Exploring Programming Aptitude: Comparing the Predictive Utility of Language Aptitude Subskills for Python and Java Learning

Malayka Mottarella (mmottare@uw.edu)
Department of Psychology, University of Washington
119A Guthrie Hall, Seattle, WA 98195 USA

Katherine Mortimore (kmortimore@ihmc.org)
Institute for Human & Machine Cognition
40 S Alcaniz St, Pensacola, FL 32502 USA

Chantel S. Prat (csprat@uw.edu)
Department of Psychology and Institute for Leaning and Brain Science, University of Washington
119A Guthrie Hall, Seattle, WA 98195 USA

Abstract

The present study examines how natural language aptitude subskills predict individual differences in learning Python and Java. Past work has demonstrated that overall performance on the Modern Language Aptitude Test (MLAT), a standardized measure of language aptitude, is a strong predictor of both the speed and accuracy with which individuals learn Python. However, language aptitude is a broad multidimensional construct made up of individual subskills. In the present study, we examine how two of these subskills - sensitivity to form and meaning mapping - relate to programming outcomes in both Python and Java. Results indicate that both sensitivity to form (MLAT IV) and meaning mapping (MLAT V) are related to programming acquisition in both languages - this relationship remains even after controlling for fluid intelligence. We also examined how programming skills tied to semantics and syntax related between Python and Java in a subset of learners who learned both languages. These results demonstrated that proficiency in Python predicted individual differences in both syntactic and semantic knowledge in Java. Taken together, these results further elucidate the role of natural language aptitude in programming learning and suggest that semantic and syntactic content may transfer across programming languages.

Keywords: programming; learning; language aptitude; individual differences

Introduction

The ability to code in a computer programming language is an increasingly desirable skill in both the workforce and in educational settings. However, learning to program is notoriously difficult, and learners vary greatly in both the speed and accuracy with which they acquire programming skills. Empirical work dedicated to understanding the factors that drive success in modern programming languages has been on the rise (e.g., Floyd, Santander, & Weimer, 2017; Parkinson & Cutts, 2022; Prat et al., 2020; Ivanonva et al., 2020); but much is still unknown about the cognitive factors underpinning acquisition of a programming language.

By comparison, an extensive body of research has investigated the factors that drive acquisition of a second *natural* language (for reviews see Chalmers et al., 2021; Wen, Biedroń, & Skehan, 2017). This research may function well as a starting framework for understanding individual differences in learning programming languages, as the cognitive parallels between the two skills have been well laid out (Fedorenko et al., 2019). At least two studies have shown that the Modern Language Aptitude Test (MLAT: Carrol & Sapon, 1959), a classic predictor of natural language learning, showed strong predictive utility for explaining variation in Python learning (Kuo et al., 2022; Prat et al., 2020).

Natural language aptitude is an umbrella term used to describe a set of language-related subskills that characterize one's potential to successfully acquire a new language in the future. Individually, these subskills assess factors such as phonetic coding, grammatical sensitivity, rote memory, and inductive learning (Carroll, 1981). The MLAT is a standardized measure comprised of five subtests, which differentially draw on the underlying subskills that contribute to natural language aptitude (Carroll & Sapon, 1959). The MLAT has been reliably used as the "gold standard" assessment of natural language aptitude for over 60 years (Chalmers et al., 2021; Sasaki, 2012). To the best of our knowledge, Prat and colleagues' (2020) study demonstrating that MLAT performance explains ~30% of the variability in Python acquisition marks the first use of the MLAT as a predictor of programming learning skills. However, Prat and colleagues (2020) only used the total cumulative score on the MLAT to predict Python learning, making it difficult to determine the specific language aptitude subskills that may be driving this effect. Examining how performance on MLAT subtests that have clear theoretical relevance for programming language acquisition predict subsequent

In L. K. Samuelson, S. L. Frank, M. Toneva, A. Mackey, & E. Hazeltine (Eds.), *Proceedings of the 46th Annual Conference of the Cognitive Science Society.*

learning may provide a more nuanced account of which specific facets of language aptitude are implicated in learning a programming language.

