


Perceptual Learning Modules (PLM) in CS1: a Negative Result and a Methodological Warning


Ricardo Caceffo

(Unicamp and Univesp, Campinas, Brasil,

 <https://orcid.org/0000-0001-9152-0658>, caceffo@ic.unicamp.br and ricardo.caceffo@univesp.br)


Jacques Wainer

(Unicamp, Campinas, Brasil,

 <https://orcid.org/0000-0001-5201-1244>, wainer@ic.unicamp.br)


Guilherme Gama

(Unicamp, Campinas, Brasil,

 <https://orcid.org/0000-0000-0000-0000>, guilhermegama@gmail.com)


Islene Garcia

(Unicamp, Campinas, Brasil,

 <https://orcid.org/0000-0002-6023-5982>, islene@ic.unicamp.br)

Rodolfo Azevedo

(Unicamp, Campinas, Brasil,

 <https://orcid.org/0000-0002-8803-0401>, rodolfo@ic.unicamp.br)

Abstract: Perceptual Learning Modules (PLMs) is a variation of Perceptual Learning based on multiple-choice questionnaires. There exists successful research of the use of PLMs in math and flight training. The possibility of designing and adopting PLMs in Introductory Programming Courses (CS1) is still an open area of study. The goal of this study is to test whether students that received a PLM training on recognising segments of programs will perform better at writing programs. Two PLM interventions were administered to students. The first intervention was a nonrandom controlled experiment, in which students opted to answer the PLM questionnaire (N=40), while the control group consisted of students that did not answer it (N=629). The second intervention was a randomized controlled experiment with a placebo, in which students were randomly assigned to perform either the PLM questionnaire (N=51) or another a placebo activity (N=51). The different forms of analysis of the first experiment results yielded Cohen's d ranging from 0.23 to 0.34 in favor of the PLM intervention. For the second experiment, the effect size was $d = -0.11$ against the PLM intervention, but the two results were significant. We believe that the cautious conclusion is that there is a null effect in using a PLM activity as part of a CS1 course. The paper is also of interest because of the methodological decisions and techniques used.

Keywords: Computer Science Education, CS1, Perceptual Learning Modules, Controlled Experiment, Propensity Score, Python, Introductory Programming

Categories: D.1.0, D.1.m, F.3.1, F.3.3, K.3.2

DOI: 10.3897/jucs.96347

1 Introduction

Perceptual Learning (PL) is the process where our brains automatically retrieve and identify information, working in a *passive mode* of background processing that does not require active thinking. For example, when looking at a chess board, a chess master would have the ability to (at least in most cases) “see” the big picture, automatically identifying a pattern in the arrangement of the pieces, thus swiftly proceeding with a suitable move. On the other hand, a beginner player, who does not have such patterns internalized, would need to inspect each piece individually, considering how moving that particular piece would impact the game [Kellman et al. 2010, Chase and Simon 1973].

Perceptual Learning Modules (PLM) [Kellman et al. 2010, Thai et al. 2015, Guerlain et al. 2004, Kellman and Kaiser 1994, Tallal et al. 1998, Chen et al. 2014, Carey 2015] are a particular form of Perceptual Learning where the student is shown a large set of multiple-choice questions (in sessions of about one hour) and receive feedback on the correct answer immediately after making their choice. Through this process, the student is trained to automatically recognize certain sets of patterns.

PLMs have been successfully used in areas where some form of pattern recognition is important, such as Radiology and Electrocardiography [Thai et al. 2015, Guerlain et al. 2004], flight training [Kellman and Kaiser 1994], and language learning [Tallal et al. 1998, Chen et al. 2014].

The idea that automatically learning patterns could be useful for an Introduction to Programming course (CS1) stems from two intuitions, which will be elaborated below. This study follows these intuitions and presents the development and assessment of a PLM intervention for a CS1 course in the Python language. In particular, the intervention is performed at the first half of the course, when, as we will discuss below, we expect the effect to be greater.

