latvia. The students completed SQ and EQ questionaries to obtain their scores for the quotients. The students' final grades in the course were used to measure programming performance. The study used SQ and EQ as individual variables for predicting aptitude and the combined variable of SQ – EQ. It also analysed their predictive power on the group of students as a whole but also on the male and female subgroups. The study found weak or no correlation between any of the variables and programming performance. The male group showed a weak positive correlation with SQ and their grades but none for EQ or SQ-EQ. The female subgroup showed weak insignificant negative correlation with SQ and EQ and course results and none for SQ-EQ. When examining the whole group there was no significant correlation between SQ, EQ or SQ-EQ and programming grades.



(a) SQ−EQ values for all students

(b) SQ−EQ values for male students

(c) SQ−EQ values for female students

SQ-EQ values compared with the grades for each group (Borzovs et al. (2017))

The study concluded that "There is no evidence of correlation between SQ, EQ or SQ-EQ and the final grade" (Borzovs et al., (2017)). The findings from Borzovs study are very interesting and contradict the finding from Wray. Wray's research seems to suggest a clear strong relationship between SQ and EQ and programming aptitude however, Borzovs replication of the study suggests that this is not the case. The two contradicting results is motivation for the study to replicated once again and compare the findings with previous research.

Methodology:

This section will discuss the methodology, deliverables and goals described in the introduction in detail describing what exactly was done and how it was achieved. It outlines the components of the project that have been completed to give an in depth understanding of how the project was completed. The methodology has been divided up into the following components.

Meet with Supervisor:

To begin the project, I met with my project supervisor, Jim Buckley, who had proposed the title "Assessing SQ-EQ as a Programmer Aptitude Test" as a project he would be open to supervising for students doing a final year project. The project consisted of recreating a previous study conducted by Stuart Wray to analyse if there is a link between SQ and EQ score and programming aptitude. Originally the project was entirely study based with no technical aspect to it. After discussing this, we altered the project to include some practical coding aspects. We decided to use the statistical programming language R for the data analysis, python for creating visualisations and developed a website that could be used as a tool to measure programming aptitude based on our data analysis of the link between SQ, EQ and programming ability. We discussed and refined the project ensuring that we were both on the same page and organised bi-weekly meetings Thursdays at three o'clock. In these meetings we discuss the current state of the project, any work completed in between meeting and what is expected for the next meeting. These meetings were very important for ensuring the project's success.

Research and Literature Review:

The research and literature review were a vital part of the project as it established the foundation and context of the project. For this project it was crucial to research the topic of programming aptitude in great detail to understand the current research and studies conducted in the area, so that we could use previous research to aid in the development of the project. To begin Google Scholar was used to find papers based on ways of measuring programming aptitude. Using search prompts such as "Measuring programming aptitude" and "Predicting programming ability", google scholar returned a number of papers and results with twenty citations were used for the review. The literature review is structured to examine all aspects of programming aptitude. It begins by reviewing various methods and approaches to measure programming aptitude as well as their results, before narrowing the focus on papers discussing SQ and EQ and their relation to programming ability. The main paper the project is based on is the paper "SQ Minus EQ can Predict Programming Aptitude" by Stuart Wray. This paper was supplied to me by my project supervisor Jim Buckley and would be the foundations for the project. It was import for the literature review to contain other papers based on examining SQ and EQ as measures for programming ability, so the google scholar prompts were updated to contain SQ and EQ to find more research. As stated previously SQ and EQ as a programming aptitude measure is quite a niche area of research so there was few results, but the paper "Can SQ and EQ Values and Their Difference Indicate Programming Aptitude to Reduce Dropout Rate?" by Juris Borzovs was found. This paper would be extremely beneficial to the project as it discussed and replicated the study from Stuart Wray's paper. Unfortunately, it was located behind a paywall but after emailing the author of the paper a copy was acquired. Overall, the research and literature review are vital parts of the project as it provides an insight into previous research and studies in the area of programming aptitude offering a broad understanding of the field in general and also giving detailed information on SQ and EQ in assessing programming potential.

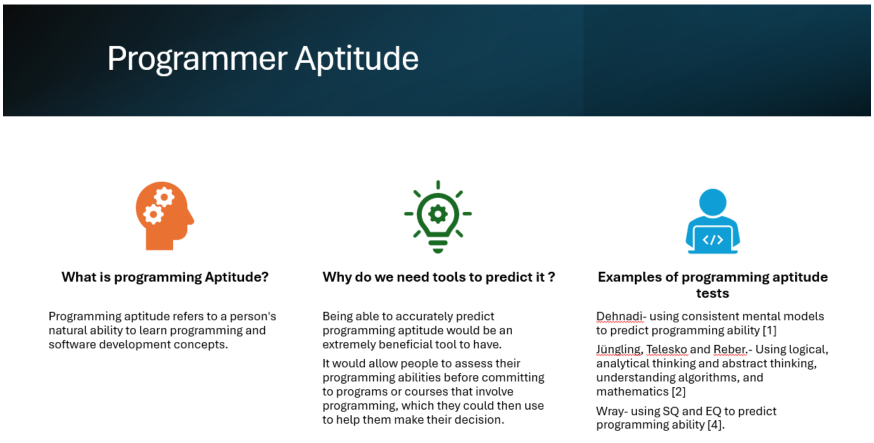
Presentation:

As part of the grading structure for the final year project, a presentation of the project was required. The goal of the presentation is to present the project to a supervisor, a second reader and your peers, detailing the scope of the project, the justifications and motivations and showcase all work done so far. There is also a five-minute question and answer session at the end of the project. The presentation followed the following structure:

- Explaining programming aptitude and why being able to measure it is beneficial.
- External motivations for project.
- Examples of studies programming aptitude from the literature review.
- Stuart Wray's paper on SQ and EQ as indicators for programming aptitude.
- Personal motivations for project.
- Goals for the project.
- Plan for the project.
- Potential challenges and blockers.

This component of the project was extremely beneficial as it provided an outside perspective on the project. The question-and-answer session offered insight into any issues overlooked so far in the project and allowed the second reader, John Noll, to give his opinion and feedback on the project. It was mentioned in the presentation about the desire to add a technical aspect to the project and John suggested using a statistical programming language, such as R, for the data analysis section of the project. This was something I had previously overlooked when examining ways to make the project more technical based, as a result the project plan was updated to incorporate this addition. Below are some examples of the slides from the presentation.

Examples of Presentation Slides:



Programmer Aptitude

What is programming Aptitude?

Programming aptitude refers to a person's natural ability to learn programming and software development concepts.

Why do we need tools to predict it ?

Being able to accurately predict programming aptitude would be an extremely beneficial tool to have.

It would allow people to assess their programming abilities before committing to programs or courses that involve programming, which they could then use to help them make their decision.

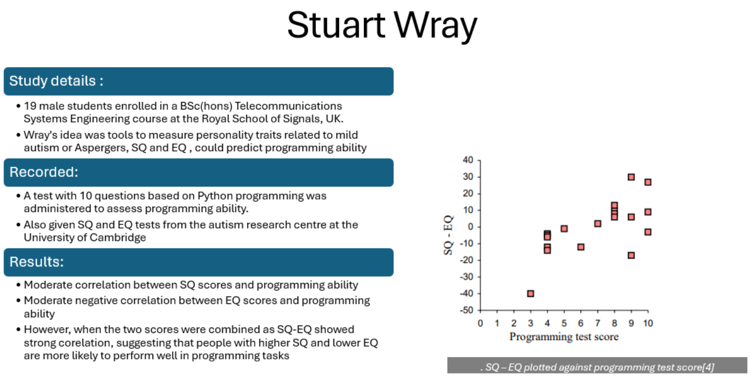
Examples of programming aptitude tests

Dehnadi- using consistent mental models to predict programming ability [1]

Jüngling, Telesko and Reber.- Using logical, analytical thinking and abstract thinking, understanding algorithms, and mathematics [2]

Wray- using SQ and EQ to predict programming ability [4].

Slide explaining programming aptitude and why being able to predict it is



Stuart Wray

Study details :

- 19 male students enrolled in a BSc(hons) Telecommunications Systems Engineering course at the Royal School of Signals, UK.
- Wray's idea was tools to measure personality traits related to mild autism or Aspergers, SQ and EQ , could predict programming ability

Recorded:

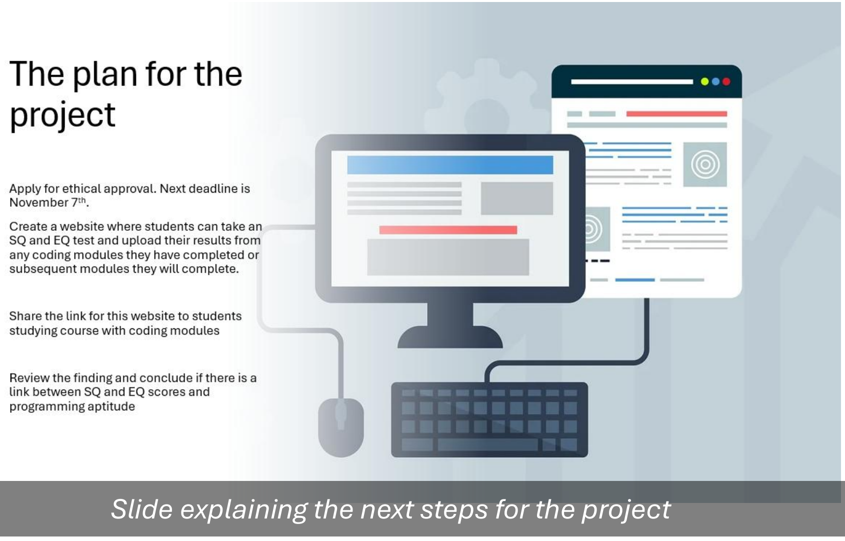
- A test with 10 questions based on Python programming was administered to assess programming ability.
- Also given SQ and EQ tests from the autism research centre at the University of Cambridge

Results:

- Moderate correlation between SQ scores and programming ability
- Moderate negative correlation between EQ scores and programming ability
- However, when the two scores were combined as SQ-EQ showed strong corelation, suggesting that people with higher SQ and lower EQ are more likely to perform well in programming tasks

. SQ – EQ plotted against programming test score[4]

Slide discussing Stuart Wray's paper on using SQ and EQ to measure



The plan for the project

Apply for ethical approval. Next deadline is November 7th.

Create a website where students can take an SQ and EQ test and upload their results from any coding modules they have completed or subsequent modules they will complete.

Share the link for this website to students studying course with coding modules

Review the finding and conclude if there is a link between SQ and EQ scores and programming aptitude

Slide explaining the next steps for the project

Design Survey:

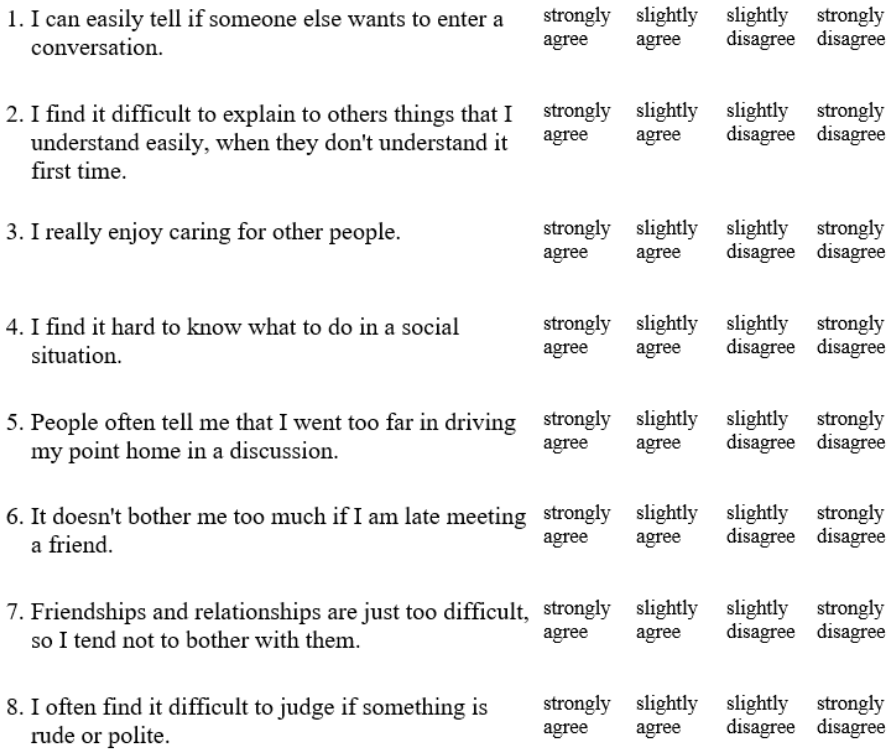
Designing the survey was possibly the most important part of the project. The survey determines the information and data we receive for analysis which will directly influence the outcome of the project as it will determine the accuracy of our results and the accuracy of SQ and EQ as determinants of programming aptitude. We want to create a survey that will capture all the information required while also following the University of Limerick's ethical guidelines. It was decided that a survey with three parts would be required to extract the desired information from students. The survey would get the following data from students:

- SQ score
- EQ score
- Previous programming results and personal data

Given that the project is based on Wray's study, we will use the same surveys he used to extract SQ and EQ scores from participants. Wray used test from the University of Cambridge to measure suspect SQ and EQ scores. Both tests follow the same structure. The participant is provided with a statement, and they circle one of the following in response to the statement.