Two candidate subskills that seem particularly relevant in programming are the ability to learn and remember the meanings of new informational units (i.e., rote memory or "meaning mapping") and the ability to parse the functional role of an informational unit within a larger context (i.e., grammatical sensitivity or "form"). This distinction in processing is supported by recent work demonstrating that skilled programmers show canonical N400 and P600 event-related potentials (ERPs) when reading code with manipulations to meaning and form, respectively, and that neural sensitivity to form violations increases with programming skill (Kuo & Prat, 2023). This work suggests that programmers show similar progressions in sensitivity to meaning and form aspects of code during skill acquisition as learners of second natural languages do (McLaughlin et al., 2010). Thus, we hypothesize that sensitivity to form and meaning may be important language aptitude subskills underpinning successful programming acquisition.

Taken together, previous work demonstrates that language aptitude measures predict individual differences in programming learning, and that sensitivity to form and meaning may be of specific importance for understanding the interplay between second language learning and learning programming languages. The present study extends this work in several novel ways. First, we examine how individual differences in sensitivity to form and meaning mapping, assessed respectively using the MLAT IV and V subtests, predict performance on a comprehensive set of programming outcome measures. Second, we will use a much larger and more diverse sample of Python learners compared to past studies (Kuo & Prat, 2023; Prat et al., 2020). Third, we will assess whether the relation between language aptitude subskills and programming outcomes generalizes from one programming language (Python) to another (Java), which is notoriously more difficult to learn and less "reader friendly". Finally, if meaning and form sensitivity do relate to learning outcomes in both Python and Java, we will explore whether outcomes tied to semantics and syntax relate within an individual across programming languages, using a subset of participants who learned both languages.

Methods

Participants

All participants were healthy, right-handed, young adults between the ages of 18 and 35. Native English fluency was a requirement for participation in both studies, although recruitment was not solely limited to monolingual English speakers as has been done in prior work (Prat et al., 2020). All participants provided informed consent before completing any experimental tasks and received compensation for their participation.

Python Group. Ninety-eight participants with no prior programming experience were recruited for the Python group. Three participants withdrew from the study before completing all learning sessions, resulting in a final group size of 95 participants (M = 20.66 years; 72 female).

Java Group. Fifty-four participants were recruited for the Java group. Participants were required either to have no prior programming experience *or* to have prior experience solely in Python. Three participants withdrew from the study before completing all learning sessions, and two participants were dismissed for not meeting the inclusion criteria (e.g. having previous experience in C). Thus, the final Java sample included 49 participants (M = 22.16 years, 37 females). Within the Java group, 24 participants had prior Python experience, and 16 of these participants learned Python specifically in our previous study and are included in the Python group.

Materials

Modern Language Aptitude Test (MLAT). The MLAT is a 1-hour long, paper-and-pencil measure of natural language aptitude consisting of five subtests (Carol & Sapon, 1959). Given our prediction that sensitivity to form and meaning mapping may be particularly important for programming, we focused our analyses on the MLAT IV and V subtests. In *MLAT IV: Words in Sentences*, participants were presented with a sample sentence and asked to identify the word in a second sentence that most closely performed the same function as the one indicated in the first. In *MLAT V: Paired Associates*, participants learned and were then tested on twenty-four Kurdish-English word associations. MLAT IV and MLAT V were used as our indices of form and meaning mapping sensitivity, respectively. Performance on each subtest was quantified as the number of correct items in that section.

Ravens Advanced Progressive Matrices (RAPM). The RAPM is a standardized measure of fluid intelligence in which participants are tasked with determining which option of eight possible choices completes a presented pattern. Participants are given 20-minutes to complete 18 test items which increase in difficulty as the test progresses.

Programming Learning Rate. The speed of programming learning for each participant was determined by fitting a regression line, with a fixed intercept at zero, to the data modeling the programming lesson each participant had finished at the end of each learning session. The slope of the

regression line was operationalized as the participant's learning rate.

Declarative Knowledge Tests. The declarative knowledge tests consisted of 50 multiple-choice items, half designed to assess ***semantic knowledge*** (e.g., what is concatenation?) and the remaining half evaluating ***syntax knowledge*** (e.g., which line of code will NOT run?). Participants were allotted 30 minutes to complete the test. Accuracy on the test was quantified separately for the semantic and syntax portions by calculating the number of correct items for each question type out of 25.