The first intuition for the automatic recognition of some programming patterns derives from the ideas known as *hierarchy of programming skills* or *learning hierarchy* [Lopez et al. 2008, Harrington and Cheng, 2018, Xie et al. 2019]. The final goal of an Introduction to Programming course is to teach students how to *write* a program, given the description of a problem to be solved or the specification of the program itself. However, there are at least two other steps in this *hierarchy of programming skills* or *learning hierarchy*. At the lower level is the ability to *trace* a program – that is, to compute the output of the program or program components given an input or state of variables. At an intermediary level is the ability to *comprehend* the program, thus being able to explain what the program or program segment does. Although there is a correlation between writing correct code and the abilities of tracing and explaining, it is unclear whether such correlations are causal [Venables et al. 2009, Lister et al. 2009, Harrington and Cheng, 2018]. One particular model of the learning hierarchy is a temporal one [Harrington and Cheng, 2018]. At the beginning of the learning process, tracing and perhaps comprehension are more important to the skill of writing code, but that importance decreases as the code-writing skill improves.

The second intuition is the work by [Van Merriënboer and Paas 1990] on automation of code-writing procedures. The authors argued that having a library of *schemata*, that is, highly structured knowledge, allows programmers to use these schemata to compose programs that solve a particular problem. Although [Van Merriënboer and Paas 1990] discussed in detail how the use of such schemata should be *automatic* in a sense not dissimilar to perceptual learning, their focus is not on *recognising* the schemata but on *using* them.

None of these two intuitions are about *automatic recognition* of patterns or schemata. Studies that specifically link comprehension and writing [Murphy et al. 2012] and other research on program comprehension applied to CS1 courses [Nelson et al. 2017] do not propose that the automatic recognition of short program structures play a part in understanding what a program does. Nonetheless, we find it a reasonable hypothesis that students that spend some cognitive effort recognising some common (short) patterns in programming could perform better in comprehending the program, and given some of the results in the learning hierarchy, could result in better writing performance.

Similarly, the study described in [Van Merriënboer and Paas 1990] is about automation of the use of schemata, and it spans research on teaching practices that could foster such a process; for example, the use of worked examples in teaching [Skudder and Luxton-Reilly 2014, Morrison et al. 2015], incomplete examples, pattern-oriented instruction [Muller and Haberman 2008] and so on. We believe, however, that learning to automatically recognise some of the patterns could help to use said patterns in writing code.

Both the learning hierarchy and the automation of the use of schemata support the idea that the PLM intervention should focus on the early phases of learning. The temporal aspects of the learning hierarchy suggest that comprehension (and thus automatic recognition) should be more important in the early stages of developing the learning skill [Harrington and Cheng, 2018]. Similarly, for the automation of the use of schema theory, early in the process of learning to program, there are fewer schemata to learn and to use. Thus, if the automatic recognition of patterns is useful at all, both theories suggest that their effect should be stronger earlier in the learning process.

This paper is organized as follows: in Section 2 we present the background, related works and motivation; in Section 3 we describe the design process of a PLM for the CS1 course; in Section 4 we present the materials, interventions and results; and in Section 6 we present the discussions and conclusions.

2 Background

2.1 Perceptual Learning Modules

In the literature [Kellman et al. 2010, Thai et al. 2015, Guerlain et al. 2004, Kellman and Kaiser 1994, Tallal et al. 1998, Chen et al. 2014, Carey 2015], a typical use of PL is through Perceptual Learning Modules (PLMs), multiple-choice questionnaires in which the students are asked to identify which learning pattern is associated to each question. Through the analysis of PLM works [Kellman et al. 2010, Kellman and Kaiser 1994] we propose, as illustrated by Figure 1, that the phases of PLM interventions may be classified into six steps (S1 to S6), although each study might have its own particularities:

- **S1:** identification of the learning patterns related to the area of study;
- **S2:** design of a PLM questionnaire that maps the learning patterns defined in step S1;
- **S3:** participants answer a pretest with questions related to the topic being addressed by the PLM;
- **S4:** assignment of the participants to two groups (experimental and control). Participants from the experimental group answer the PLM questionnaire and participants

from the control group answer the same PLM as the experimental group **or** they carry out a similar or a placebo activity or no activity;

- **S5**: participants from both groups answer a post-test on the PLM topic; it is possible to assess the learning gain when comparing the pretest (step S3) and the post-test;
- **S6**: results from steps S3, S4, and S5 are analyzed and compared;