- Strongly agree.
- Slightly agree.
- Slightly disagree.
- Strongly disagree.

Once participants have completed the questions, their answers can be used to calculate their SQ and EQ scores. Below are some sample questions.



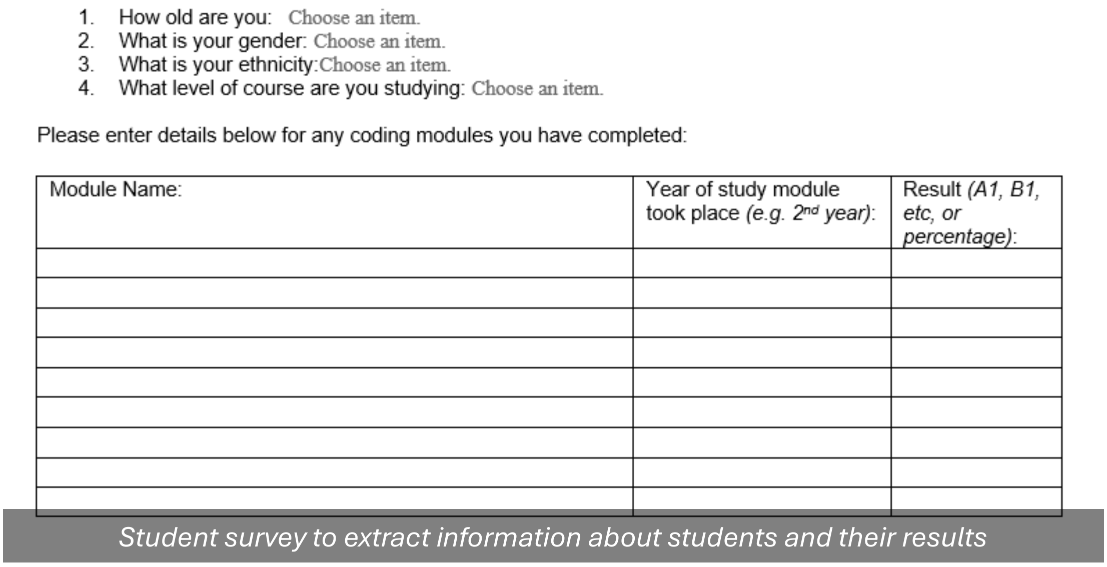
1. I can easily tell if someone else wants to enter a conversation.	strongly agree	slightly agree	slightly disagree	strongly disagree
2. I find it difficult to explain to others things that I understand easily, when they don't understand it first time.	strongly agree	slightly agree	slightly disagree	strongly disagree
3. I really enjoy caring for other people.	strongly agree	slightly agree	slightly disagree	strongly disagree
4. I find it hard to know what to do in a social situation.	strongly agree	slightly agree	slightly disagree	strongly disagree
5. People often tell me that I went too far in driving my point home in a discussion.	strongly agree	slightly agree	slightly disagree	strongly disagree
6. It doesn't bother me too much if I am late meeting a friend.	strongly agree	slightly agree	slightly disagree	strongly disagree
7. Friendships and relationships are just too difficult, so I tend not to bother with them.	strongly agree	slightly agree	slightly disagree	strongly disagree
8. I often find it difficult to judge if something is rude or polite.	strongly agree	slightly agree	slightly disagree	strongly disagree

Sample of questions from University of Cambridge's EQ questionnaire

(Autism Research Centre, University of Cambridge (n.d.))

For example, if one were to answer strongly disagree to the first statement it could suggest low empathy quotient as it shows a person is unable to pick up on behaviour ques from others when wanting to join a conversation. The test works by reviewing all answers to the statements and calculating the EQ score based on the subjects' responses.

For the third part of the survey the participants will need to answer is regarding their programming results and personal details about them that will also be useful for the study. The students will be asked for their age, gender, ethnicity and course of study and will then be asked to enter each programming module they have studied and the grade they received. When designing this part of the survey we needed to be conscious that we were following the ethical guidelines. Keeping the subjects anonymous by not asking for information that could potentially identify a participant. Here is the structure of the survey. Students are offered options for ethnicity and age ranges to preserve anonymity:



1. How old are you: Choose an item.
2. What is your gender: Choose an item.
3. What is your ethnicity:Choose an item.
4. What level of course are you studying: Choose an item.

Please enter details below for any coding modules you have completed:

Module Name:	Year of study module took place *(e.g. 2nd year)*:	Result *(A1, B1, etc, or percentage)*:

Student survey to extract information about students and their results

Once we were happy with the survey design it was implemented using Microsoft forms allowing for an easy way to distribute the survey and access the results.

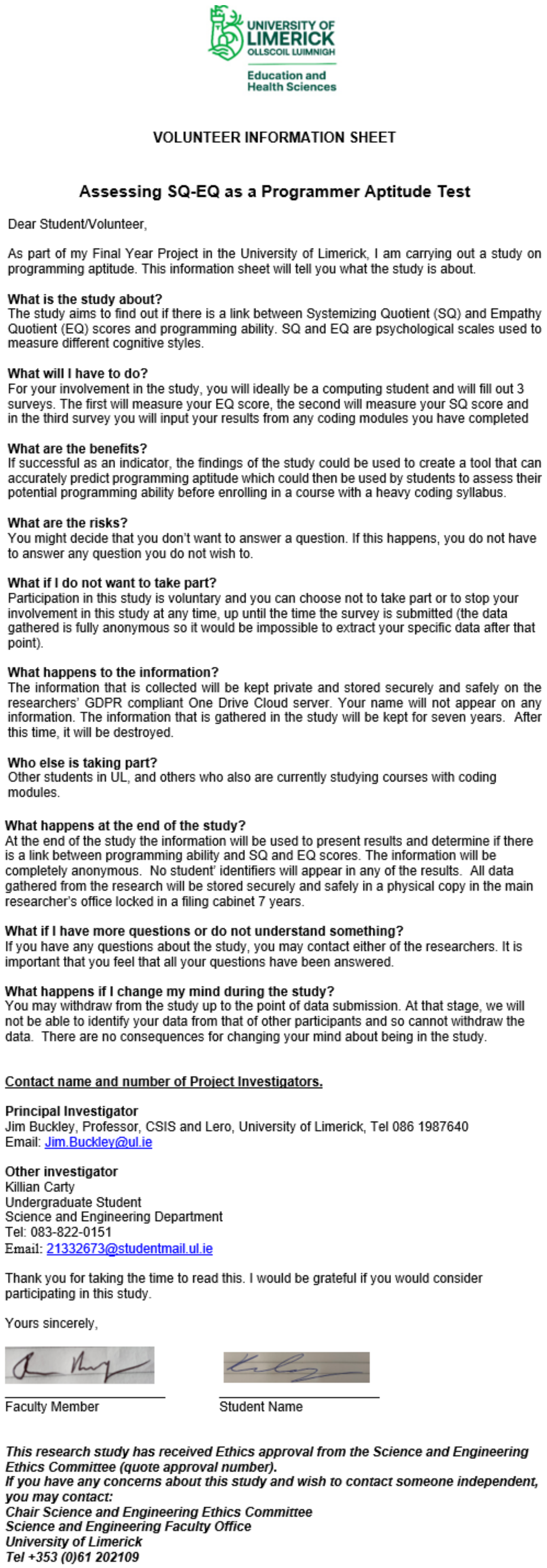
Ethical Approval:

Given that the project contained a survey using human participants as candidates, ethical approval from the University of Limerick's science and engineering research committee was required to conduct the survey. After reviewing the science and engineering research ethics webpage it was identified that the following was needed to apply for ethical approval:

- Application form.
- Information sheet.
- Consent form.
- List of survey questions.
- Research privacy notice form.

(University of Limerick, Research Ethics (n.d.))

The application form consists of a number of questions about the project in general and the survey being conducted. This form is to give the ethics committee an overall understanding of the project and its purpose and the intended method for performing the survey. A list of all survey questions was also required to ensure all questions asked followed ethical guidelines. The research privacy notice form outlines how the university will manage and protect the data collected throughout the course of the project, complying with GDPR and other data protection acts. Once these forms were filled out the consent form and information sheet had to be designed to inform participants. The information sheet is used to inform participants about the research study in clear and concise language, while the consent form is needed to document participants agreement to take part in the study. Below are the information sheet and consent forms that will be provided to students.



UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH
Education and Health Sciences

VOLUNTEER INFORMATION SHEET

Assessing SQ-EQ as a Programmer Aptitude Test

Dear Student/Volunteer,

As part of my Final Year Project in the University of Limerick, I am carrying out a study on programming aptitude. This information sheet will tell you what the study is about.

What is the study about?
The study aims to find out if there is a link between Systemizing Quotient (SQ) and Empathy Quotient (EQ) scores and programming ability. SQ and EQ are psychological scales used to measure different cognitive styles.

What will I have to do?
For your involvement in the study, you will ideally be a computing student and will fill out 3 surveys. The first will measure your EQ score, the second will measure your SQ score and in the third survey you will input your results from any coding modules you have completed

What are the benefits?
If successful as an indicator, the findings of the study could be used to create a tool that can accurately predict programming aptitude which could then be used by students to assess their potential programming ability before enrolling in a course with a heavy coding syllabus.

What are the risks?
You might decide that you don't want to answer a question. If this happens, you do not have to answer any question you do not wish to.

What if I do not want to take part?
Participation in this study is voluntary and you can choose not to take part or to stop your involvement in this study at any time, up until the time the survey is submitted (the data gathered is fully anonymous so it would be impossible to extract your specific data after that point).

What happens to the information?
The information that is collected will be kept private and stored securely and safely on the researchers' GDPR compliant One Drive Cloud server. Your name will not appear on any information. The information that is gathered in the study will be kept for seven years. After this time, it will be destroyed.

Who else is taking part?
Other students in UL, and others who also are currently studying courses with coding modules.

What happens at the end of the study?
At the end of the study the information will be used to present results and determine if there is a link between programming ability and SQ and EQ scores. The information will be completely anonymous. No student' identifiers will appear in any of the results. All data gathered from the research will be stored securely and safely in a physical copy in the main researcher's office locked in a filing cabinet 7 years.

What if I have more questions or do not understand something?
If you have any questions about the study, you may contact either of the researchers. It is important that you feel that all your questions have been answered.

What happens if I change my mind during the study?
You may withdraw from the study up to the point of data submission. At that stage, we will not be able to identify your data from that of other participants and so cannot withdraw the data. There are no consequences for changing your mind about being in the study.

Contact name and number of Project Investigators.