Java participants with prior Python knowledge completed both declarative knowledge tests. As part of their pre-learning battery, they were given the Python declarative knowledge test to evaluate their proficiency in Python before learning Java. Subsequently, following their Java learning, they completed the Java declarative knowledge test along with the other post-learning assessments.

Coding Tests. The coding tests measured participants' ability to generate code correctly. The tests were adapted from free-form projects included in the appropriate Codecademy course. Python coding accuracy was computed using the score on Codecademy's Rock, Paper, Scissors project in the Learn Python 2 course (Score/51 points).

Java coding accuracy was computed by averaging performance on two projects adapted from Codecademy's Learn Java course - the Build a Droid (Score/59 points) and the Simple Car Loan Payment Calculator (Score/41 points) projects. Participants were given 30 minutes to program the projects as specified in the projects' instructions. Following similar methods to those used by Prat et al. (2020), performance was assessed using a rubric developed by programming experts. Participants received full credit if they correctly coded a step, partial credit for minor errors, and no credit for missed or largely incorrect steps. All programming tests were independently scored by two reviewers using the appropriate rubric.

Debugging Tests. The debugging tests assessed participants' ability to identify and correct programming errors. All debugging tests were based on incorrect versions of the previously discussed coding tests. Participants were given 15 minutes to correct as many errors as possible towards the goal of getting the code to run as specified in the project instructions. The debugging tests were independently reviewed by two reviewers using a rubric created by programming experts. Participants received full credit for corrected errors, partial credit for identified but uncorrected errors, and no credit for errors they did not identify. Additionally, participants were penalized for "correcting" a line of code with no errors. Accuracies were determined as the number of errors corrected, minus any penalties, out of the total number of points possible (Python: 22, Java Droid: 22, Java Car Loan: 20). The final Java debugging score averaged together performance on the Droid and Car Loan projects.

Procedure

Pre-learning Sessions. Before programming learning, participants completed three 1.5-2.0 hour long sessions during which we collected a comprehensive battery of demographic, general cognitive, and language assessments. The MLAT and RAPM were administered during these sessions.

Learning Sessions. Next, participants underwent eight one-hour, online learning sessions during which they worked through either Codecademy's *Learn Python 2* course or *Learn Java* course at their own pace. To advance to the next lesson, participants were required to achieve a minimum score of 50% on Codecademy's end-of-lesson quizzes. At the end of each learning session, we recorded which lesson number the participant reached, which was subsequently used to compute each participant's learning rate.

Post-learning Assessment Sessions. Following the eight hours of programming learning, participants took the declarative multiple-choice knowledge test, coding test(s), and debugging test(s).

Results

Correlating Language Aptitude Subskills with Programming Outcomes

To examine the relation between individual differences in sensitivity to form and meaning mapping and programming learning outcomes, MLAT IV and V scores were separately correlated with programming outcomes for the Python and Java learning groups (see Figure 1; for descriptive statistics see Table 1). Significance levels are reported using a False Discovery Rate correction. We also compared the magnitudes of the correlation coefficients between the MLAT and programming outcome measures between the Python and Java groups (Soper, 2024) and ran partial correlations to control for individual differences in RAPM.

Our results indicated that the MLAT IV measure of sensitivity to form was a significant predictor of programming success for all Java and Python learning outcomes. Specifically, MLAT IV was positively correlated with learning rate in both Java [$r(48) = 0.50$, $pfdr < 0.001$] and Python [$r(94) = 0.40$, $pfdr < 0.001$]. In both programming languages, MLAT IV was also positively correlated with performance on the declarative knowledge test. These correlations were significant, for both questions testing semantic knowledge [Java: $r(48) = 0.42$, $pfdr = 0.002$; Python: $r(94) = 0.36$, $pfdr < 0.001$] and those testing syntactic

Table 1. Descriptive Statistics for Measures of Interest

		Descriptive Statistics				
Language Aptitude[a]	**Group**	**N**	**Mean**	**SD[d]**	**Min[d]**	**Max[d]**
MLAT IV	Python	95	19.14	4.84	8.00	38.00
	Java	49	18.59	6.34	7.00	38.00
MLAT V	Python	95	18.57	5.20	4.00	24.00
	Java	49	17.78	5.48	3.00	24.00
Fluid Intelligence[b]						
RAPM	Python	95	0.68	0.16	0.28	1.0
	Java	49	0.67	0.18	0.22	1.0
Learning Outcomes[c]						
Learning Rate	Python	95	1.33	0.35	0.73	2.65
	Java	49	1.16	0.42	0.49	2.36
Declarative Knowledge						
Semantics	Python	95	0.72	0.13	0.40	1.00
	Java	49	0.64	0.12	0.32	0.88
Syntax	Python	95	0.68	0.14	0.28	0.96
	Java	49	0.66	0.19	0.24	1.00
Coding Test(s)	Python	95	0.51	0.18	0.06	0.87
	Java	49	0.60	0.21	0.07	0.99
Debugging Test(s)	Python	95	0.49	0.20	0.08	0.96
	Java	49	0.61	0.23	0.00	0.98

[a]MLAT scores are the total number of correct items on the respective subtest.

[b]RAPM scores are the percentage of correct items

[c]All learning outcomes, except learning rate, are accuracies reported as decimals.

[d]*SD* = Standard Deviation, *Min* = Minimum score, *Max* = Maximum score.

knowledge [Java: $r(48) = 0.55$, $pfdr < 0.001$; Python: $r(94) = 0.47$, $pfdr < 0.001$].

Finally, we examined how MLAT IV predicted measures of coding and debugging in both languages. Results indicated that MLAT IV showed a significant positive correlation with coding performance in the Java group [$r(48) = 0.66$, $pfdr < 0.001$] and in the Python group [$r(94) = 0.37$, $pfdr < 0.001$]. A similar pattern was observed for the correlation between MLAT IV and debugging performance. While both groups showed a significant positive correlation between MLAT IV and debugging, the magnitude of the correlation was slightly stronger for the Java group [$r(48) = 0.64$, $pfdr < 0.001$] than the Python group [$r(94) = 0.45$, $pfdr < 0.001$]. The tests for differences in correlation coefficient strength between the groups found that *only* the magnitude of the MLAT IV and coding performance correlation was significantly stronger in the Java group than in the Python group ($z = 2.24$, $p = 0.03$). The other correlation coefficient magnitudes were not significantly different between groups ($ps > 0.10$).

MLAT V, meaning mapping, also predicted success in some programming outcomes, but these results were not as consistent across outcome measures or as robust as those seen with MLAT IV, form sensitivity. MLAT V did not correlate significantly with learning rate in the Java group [$r(48) = 0.19$, $pfdr = 0.18$] and was only marginally positively correlated in the Python group [$r(94) = 0.20$, $pfdr = 0.05$]. MLAT V was positively correlated with the portions of the declarative knowledge test measuring semantic knowledge, marginally in the Java group [$r(48) = 0.26$, $pfdr = 0.07$] and significantly in the Python group [$r(94) = 0.35$, $pfdr < 0.001$]. For syntax knowledge, MLAT V was a significant positive predictor in both groups [Java: $r(48) = 0.33$, $pfdr = 0.02$; Python: $r(94) = 0.26$, $pfdr = 0.01$]. Next, we examined the relation between MLAT V and real-time coding and debugging skills. MLAT V was positively correlated with coding performance significantly in the Java group [$r(48) = 0.44$, $pfdr = 0.002$] and marginally in the Python group [$r(94) = 0.20$, $pfdr = 0.07$]. While for debugging, MLAT V was significantly positively correlated in both groups [Java: $r(48) = 0.47$, $pfdr = 0.001$; Python: $r(94) = 0.27$, $pfdr = 0.01$]. None of the correlation coefficient magnitudes were significantly different between the Java and Python groups ($ps > 0.10$).

In summary, higher MLAT IV, form sensitivity, and MLAT V, meaning mapping, scores consistently correlated with better programming outcomes across programming languages. Though the magnitudes of the correlations

between programming outcomes and MLAT IV were generally stronger than those with MLAT V, statistical comparisons of the relative strengths of these correlation coefficients revealed no significant differences (*ps* > 0.05). To control for the possibility that a general cognitive factor was not jointly predicting MLAT scores and programming we computed partial correlations between the MLAT measures of interest and the programming outcomes controlling for individual differences in fluid intelligence

(RAPM). As depicted in Figure 1, all correlations remained significant after controlling for RAPM with the exception of the correlation between MLAT V and Syntax in the Java group [$r(48) = 0.25$, $p = 0.09$]. This result suggests that both MLAT measures explain unique variance in programming learning that is not underpinned by individual differences in fluid intelligence as measured by the RAPM.