Principal Investigator
Jim Buckley, Professor, CSIS and Lero, University of Limerick, Tel 086 1987640
Email: Jim.Buckley@ul.ie

Other investigator
Killian Carty
Undergraduate Student
Science and Engineering Department
Tel: 083-822-0151
Email: 21332673@studentmail.ul.ie

Thank you for taking the time to read this. I would be grateful if you would consider participating in this study.

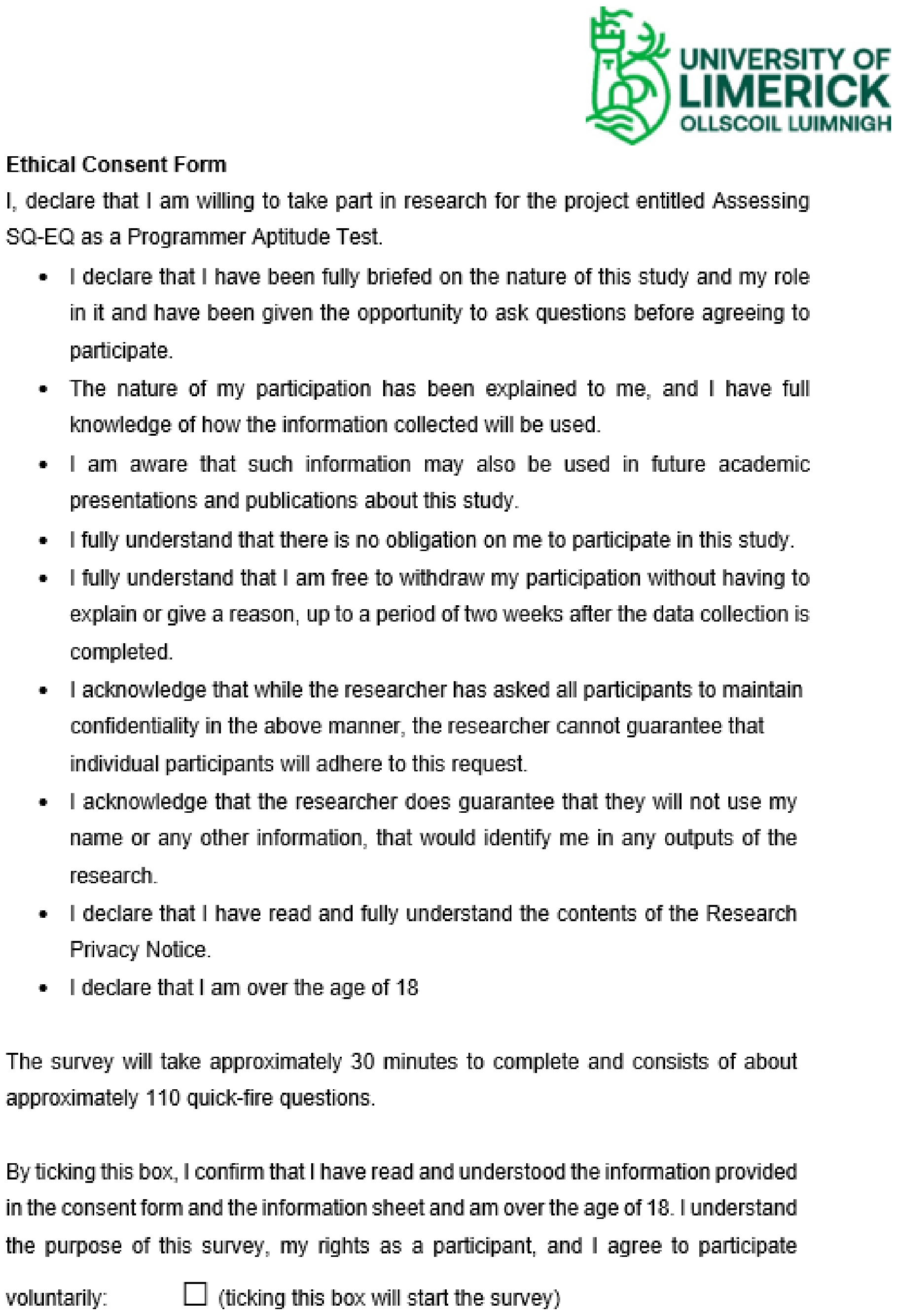
Yours sincerely,

Faculty Member

Student Name

This research study has received Ethics approval from the Science and Engineering Ethics Committee (quote approval number).
If you have any concerns about this study and wish to contact someone independent, you may contact:
Chair Science and Engineering Ethics Committee
Science and Engineering Faculty Office
University of Limerick
Tel +353 (0)61 202109

Survey information sheet



Ethical Consent Form

I, declare that I am willing to take part in research for the project entitled Assessing SQ-EQ as a Programmer Aptitude Test.

- I declare that I have been fully briefed on the nature of this study and my role in it and have been given the opportunity to ask questions before agreeing to participate.
- The nature of my participation has been explained to me, and I have full knowledge of how the information collected will be used.
- I am aware that such information may also be used in future academic presentations and publications about this study.
- I fully understand that there is no obligation on me to participate in this study.
- I fully understand that I am free to withdraw my participation without having to explain or give a reason, up to a period of two weeks after the data collection is completed.
- I acknowledge that while the researcher has asked all participants to maintain confidentiality in the above manner, the researcher cannot guarantee that individual participants will adhere to this request.
- I acknowledge that the researcher does guarantee that they will not use my name or any other information, that would identify me in any outputs of the research.
- I declare that I have read and fully understand the contents of the Research Privacy Notice.
- I declare that I am over the age of 18

The survey will take approximately 30 minutes to complete and consists of about approximately 110 quick-fire questions.

By ticking this box, I confirm that I have read and understood the information provided in the consent form and the information sheet and am over the age of 18. I understand the purpose of this survey, my rights as a participant, and I agree to participate voluntarily: □ (ticking this box will start the survey)

Survey consent form

After all necessary documents were created and all required forms were filled out ethics approval was applied for. The project experienced some delays due to the ethical approval process, which required several rounds of communication with the ethics committee. This involved addressing feedback and changes requested by the committee and clarifying aspects of the study. Once all requested changed and updates were made ethical approval was granted on February 21st.

Interim Report:

The interim report is another part of the grading structure for the final year project. It is a report detailing the work done to date. The report includes the following chapter:

- Introduction
- Background/Literature Review
- Methodology
- Conclusion
- Appendices

The overall goal of the interim report is similar to the presentation aspect. It showcases the work done on the project. It offers an outside perspective on the project offering feedback for potential improvements. The interim report was then updated and modified to produce the final report.

Conduct Survey:

Once the survey was ready and we had ethical approval, we began conducting the survey by sending the survey link out to lecturers who taught modules with a heavy coding syllabus. I also reached out to my fellow students in my computer systems courses asking them to participate in a survey for my final year project. However, to adhere to ethics guidelines it was made very clear that the survey was not mandatory, and the students were under no obligation to take part. The survey was collecting data and responses for approximately 3 weeks.

Data Analysis:

Data analysis began, after the survey had been completed. The main goal of this component was to evaluate if there is a link between SQ and EQ scores, and programming aptitude using the data collected from the survey. The survey received 14 responses. Ideally, we would have liked a larger sample size, but we could still perform analysis on the data we had.

1. Data preparation: The data collected from the study was examined, removing any incomplete or invalid responses. We also ensured that all the data is the same type. For example, in the programming results survey students could input their programming results in percentage format or as the grade they received (A1, A2, B1 etc..), we needed to ensure all the data was the same so it could be properly analysed.
2. Data Analysis: Once the data had been cleaned and compiled into a single dataset, we started analysing it. We began by generating statistics such as the mean, median and standard deviation for the SQ score, EQ score and programming results. We then examined if there is any correlation between SQ scores, EQ scores, and programming aptitude. Regression analysis was then carried out on the data to determine the extent SQ and EQ could be used to predict programming ability. All this was done using the statistical programming language R.
3. Summarise and create visuals: To finish the data analysis process the findings from the study were summarised along with graphs and visualizations, created using python, to display the findings which were added to the final project report.

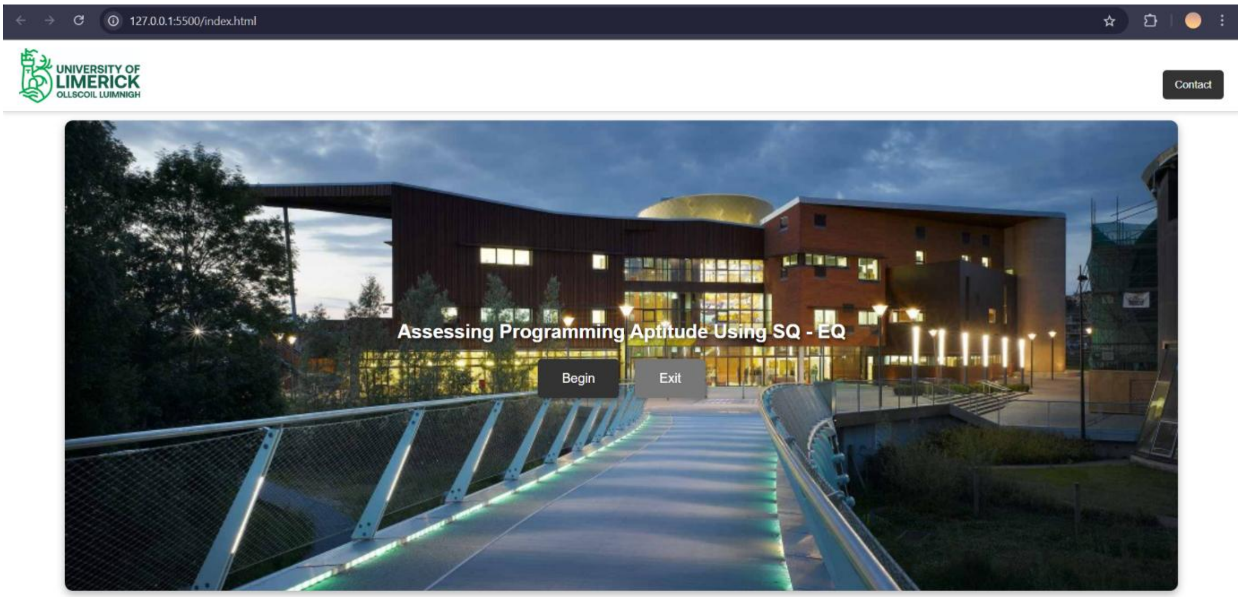
There is an in-depth data analysis later in this report.