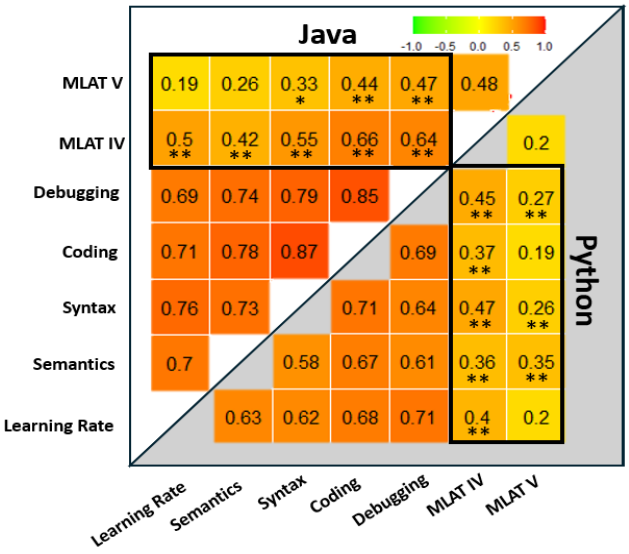

Figure 1. Relation Between MLAT Subtests and Programming Outcomes. Top: Correlations for the Java group. Bottom: Correlations for the Python group. The black boxes highlight the correlations between the MLAT subtests and programming outcome measures. * $p < 0.05$; ** $p < 0.05$ after RAPM partial correlation

Relating Individual Differences in Semantics and Syntax Across Programming Languages

To further investigate the role of meaning mapping and form sensitivity in programming, we examined how programming outcomes tied to semantics and syntax related to one another across Java and Python. To examine the convergence of these skills across programming languages, semantic and syntactic declarative knowledge test scores were correlated across Python and Java. These exploratory analyses were conducted on the subset of participants (N = 16) who learned both programming languages. All participants in this subset learned Python prior to Java.

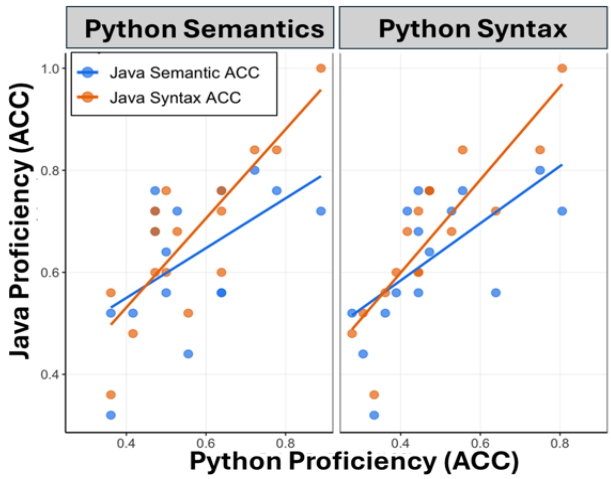

Figure 2. Scatterplot depicting the relation between semantic and syntax knowledge. Left: Depicts the correlations between Python semantics and Java semantics (blue) and syntax (orange). Right: Depicts the correlations between Python syntax and Java semantics (blue) and syntax (orange).

The declarative knowledge outcome measures were strongly positively correlated across programming languages. Individual differences in Python syntax positively correlated with both Java syntax [$r(15) = 0.87$, $pfdr < 0.001$] and Java semantics [$r(15) = 0.62$, $pfdr = 0.01$]. Similarly, Python semantics positively correlated with both Java syntax [$r(15) = 0.83$, $pfdr < 0.001$] and Java semantics [$r(15) = 0.53$, $pfdr = 0.03$]. While the magnitude of the correlations between Python proficiency and Java syntax were stronger than those between Python proficiency and Java semantics (see Figure 2), the differences between the correlation coefficients were not statistically significant (*ps* > 0.1).

Discussion

The present study examined how individual differences in language aptitude subskills indexing form and meaning mapping sensitivity related to programming outcomes in both Python and Java. Our results meaningfully extend past work (e.g., Kuo et al., 2022; Kuo & Prat, 2023; Prat et al., 2020) by showing that: 1) both subskills predicted some programming outcomes in Python, 2) both subskills predicted some programming outcomes in Java, and 3) form sensitivity trended towards being the stronger predictor of the two subskills. In the sections that follow we expand on these contributions and their implications for programming acquisition.

Our results showed that both language aptitude measures of interest were related to programming acquisition to some degree. Specifically, we found that MLAT IV, measuring grammatical sensitivity correlated with all programming outcomes across our Python and Java groups. Whereas

MLAT V, meaning mapping, was only correlated with select programming outcomes. These correlations remained significant after controlling for RAPM scores, which supports the idea that natural language skills contribute to programming beyond what can be explained by variation in general fluid reasoning. These findings have important implications for understanding how language aptitude subskills assessing meaning mapping and grammatical sensitivity contribute to programming skill acquisition. These two subskills were selected based on their theoretically hypothesized importance in programming, however, examining less intuitively important language aptitude subskills, such as phonemic coding, is an important next step in deriving a holistic understanding of the role of natural language aptitude in programming. Ongoing work in our lab plans to address this question.

The intention of using a large battery of programming outcome measures was to tap into different components of learning a programming language, which may rely more or less strongly on distinct cognitive abilities. However, our results indicated that by and large, each MLAT measure showed relatively similar predictive utility across programming outcomes, particularly when these outcomes were considered within a programming language. This pattern highlights a central challenge in complex skill learning research. Namely, it is difficult to determine the extent to which a complex skill, like programming, should be broken into separate components, especially when these components are interrelated or build on one another hierarchically (e.g., Federenko et al., 2019). This finding is also in line with theories suggesting programming skills are highly interconnected, such that proficiency in one programming skill is strongly related to proficiency in another programming skill (Robins, 2010). Future work using approaches such as factor analysis may help disentangle how much unique information is collected in these programming outcomes.

Our results demonstrated a persistent trend of programming outcomes relating more robustly to grammatical sensitivity than meaning mapping. In our correlations between the MLAT subtests and programming outcomes, MLAT IV was a stronger predictor than MLAT V for all programming outcomes across both Python and Java. While the magnitudes of the correlation coefficients were only *statistically* different between MLAT IV and V for the coding test in Java, this general pattern held across every measure in both programming languages. Our follow-up analysis of the subset of participants with both Python and Java experience further highlights the importance of syntactic information in programming acquisition. These results demonstrated that greater Python proficiency predicted Java syntax more strongly than Java semantics. While the conclusions that can be drawn from these analyses are limited due to the sample size, they do raise the important discussion of which underlying components may transfer across programming languages.

Prior work examining transfer, has primarily focused on the idea that programming concepts (e.g., loops, conditionals, etc.) are used in multiple languages, resulting in semantic transfer of these concepts across languages (e.g., Kao, Matlen, & Weintrop, 2022; Tshukudu & Cutts, 2020). However, our results suggest that even when the surface-level syntactic features may change, one's ability to parse and understand syntactic features in context may be an overlooked ability supporting programming aptitude. This idea is consistent with Kuo & Prat's (2023) finding that greater programming skill was associated with a canonical shift from an N400 to a P600 ERP deflection when viewing syntactic violations in code. In the second-language literature, this ERP shift is referred to as "grammaticalization" and has been used as a neural index corresponding to behavioral second-language proficiency (McLaughlin et al., 2010).

The results of the present study and those of Kuo & Prat (2023) suggest that grammaticalization may serve as a similar inflection point in programming learning. It is an open question whether the timescale of grammaticalization may look fundamentally different in programming languages. One speculation is that grammaticalization may be even more central to programming than it is in natural language learning. In natural language learning, syntax is typically learned through slower procedural memory systems relative to vocabulary learning (e.g., Ullman, 2001; Walker et al., 2020). This makes sense considering that humans are highly skilled at parsing meaning even when the syntax of the speaker may not be perfect (e.g., Fairchild & Papafragou, 2018). In contrast, syntax in programming languages is learned through rigid binary reinforcement such that the code will not compile if the syntax is incorrect. Thus, it is plausible that an earlier grammaticalization shift may be even more advantageous in programming than in natural language. This idea is in line with the results reported herein showing that greater sensitivity to syntactic elements may underpin both the success of learning a first programming language and the degree of transfer between programming languages.