Website Development:

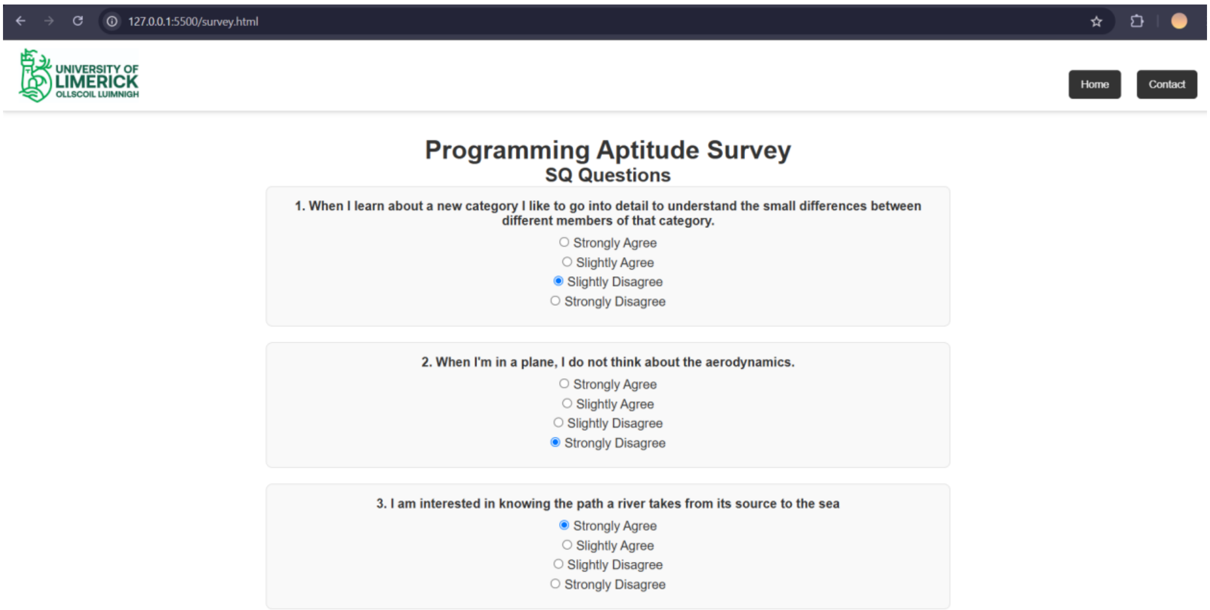
Following the completion of the data analysis I designed a simple web tool, using HTML, CSS and JavaScript, which used the findings of the study to predict a person's average programming grade based on their SQ and EQ score. The website allowed the user to complete the same 10 EQ and 10 SQ questions the survey participants completed and receive an estimated programming aptitude score based on the regression model derived during data analysis. The website uses the following formula to calculate programming aptitude:

```
const aptitude = Math.round(0.786 * diff + 68.38);
```

The website provides a user-friendly interactive way for individuals to assess their programming aptitude based on their cognitive quotients. Although the predictive power of the model was not statistically significant the tool provides an interactive component to the project and showcases how a model could be applied if future research results in statistically significant results. Below are some screenshots from the web application.


127.0.0.1:5500/index.html
UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH
Contact
Assessing Programming Aptitude Using SQ - EQ
Begin
Exit

Landing page of website



Programming Aptitude Survey

SQ Questions

1. When I learn about a new category I like to go into detail to understand the small differences between different members of that category.

○ Strongly Agree
○ Slightly Agree
● Slightly Disagree
○ Strongly Disagree

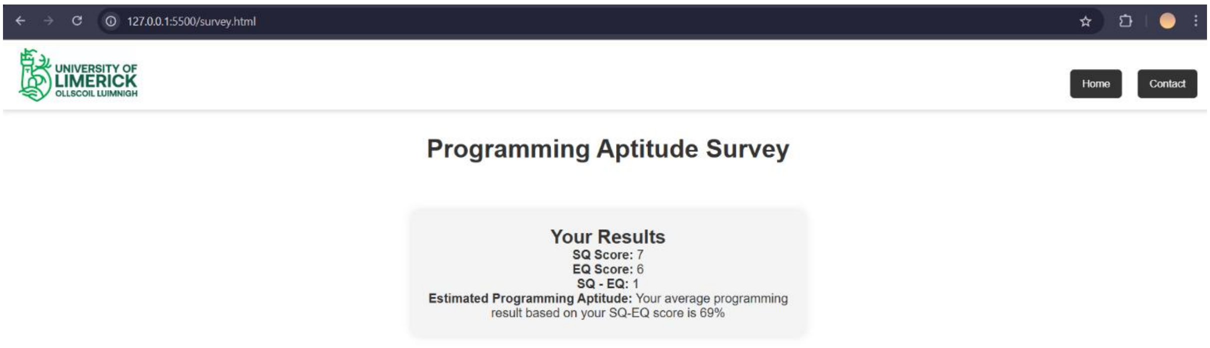
2. When I'm in a plane, I do not think about the aerodynamics.

○ Strongly Agree
○ Slightly Agree
○ Slightly Disagree
● Strongly Disagree

3. I am interested in knowing the path a river takes from its source to the sea

● Strongly Agree
○ Slightly Agree
○ Slightly Disagree
○ Strongly Disagree

SQ question page of website



Programming Aptitude Survey

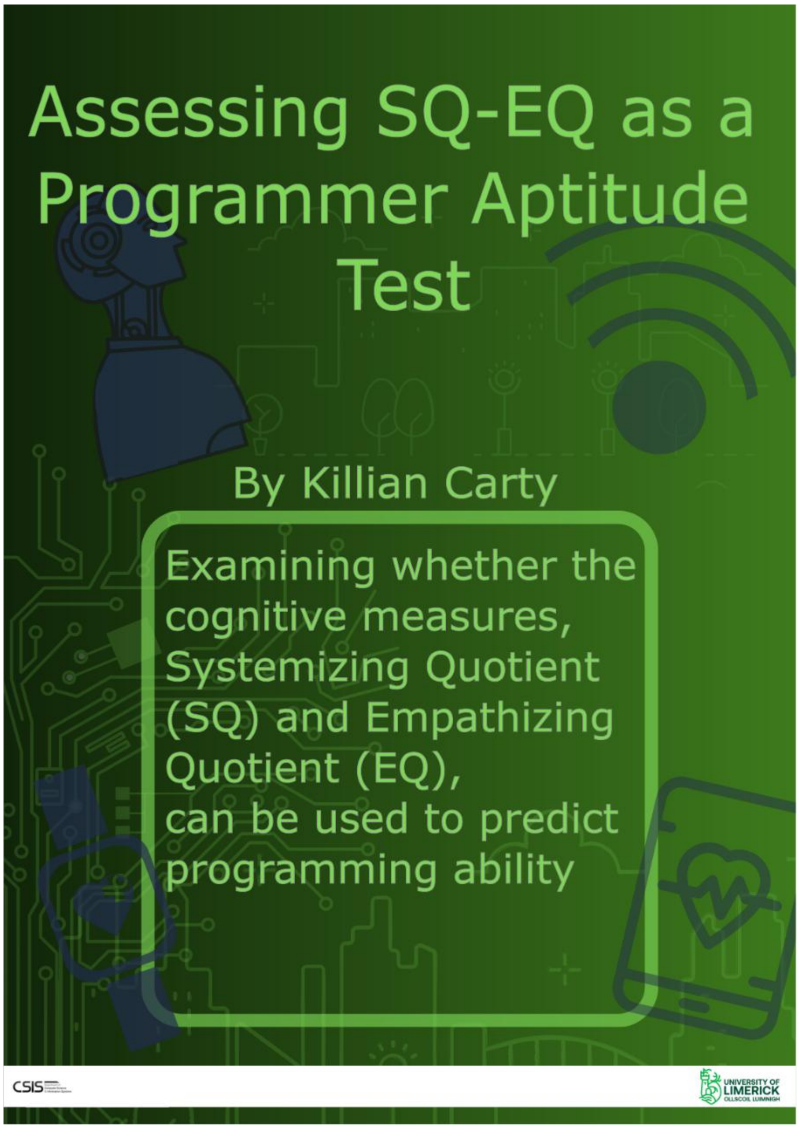
Your Results

SQ Score: 7
EQ Score: 6
SQ - EQ: 1
Estimated Programming Aptitude: Your average programming result based on your SQ-EQ score is 69%

Results page of website

Demo Day Poster and Video:

The demo day poster and video are other aspects of the grading structure for the final year project. A short video presentation between 2 and 3 minutes long was produced to give an overview of the project along with a poster. During Demo Day I presented my project to any individuals who were interested. I explained the motivation behind the project and how it was completed. People could also use the website I had developed to test their own programming aptitude based on the findings of our project. Below is the demo day poster.



Assessing SQ-EQ as a Programmer Aptitude Test

By Killian Carty

Examining whether the cognitive measures, Systemizing Quotient (SQ) and Empathizing Quotient (EQ), can be used to predict programming ability

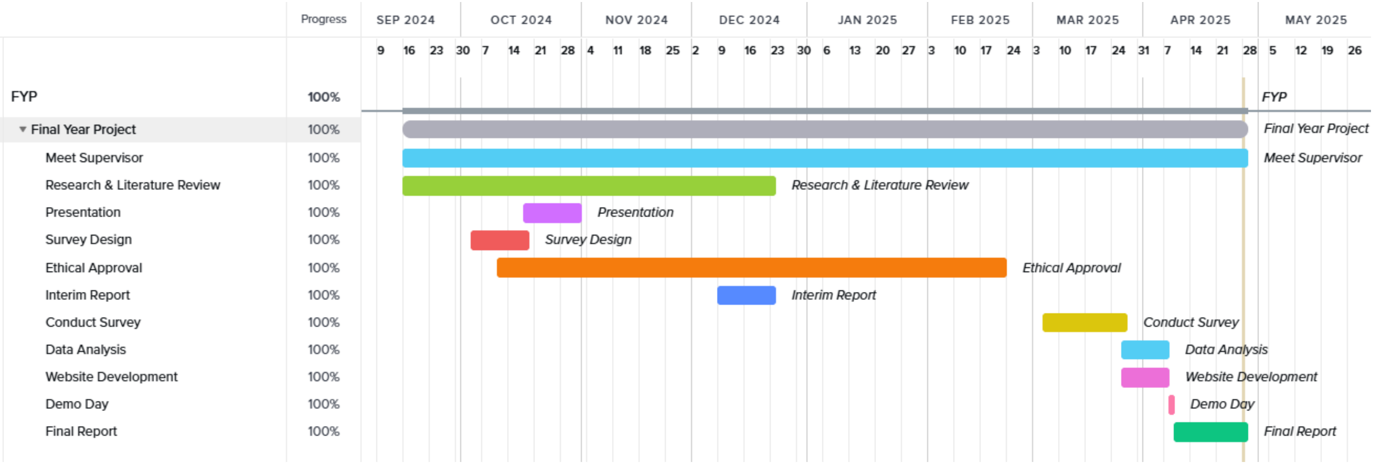
Demo day poster

Final Report:

The final component of the project was the final report. This was a detailed report containing all aspects of the project. The report built on the interim report and outline the projects scope and purpose, methodology and described the findings and results from the finished project.

Gantt Chart for the project:

Below is a Gantt chart showing the complete timeline of the project from its beginning on September 16th 2024 till its completion on the 28th of April 2025.



	Progress
FYP	100%
▾ Final Year Project	100%
Meet Supervisor	100%
Research & Literature Review	100%
Presentation	100%
Survey Design	100%
Ethical Approval	100%
Interim Report	100%
Conduct Survey	100%
Data Analysis	100%
Website Development	100%
Demo Day	100%
Final Report	100%

Data Analysis:

This section of the report will discuss the analysis of the data obtained from the study. It will discuss how the data was prepared and cleaned for analysis and what methods were used to perform the data analysis. This section also examines the results and conclusions we can draw from the data gathered from the study, as well as answering the question can SQ and EQ be used to predict programming aptitude.

Data Preparation and cleaning:

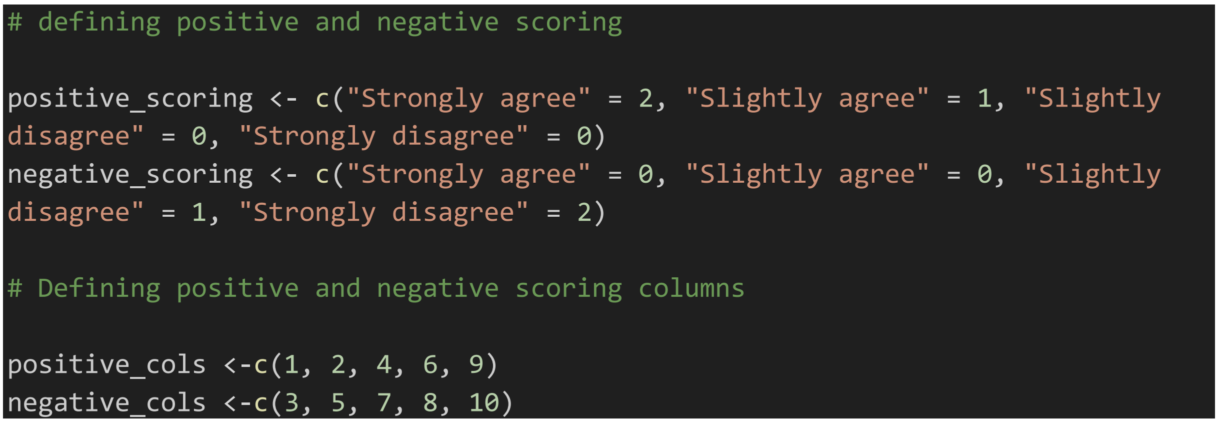
Given that the survey was conducted using Microsoft forms the resulting raw data was provided in the form of an excel spreadsheet. The dataset consisted of a sample size of 14 participants. While a larger sample size would have been preferred, analysis can still be conducted to explore any patterns or trends in the available data. Before beginning statistical analysis, this raw data collected during the study needed to be cleaned and prepared to ensure the data was in a suitable state for the analysis phase of the project.