Taken together the results of the present study demonstrate that language aptitude subskills tied to meaning mapping and form sensitivity show strong predictive utility for explaining variance across multiple outcomes when learning two distinct programming languages. These results have implications for further informing our understanding of the cognitive computations that contribute to both programming aptitude and transfer between programming languages.

Acknowledgments

This research was supported by an award from the Office of Naval Research (GR010970) to Chantel S. Prat.

References

Carroll, J. B. (1981). Twenty-five years of research on foreign language aptitude. In K. C. Diller (Ed.), *Individual differences and universals in language learning aptitude*. Rowley, MA: Newbury House.

Carroll, J. B., & Sapon, S. M. (1959). Modern Language Aptitude Test (MLAT). New York: Psychology Corporation.

Chalmers, J., Eisenchlas, S. A., Munro, A., & Schalley, A. C. (2021). Sixty years of second language aptitude research: A systematic quantitative literature review. *Language and Linguistics Compass*, *15*(11), e12440.

Fairchild, S., & Papafragou, A. (2018). Sins of omission are more likely to be forgiven in non-native speakers. *Cognition*, *181*, 80–92.

Fedorenko, E., Ivanova, A., Dhamala, R., & Bers, M. U. (2019). The language of programming: A cognitive perspective. *Trends in Cognitive Sciences*, *23*(7), 525–528.

Floyd, B., Santander, T., & Weimer, W. (2017). Decoding the Representation of Code in the Brain: An fMRI Study of Code Review and Expertise. *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 175–186.

Ivanova, A. A., Srikant, S., Sueoka, Y., Kean, H. H., Dhamala, R., O'Reilly, U.-M., Bers, M. U., & Fedorenko, E. (2020). Comprehension of computer code relies primarily on domain-general executive brain regions. *eLife*, *9*, e58906.

Kao, Y., Matlen, B., & Weintrop, D. (2022). From one language to the next: Applications of analogical transfer for programming education. *ACM Transactions on Computing Education*, *22*(4), 1–21.

Kuo, C.-H., Mottarella, M., Haile, T., & Prat, C. S. (2022). Predicting Programming Success: How Intermittent Knowledge Assessments, Individual Psychometrics, and Resting-State EEG Predict Python Programming and Debugging Skills. *2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 1–6.

Kuo, C.-H., & Prat, C. (2023). *Programmers show distinct, language-like brain responses to violations in form and meaning when reading code* [Preprint].

McLaughlin, J., Tanner, D., Pitkänen, I., Frenck-Mestre, C., Inoue, K., Valentine, G., & Osterhout, L. (2010). Brain potentials reveal discrete stages of L2 grammatical learning. *Language Learning*, *60*(s2), 123–150.

Parkinson, J., & Cutts, Q. (2022). Relationships between an early-stage spatial skills test and final CS degree outcomes. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, 293-299.

Prat, C. S., Madhyastha, T. M., Mottarella, M. J., & Kuo, C.-H. (2020). Relating natural language aptitude to individual differences in learning programming languages. *Scientific Reports*, *10*(1), 3817.

Robins, A. (2010). Learning edge momentum: a new account of outcomes in CS1. *Computer Science Education*, *20*(1), 37-71.

Sasaki, M. (2012). The Modern Language Aptitude Test (Paper-and-Pencil Version). *Language Testing*, *29*(2), 315–321.

Soper, D.S. (2024). *Significance of the Difference between Two Correlations Calculator* (Version 4.0). Publisher. https://www.danielsoper.com/statcalc

Tshukudu, E., & Cutts, Q. (2020). Semantic Transfer in Programming Languages: Exploratory Study of Relative Novices. *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 307–313.

Ullman, M. T. (2001). A neurocognitive perspective on language: The declarative/procedural model. *Nature Reviews Neuroscience*, *2*(10), 717–726.

Walker, N., Monaghan, P., Schoetensack, C., & Rebuschat, P. (2020). Distinctions in the acquisition of vocabulary and grammar: An individual differences approach. *Language Learning*, *70*(S2), 221–254.

Wen, Z. (Edward), Biedroń, A., & Skehan, P. (2017). Foreign language aptitude theory: Yesterday, today and tomorrow. *Language Teaching*, *50*(1), 1–31.