The first thing that was done to the data was all unnecessary columns and rows were removed. This included columns such as "start time" and "completion time" as well as other columns added by Microsoft forms that were not required for the data analysis. Once the unnecessary columns were removed the data consisted of 46 columns and separated into 4 separate sections:

- Demographic information
- Programming results
- Systemizing Quotient (SQ) responses
- Empathy Quotient (EQ) responses

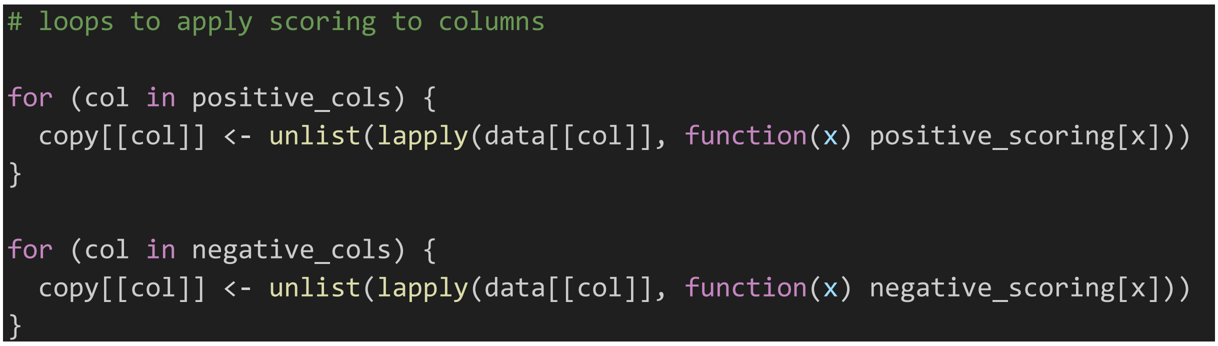
The next step was calculating the participants SQ and EQ scores from their answers to the SQ and EQ questionnaires. The SQ and EQ responses were scored using the rules provided with the questionnaire by the Cambridge University Autism Research Centre (Autism Research Centre, University of Cambridge (n.d.)). The scoring system worked by giving a score between 0 and 2 for each of the 10 questions based on what the participants response was and what question they were answering. For SQ the scoring was "Score 2 points for "definitely agree", 1 point for "slightly agree", and 0 points for "slightly disagree" and "strongly disagree" for items 1, 3, 4, 6, 7, 9 and 10. Score 2 points for "strongly disagree", 1 point for "slightly disagree", and 0 points for "slightly agree" and "strongly agree" for items 2, 5, and 8." (Autism Research Centre, University of Cambridge (n.d.)). The EQ scoring was similar " Score 2 points for "strongly agree", 1 point for "slightly agree", and 0 points for "slightly disagree" and "strongly disagree" for items 1, 2, 4, 6, and 9. Score 2 points for "strongly disagree", 1 point for "slightly disagree", and 0 points for "slightly agree" and "strongly agree" for items 3, 5, 7, 8 and 10." (Autism Research Centre, University of Cambridge (n.d.)). Using these scoring rules two R scripts were developed to automate this scoring process for both SQ and EQ. Its input was an excel sheet that contained the data for the Quotient it was calculating, either SQ or EQ, and it returned the participants respective scores. Below is a series of code snippets from the R script that calculated EQ score:

The positive and negative scoring and columns are defined



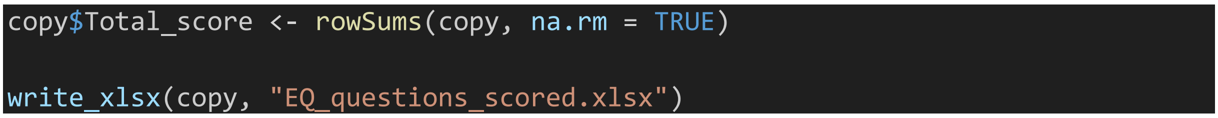
```
# defining positive and negative scoring

positive_scoring <- c("Strongly agree" = 2, "Slightly agree" = 1, "Slightly
disagree" = 0, "Strongly disagree" = 0)
negative_scoring <- c("Strongly agree" = 0, "Slightly agree" = 0, "Slightly
disagree" = 1, "Strongly disagree" = 2)

# Defining positive and negative scoring columns

positive_cols <-c(1, 2, 4, 6, 9)
negative_cols <-c(3, 5, 7, 8, 10)
```

We then loop through the positive and negative columns applying the respective scoring



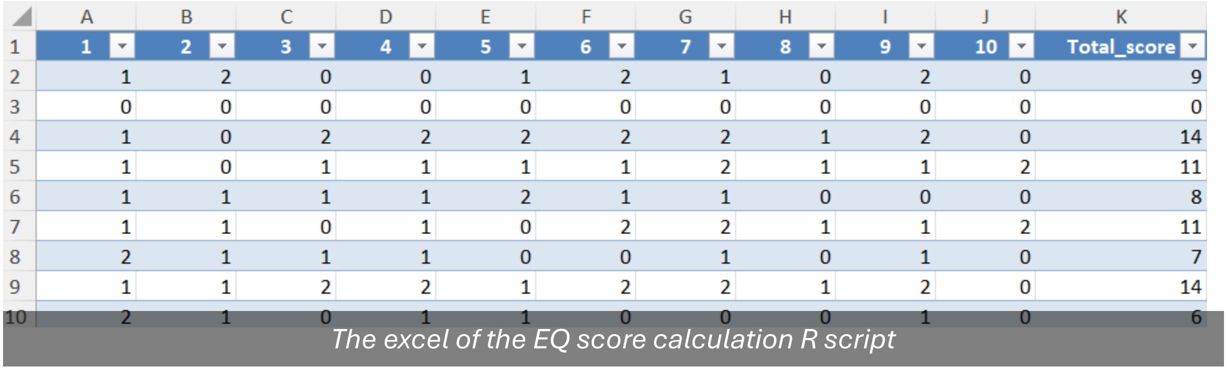
```
# loops to apply scoring to columns

for (col in positive_cols) {
  copy[[col]] <- unlist(lapply(data[[col]], function(x) positive_scoring[x]))
}

for (col in negative_cols) {
  copy[[col]] <- unlist(lapply(data[[col]], function(x) negative_scoring[x]))
}
```

We then create new column to hold the score and sum up all the scores and generate an excel file containing each participant's score for each question and their total score



```
copy$Total_score <- rowSums(copy, na.rm = TRUE)

write_xlsx(copy, "EQ_questions_scored.xlsx")
```

The output of the R scripts was an excel file where each answer to a question was changed to its score value and a new column containing the total score was added as shown below:



	A	B	C	D	E	F	G	H	I	J	K
1	1	2	3	4	5	6	7	8	9	10	Total_score
2	1	2	0	0	1	2	1	0	2	0	9
3	0	0	0	0	0	0	0	0	0	0	0
4	1	0	2	2	2	2	2	1	2	0	14
5	1	0	1	1	1	1	2	1	1	2	11
6	1	1	1	1	2	1	1	0	0	0	8
7	1	1	0	1	0	2	2	1	1	2	11
8	2	1	1	1	0	0	1	0	1	0	7
9	1	1	2	2	1	2	2	1	2	0	14
10	2	1	0	1	1	0	0	0	1	0	6

The excel of the EQ score calculation R script

The total score column from the SQ and EQ score calculation scripts were added to the original data excel sheet and a simple excel formula was used to generate the SQ minus EQ column. We now had the participants SQ, EQ and EQ-EQ scores.

With the SQ and EQ scoring completed we now needed to prepare the programming results for data analysis. This section of the dataset was very inconsistent as participants could enter their results in both percentage and letter grade format. To ensure consistency and usability all grades needed to be converted into percentage format using the computer science grading system:

GRADE	PERCENTAGE
A1	80%
A2	72%
B1	64%
B2	60%
B3	56%
C1	52%
C2	48%
C3	40%
D1	35%
D2	30%
F	20%

System used to convert letter grades to percentages

Using this grading system an R script was developed to convert all letter grades into percentage format and then calculate an average grade for each participant. The average grade would then be used as the value to examine any potential links between SQ and EQ and programming aptitude. Below are some snippets from the grade calculation R script:

Defining the grading system we will use to convert the letter grades to numbers

```
#Defining grade bands and columns

grade_band <- c(
  "A1" = 80,
  "A2" = 72,
  "B1" = 64,
  "B2" = 60,
  "B3" = 56,
  "C1" = 52,
  "C2" = 48,
  "C3" = 40,
  "D1" = 35,
  "D2" = 30,
  "F" = 20
)
```

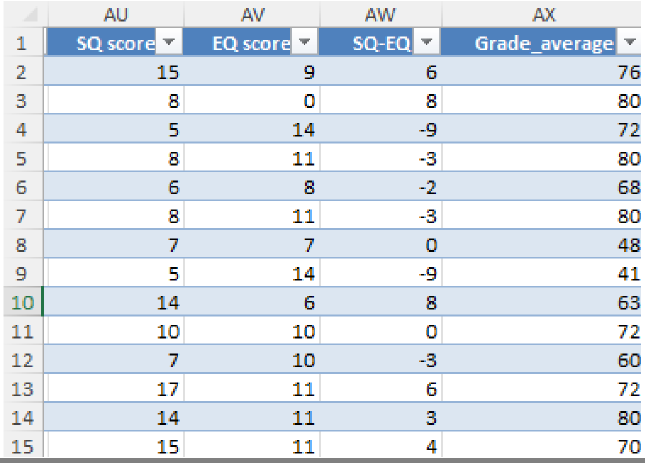
Defining the function to convert the grades and looping through grade columns applying function

```r
# Function to handle grade conversion logic

convert_grade <- function(x){
  x <- trimws(toupper(as.character(x)))
  if (grepl("%", x)) {
    return(as.numeric((gsub("%", "", x))))
  } else if (x %in% names(grade_band)){
    return(grade_band[x])
  }else {
    return(NA)
  }
}

# Looping through each column and converting grades

for (col in grade_columns) {
  copy[[col]] <- sapply(copy[[col]], convert_grade)
}
```

Creating a new column for the participants average grade and saving the results to an excel file

```r
# Creating new column with average grade score

copy$Grade_average <- rowMeans(copy[, grade_columns], na.rm = TRUE)

write_xlsx(copy, "survey_data_cleaned_and_prep.xlsx")
```

After the unnecessary column deletion, SQ and EQ scoring, the grade conversion and grade average calculation the data was fully cleaned and prepared for data analysis. After all data preparation, we now had the original dataset cleaned and prepared with the following columns added:



	AU	AV	AW	AX
1	SQ score	EQ score	SQ-EQ	Grade_average
2	15	9	6	76
3	8	0	8	80
4	5	14	-9	72
5	8	11	-3	80
6	6	8	-2	68
7	8	11	-3	80
8	7	7	0	48
9	5	14	-9	41
10	14	6	8	63
11	10	10	0	72
12	7	10	-3	60
13	17	11	6	72
14	14	11	3	80
15	15	11	4	70

The new columns added to the cleaned dataset

Analysis:

Once the data had been cleaned and prepared, we could begin the data analysis to explore any potential relationships between the participants SQ, EQ and programming results. The first thing we did was examine each attribute individually to get a general understanding about the dataset and each attribute before we began further statistical analysis. All graphs and visualisations used in this section were created using python.

SQ Analysis:

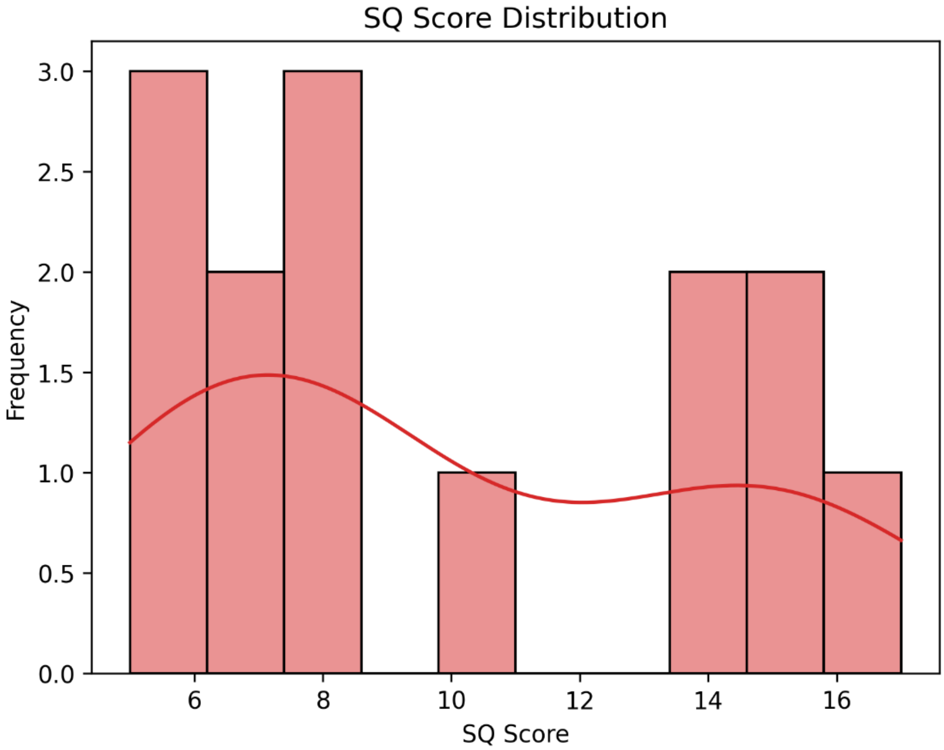
The first attribute we examined was SQ. Upon examining the SQ we found there was a split in the data forming two distinct clusters:

- One group with low SQ scores ranging between 5 and 9.
- A second group with higher SQ scores between 13 and 17.

The SQ data had the following values:

- Minimum score: 5.
- Maximum score: 17.
- Mean score: 9.93.

The SQ scoring ranges from 0-20 so having a mean close to the mid-point and maximum and minimum values close to the maximum and minimum range of the SQ score range means we have a nice balanced spread across the full SQ scale in the data. The graph below supports these observations with a bimodal distribution peaking in both the higher and lower ranges. The split in the data suggests that participants were either highly systemizing or lowly systemizing individuals with very few subjects in the middle with moderate systemizing scores.



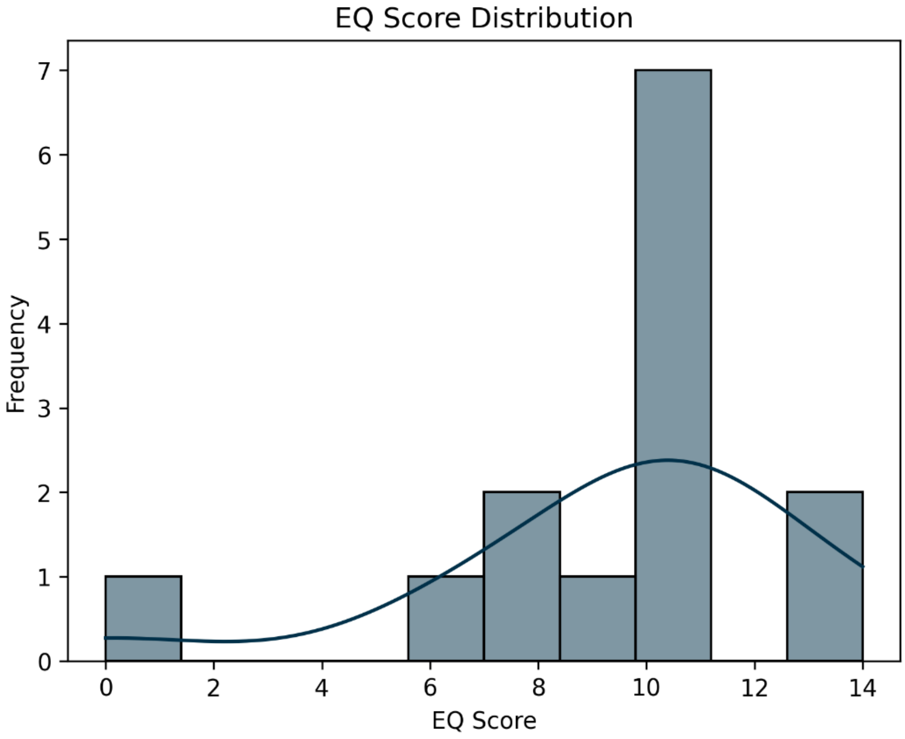
Graph showing the SQ distribution among the participants

EQ Analysis:

After the SQ analysis we examined the EQ scores in further detail. In contrast to the SQ scores the EQ scores did not show 2 distinct groups but instead a distribution that was left skewed with a clear concentration of scores towards the higher end of the scale. This was an interesting observation as the original hypothesis assumes that programmers traditionally have a lower empathizing quotient. The EQ analysis produced the following values:

- Minimum score: 0.
- Maximum score: 14.
- Mean score: 9.5.

Similarly to SQ we had a mean around the midpoint of the SQ scoring range (0-20). In general, the EQ scores tended to cluster around the midpoint with half of the participants having and EQ score of 10 this indicates moderate levels of empathizing quotient among the participants. Another interesting observation was one of the participants scored a 0, providing some variability to the data that was primarily dominated by scores within the 7-10 range. Overall, the EQ data suggests the majority of the participants have moderate empathizing quotients. The graph below supports these observations, showing a clear distinct peak around score 10.



Graph showing the EQ distribution among the participants

SQ-EQ Analysis:

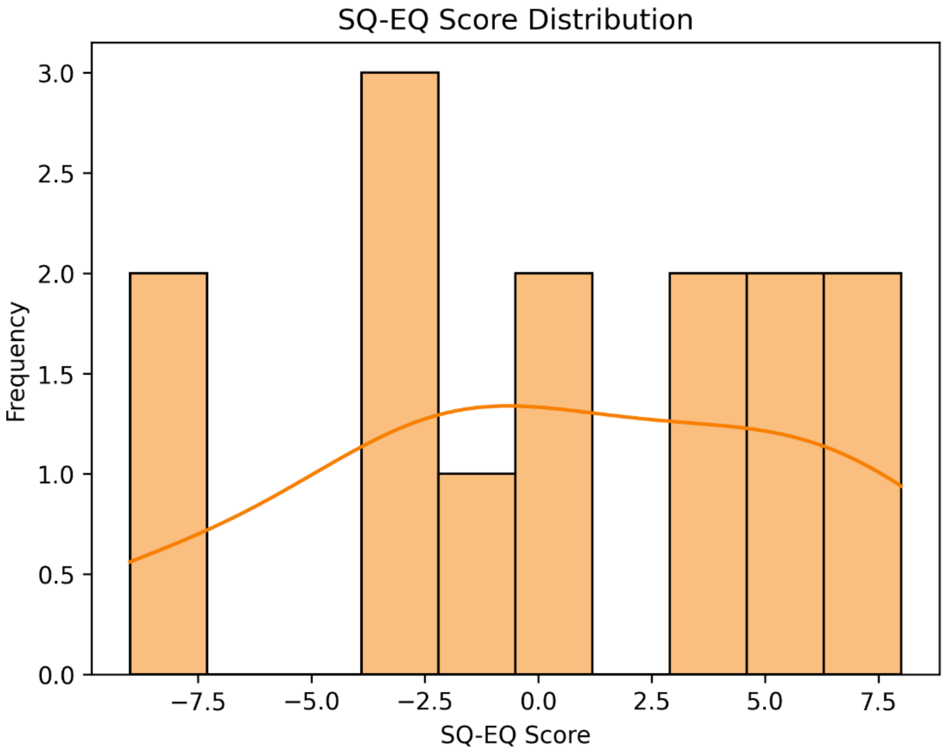
We then examined the SQ-EQ data. After examining both the quotients individually we then examined the combination of the two in the form of SQ-EQ. The statistical analysis returns these values for the SQ- EQ scores:

Minimum: -9

Maximum: 8

Mean: 0.43

The statistical values and the graph below show a fairly even spread across the SQ-EQ scores. We can see that participants leaned towards a slightly higher systemizing scores with a positive mean of 0.43. The range of -9 to 8 suggests diversity among the population. The Mean being close to 0 suggests the group as a whole did not favour one cognitive style over the other.



Graph showing the SQ-EQ distribution among the participants

Programming Results Analysis:

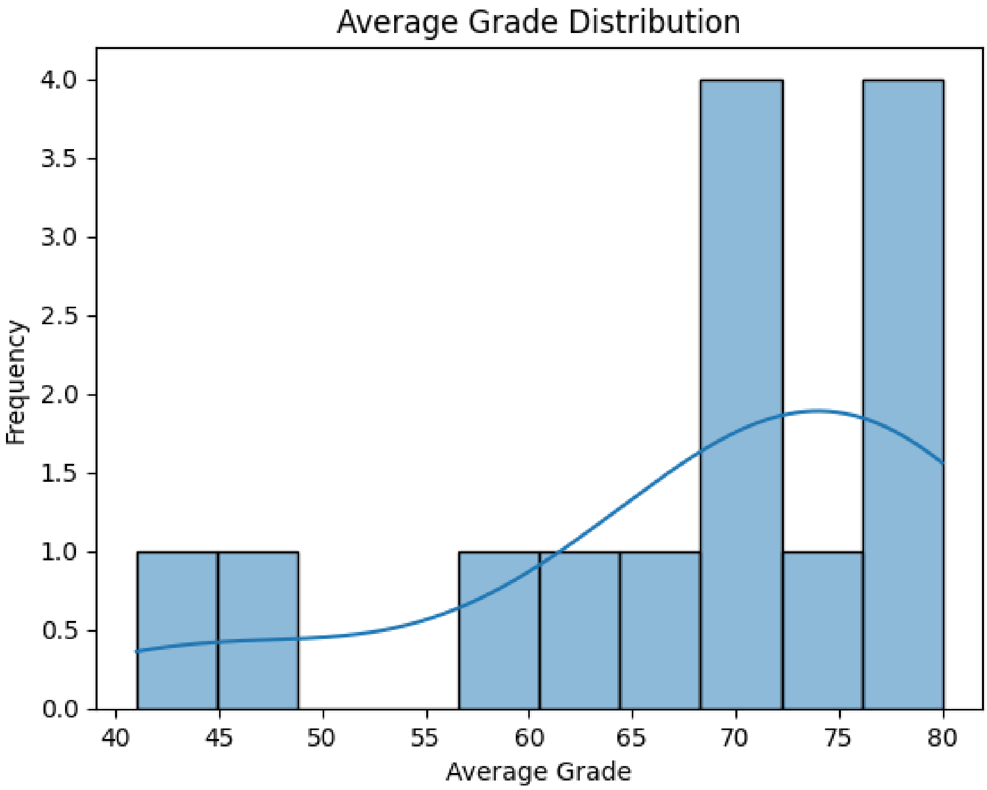
Finally, we examined the programming results. As stated in the data preparation section it was decided to get the average grade score for each participant and use that as the measure to analyse programming performance. The average grade data has the following statistics:

Minimum: 41

Maximum: 80

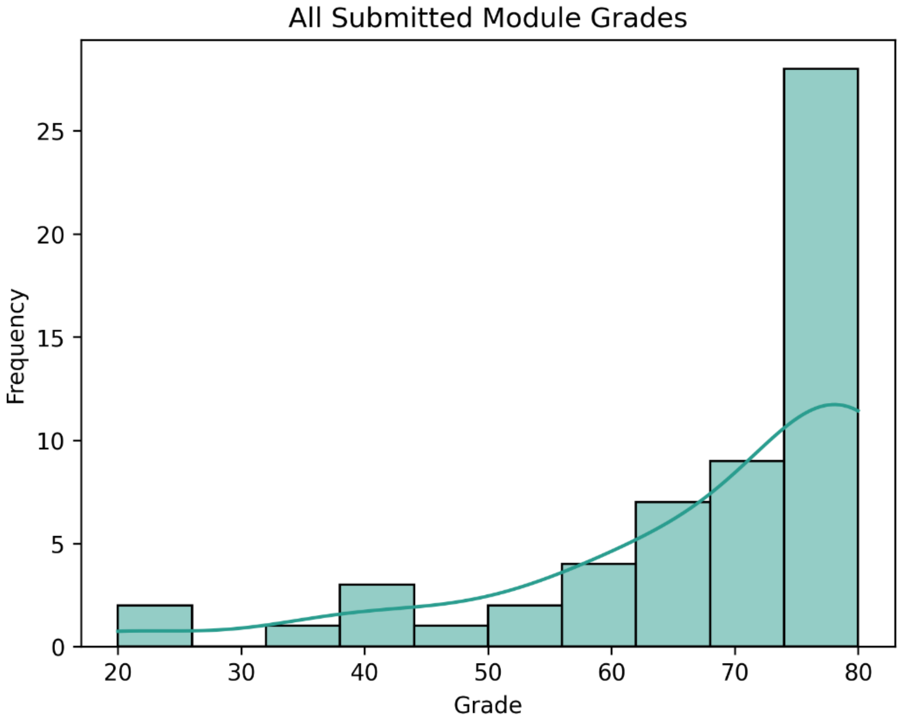
Mean: 68.71

We can see from these statistics and the graph below that there is a clear left skewed distribution in the data with participants tending to have a higher programming average. More than half the participants having an average of 70% or above, indicating a high performing population.



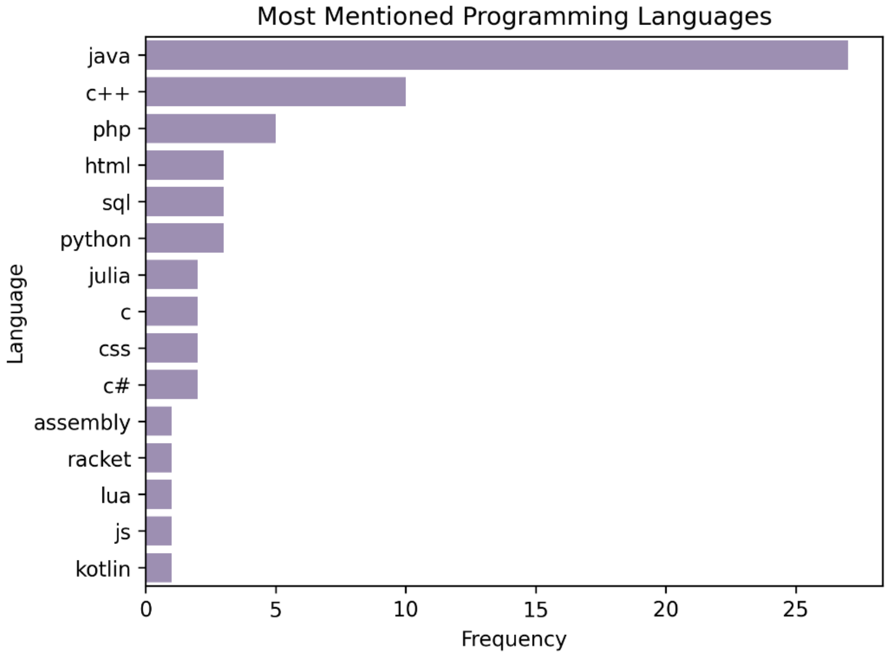
Graph showing the average grade distribution among the participants

When we examine all grades that were submitted not just the average this trend becomes even clearer with a significant number of grades submitted being above 60% and the most frequent grade being 80%. This trend could be explained by the self-selection aspect of the survey as participants were allowed to select what grades they uploaded. This means many participants may have sub consciously or consciously selected modules they had performed well in when submitting their grades to the survey. While this is not the case for all participants, as some participants submitted modules they had performed worse in with one student even submitting a module they had failed, there is a clear trend of higher grades being submitted. This could be something that is examined and addressed in a future study. However, we cannot say for sure what has caused this pattern in the data and the trend could just suggest that the participants who took part in the survey were generally more proficient programmers with better academic performance.



Graph showing the SQ distribution among the participants

The data also revealed what programming languages the participants were being examined in for each module. The graph below shows that Java was clearly the dominant programming language, something that was expected given that Java would be the main language taught to computer science students in the University of limerick, where the survey was conducted. These results also show a nice variation in the programming languages examined with 15 different languages being mentioned in the survey.



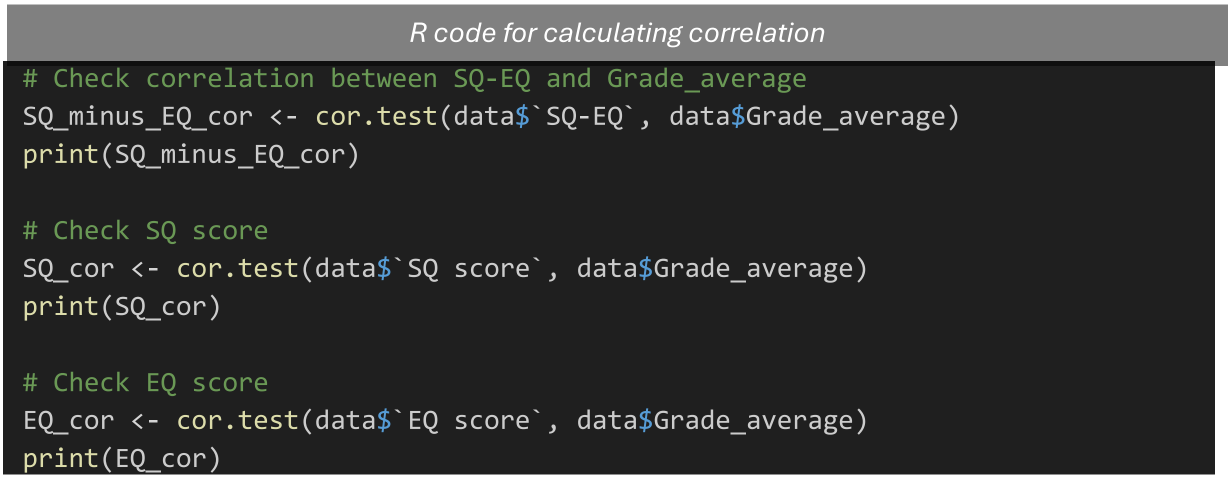
Graph showing the programming languages mentioned in the survey

Correlation Analysis:

After examining each attribute individually, we explored whether there was any relationship between SQ, EQ, SQ-EQ and programming grades.

To assess the strength of the relationships between the cognitive quotients and the programming performance, we used the Pearson correlation coefficient. The correlation test was conducted using a simple R script as the language already had the function built in:

R code for calculating correlation



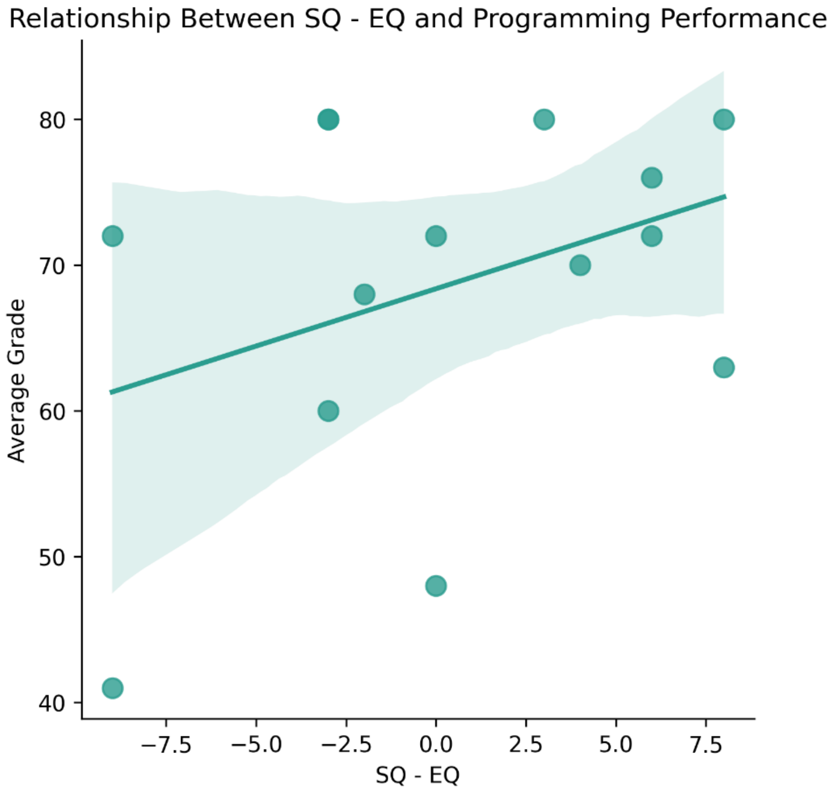
```
# Check correlation between SQ-EQ and Grade_average
SQ_minus_EQ_cor <- cor.test(data$`SQ-EQ`, data$Grade_average)
print(SQ_minus_EQ_cor)

# Check SQ score
SQ_cor <- cor.test(data$`SQ score`, data$Grade_average)
print(SQ_cor)

# Check EQ score
EQ_cor <- cor.test(data$`EQ score`, data$Grade_average)
print(EQ_cor)
```

These functions returned a correlation coefficient that indicated the relationship between the variables, a p- value indicating statistical significance and a confidence interval.

The SQ-EQ and grade average correlation test returned the following values:

- Correlation: 0.367
- P-Value: 0.196
- 95% Confidence Interval: [-0.20, 0.75]

Based on the results we can see a moderate positive correlation was found between SQ-EQ scores and average programming grade, this suggests that participants with higher SQ scores and lower EQ scores tended to have higher grades. However, based on the p-value being greater that 0.05 we cannot say that this relationship is statistically significant. The trend we are observing could be by chance or being caused by a different underlying factor. The scatterplot below visualizes the relationship between SQ-EQ and programming grade average among the participants. It confirms the results from our correlation analysis there is a clear positive moderate correlation, but the data is very scattered.



The two cognitive measures were then tested individually for correlation with programming aptitude. The SQ score and grade average correlation test returned the following:

- Correlation: 0.356
- P-Value: 0.212
- 95% Confidence Interval: [-0.22, 0.75]

These results are similar to the SQ-EQ correlation test but reveal that SQ on its own showed a slightly lower correlation than SQ-EQ and also had a higher p-value. Again, these results were not statistically significant.

Finally, the correlation test was performed on EQ providing these results:

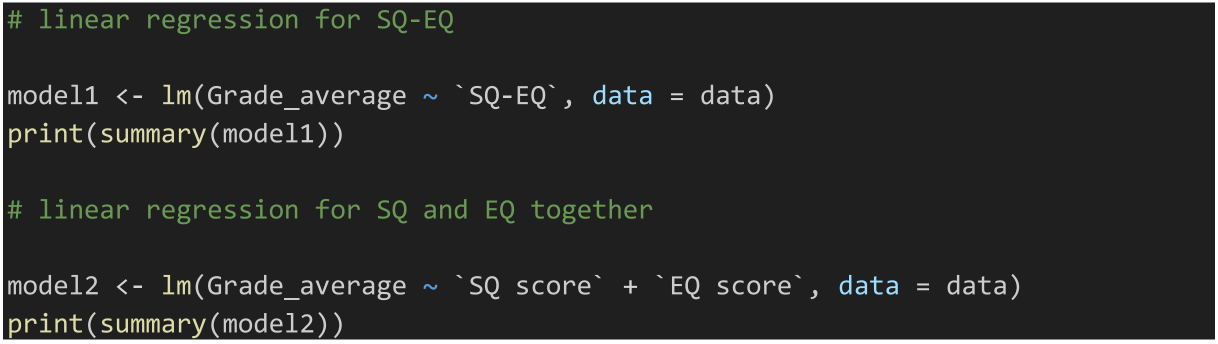
- Correlation: -0.167
- P-Value: 0.569
- 95% Confidence Interval: [-0.64, 0.40]

These results show a weak negative correlation between EQ and average grade suggesting that lower EQ score may lead to higher programming grades. However, the correlation is quite weak and not statistically significant with a very high p-value of 0.569.

Regression Models Analysis

After analysing the correlation to investigate the predictive power of SQ, EQ and SQ-EQ further, linear regression models were developed. The models attempt to explain how much of the variation in programming performance could be explained by the cognitive styles. Two models were developed. Model 1 used SQ-EQ to predict grade average and model 2 used SQ and EQ together as separate predictors. We again used simple R scripts to accomplish this as they have prebuilt regression model functions.

R code for creating the regression models



```
# linear regression for SQ-EQ

model1 <- lm(Grade_average ~ `SQ-EQ`, data = data)
print(summary(model1))

# linear regression for SQ and EQ together

model2 <- lm(Grade_average ~ `SQ score` + `EQ score`, data = data)
print(summary(model2))
```

The models produced the following results.

Model 1: SQ-EQ as a predictor for grade average:

- Slope: 0.79
- Intercept: 68.38
- p-value: 0.196
- R^2: 0.135

The model again suggests a positive trend between SQ-EQ and programming performance with the slope of 0.79 meaning that for every one-point increase in SQ-EQ the average grade goes up by 0.79%. The R^2 of 0.135 means that SQ-EQ can explain 13.5% of the variance in the grades. However, this is not statistically significant, and we cannot say for certain that SQ-EQ results in higher programming ability.

Model 2: SQ and EQ together as separate predictors for grade average:

- SQ Slope: 1.00
- EQ Slope: -0.49
- SQ p-value: 0.24
- EQ p-value: 0.61
- R^2:0.147

This model shows a slight positive trend between SQ and programming ability (Every 1-point increase in SQ predicts a 1% higher grade.) and a slightly negative relationship between EQ and programming grades (Every 1-point increase in EQ predicts a 0.49% lower grade.). the R^2 value suggests that combining using SQ and EQ together as independent values explained 14.7% of the variation in grades, slightly better than the SQ-EQ model. Again, this is not statistically significant, and we cannot say for certain that SQ and EQ scores influence programming ability.

Statistical Analysis Conclusion:

Based on the results from the statistical analysis we cannot definitely conclude that SQ, EQ or SQ-EQ can predict programming ability. While the data revealed some interesting trends and moderate correlations, none of these relationships were statistically significant due to the very large p-values observed. The regression models developed were only able to account for a small proportion of the variance in the programming grades, showing limited predictive power. The limited explanatory power and high p-values could be due to the small sample size of the data. However, these results do not definitively rule out the potential of SQ and EQ predicting programming ability. Despite the lack of statistical significance, the trends and findings are intriguing and highlight patterns that I believe warrant further investigation. Another study with a larger sample size may help clarify and determine whether the cognitive quotients SQ and EQ can predict programming success, encouraging future research into this area of programming aptitude prediction.

Report Conclusion:

What I Would Do Differently:

Upon completion of this project and reflecting on what I had done I have identified several key aspects of the project that could have been done differently to enhance the quality of the project. If I were to complete this project again, I make the following changes.

Targeting First Year students:

A significant change I would make to the project would be altering the target population for the survey to specifically target first year computer science students who have little to no experience in programming. Given that the original project surveyed students from all year groups, some of the participants had already engaged in several years of programming modules. This means their cognitive styles and ways of thinking could have been influenced or changed by their experience learning how to code. By only selecting first year students with minimal programming background, we would be able to test the hypothesis against their natural systemizing and empathizing quotients, we could then even extend the study to test their SQ and EQ scores once they had completed their programming courses to examine any changes in their cognitive styles. This approach to the project could not have been implanted in the original project due to time constraints. Given that you only begin your final year project when the college academic year begins, it was too late to organise the study to use a group of incoming first year computer science students.

Conducting the Survey in Person:

For my project I had distributed the survey online by emailing various lecturers and asking them to forward the survey to any programming classes they had. While this was convenient and gave us access to a large group of suitable participants for the survey, the overall response numbers were quite low. It is possible that the large number of emails students receive every day, and the busy and stressful lives of students reduced the number of participants. I believe we would see higher levels of engagement if instead of asking lecturers to email students about the survey we asked them if we could conduct the quick survey during a scheduled lecture or lab time. This method would have also allowed for any questions the students may have had about the survey to be answered in person.

Coordinating with Lectures:

If repeating the project again an important change I would make would be involving the programming lectures more in the project. Coordinating with the lecturers to collect students grades directly instead of relying on self-reported grades would have been extremely beneficial for the project. This approach would remove any risk of student consciously or subconsciously selecting grades they had done better in and any chance of students lying about their grades. Additionally, we could take this even further by working with the lecturers directly to design a specific programming task or assessment tool catered towards the study which would allow for consistent comparison of results across all participants. Again, these changes could not be implanted in the original project due to time constraints as by the time I began my final year project it was too late to organise and coordinate with lecturers.

Ethics Approval Process:

One major roadblock encountered during the project was the delay in receiving ethical approval. The overall process took much longer than I had anticipated and resulted in several emails with the ethics committee requesting changes to the application forms and the design of the study. The delay in ethical approval resulted in the delay of conducting the survey later into the academic semester than I had hoped. This meant we had a smaller window for data collection and potentially resulted in fewer responses as students were focused on academic commitments later into the college semester. If I was to complete the project again, I would begin the ethical approval process earlier to allow more time to address any feedback received from the ethics committee without delaying other aspects of the project.

Overall, while I do believe that the original project was planned and completed to a high standard the changes outlined above would improve the study's reliability, engagement and accuracy. If I was to complete the project again, I would start earlier, ideally over the summer before the academic year began to introduce the improvements outlined above.

Conclusion:

This project began with the goal of exploring if the cognitive measures systemizing quotient (SQ) and empathizing quotient (EQ) could be used as effective, reliable indicators for programming aptitude. Through a literature review, survey design, data collection, data analysis and the development of a web-based tool I was able to examine and investigate this hypothesis in detail.

While the data analysis showed a moderate positive correlation between SQ-EQ and programming aptitude, the results were not statistically significant. Nevertheless, I still believe the study produced some intriguing results and highlighted areas for further research in examining the influence of cognitive behaviour on programming aptitude. A replication of this project or the completion of a similar project with a larger sample size could results in more statistically significant results.

Overall, this project took an interesting hypothesis, "Can SQ minus EQ predict programming aptitude" and examined it in detail, outlining the importance and usefulness of accurate programming aptitude prediction. The project also showed how results and findings from similar research projects can be used to develop simple web tools which can be used to measure programming aptitude. I believe programming aptitude is an area of research with real world implications, and I look forward to more research being conducted in this field.

References:

Groen, Y., Fuermaier, A.B., Den Heijer, A.E., Tucha, O. and Althaus, M. (2015) ‘The Empathy and Systemizing Quotient: The psychometric properties of the Dutch version and a review of the cross-cultural stability’, *Journal of Autism and Developmental Disorders*, 45(9), 2848–2864. doi: 10.1007/s10803-015-2448-z.

TestGorilla (n.d.) ‘What is a programming aptitude test?’, *TestGorilla*. Available at: https://www.testgorilla.com/blog/programming-aptitude-test/ (Accessed: 28-11-2024).

McGuire, P. (2019) ‘Some third-level computing courses have 80% drop-out rate’, *The Irish Times*, 18 February. Available at: https://www.irishtimes.com/news/education/some-third-level-computing-courses-have-80-drop-out-rate-1.3792910 (Accessed: 28-11-2024).

University of Limerick (n.d.) ‘Computer Science Common Entry (LM121)’, *University of Limerick*. Available at: https://www.ul.ie/courses/computer-science-common-entry (Accessed: 11-12-2024).

Business Wire (2021) ‘Skillsoft to acquire Codecademy, a leading platform for learning high-demand technical skills, creating a worldwide community of more than 85 million learners’, Business Wire, 22 December. Available at: https://www.businesswire.com/news/home/20211221005804/en/Skillsoft-to-Acquire-Codecademy-a-Leading-Platform-for-Learning-High-Demand-Technical-Skills-Creating-a-Worldwide-Community-of-More-Than-85-Million-Learners/ (Accessed: 11-12-2024)

American Psychological Association (2018). ‘Aptitude’, APA Dictionary of Psychology. Available at: https://dictionary.apa.org/aptitude (Accessed: 14-12-2024).

Dehnadi, S. (2006) ‘Testing Programming Aptitude’, *18th Workshop of the Psychology of Programming Interest Group*. University of Sussex, September.

Caspersen, M.E., Bennedsen, J. and Larsen, K.D. (2007) ‘Mental Models and Programming Aptitude’, *Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, Dundee, Scotland, United Kingdom, pp. 205-209. doi: 10.1145/1268784.1268845.

American Psychological Association (2018). ‘Cognitive ability’, APA Dictionary of Psychology. Available at: https://dictionary.apa.org/aptitude (Accessed: 16-12-2024).

Cambridge Dictionary (n.d.). ‘Scholastic’, Cambridge Dictionary. Available at: https://dictionary.cambridge.org/dictionary/english/scholastic (Accessed: 16-12-2024).

Tukiainen, M. and Mönkkönen, E. (2002) ‘Programming aptitude testing as a prediction of learning to program’, 14th Workshop of the Psychology of Programming Interest Group, Brunel University, June, pp. 45–57.

Alspaugh, C.A. (1972) ‘Identification of Some Components of Computer Programming Aptitude’, *Journal for Research in Mathematics Education*

Bergin, S. and Reilly, R. (2005) ‘Programming: Factors that Influence Success’, *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, St. Louis, Missouri, USA

Scott, M.J. and Ghinea, G. (2014) ‘On the Domain-Specificity of Mindsets: The Relationship Between Aptitude Beliefs and Programming Practice’, *IEEE Transactions on Education*

Baron-Cohen, S. (2003) *The Essential Difference: Men, Women and the Extreme Male Brain*. London: Penguin.

The Guardian (n.d.) ‘The Systemizing Quotient (SQ)’. Available at: https://www.theguardian.com (Accessed: 23-12-2024).

American College of Surgeons (n.d.) ‘The Empathy Quotient (EQ): A 60-Item Questionnaire’. Available at: https://www.facs.org/media/mbseliy5/empathy-quotient (Accessed: 23-12-2024).

Wray, S. (2007) ‘SQ Minus EQ Can Predict Programming Aptitude’, *19th Annual Workshop of the Psychology of Programming Interest Group (PPIG)*, Bournemouth, UK

Borzovs, J., Kozmina, N., Niedrite, L., and Solodovnikova, D. (2017) ‘Can SQ and EQ Values and Their Difference Indicate Programming Aptitude to Reduce Dropout Rate?’, *Symposium on Advances in Databases and Information Systems*, Communications in Computer and Information Science,

Cheah, C.S. (2020) ‘Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review’, *Contemporary Educational Technology*

Autism Research Centre, University of Cambridge (n.d.) ‘Tests - Autism Research Centre’, Autism Research Centre. Available at: https://www.autismresearchcentre.com/tests (Accessed: 13-12-2024).

University of Limerick, Research Ethics (n.d.) ‘Research Ethics’, Faculty of Science and Engineering. Available at: https://www.ul.ie/scieng/scieng-research/research-ethics (Accessed: 13-12-2024).