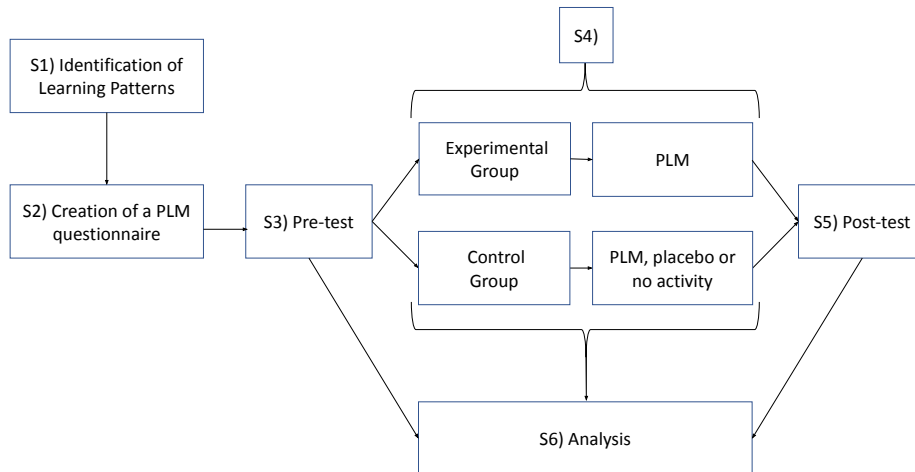


Figure 1: Methodological approach adopted in this study, based on [Kellman et al. 2010, Kellman and Kaiser 1994].

In the next subsections, we describe 3 PLM interventions related to Mathematics. The research steps of each example were classified according to our systematic approach (steps S1 to S7).

2.1.1 Mathematics MultiRep PLM

As explained by [Kellman et al. 2010], the Multiple Representation PLM was designed “to help middle and high school students develop pattern recognition and structure mapping with representations of linear functions, in graphs, equations, and word problems.” The goal was not to ask students to solve problems, but rather to facilitate fluent identification of features and patterns.

In step S1, the researchers initially defined three possible patterns to represent linear equations: a typical *math equation* (e.g. $y = m \cdot x + b$); a *graph*; and the textual description of the equation, defined as *word problem*.

In step S2, the researchers designed a PLM questionnaire with 2 trial blocks and 60 multiple-choice questions per block, for a total of 120 questions. In the questions, students were asked to select, among three possible choices, a representation depicting the same information provided in the question statement. Questions were divided into

four groups: question statement following the *equation* pattern and the available choices as *graphs*; question statement following the *graph* pattern and the available choices as *equations*; question statement as a *word problem* and the available choices as *equations*; and question statement as a *word problem* and the available choices as *graphs*. The possibility to use the *word problem* as a pattern for the available choices was discarded because of the variability of possible correct responses.

In step S3, all participants answered a paper-and-pencil pretest with 12 open-ended questions: 4 questions related to solving word problems and 8 questions involving the translation of a given target (word problem, graph or equation) to a new representation.

In step S4, 68 ninth- and tenth-grade students were divided into experimental and control groups.

Participants in the experimental group answered 2 trials with 60 PLM questions each in a computer program. After each question, they received visual and auditory feedback indicating whether it had been answered correctly, and, if not, what the correct response would be. The intervention took two class periods per day on two consecutive days.

Participants in the control group answered, using paper and pencil, sets of 32 open-ended questions designed to closely resemble the pre- and post-test translation problems. The feedback was provided in the form of an answer key sheet provided to students after each section was completed. The time for this task was similar to that given to the experimental group.

In step S5 all participants answered a paper-and-pencil post-test, with similar – although not the same – questions to the pretest described in step S3.

In step S6, the researchers state that the PLM intervention led to dramatic improvements in the speed and accuracy of equation solving, with the PLM group showing greater improvement than the control group from pretest to post-test. Results indicate that practice in mapping problems across multiple representations (PLM group) led to strong improvements in the *transfer task* problems administered in the pre- and post-tests ($F(1, 66) = 21.27$).

2.1.2 Algebraic Transformations PLM

[Kellman et al. 2010] considered that in a PL approach, it “*should be possible for a student to have relevant declarative and procedural knowledge in some domain and yet lack fluent information extraction skills*”. The authors hypothesized that, in the algebraic topic for middle school, inducing the students’ attention to structure and transformation would lead to an improvement in their ability to identify patterns, which could positively affect problem-solving.

In step S1, authors defined the domain as algebraic equations, specifically of the first degree. Then, in step S2, the PLM was designed as a set of multiple-choice questions that asked students to select, from several choices, the equation that best represented the proper algebraic transformation of a target equation. An example presented by [Kellman and Kaiser 1994] is a target equation defined as $6k + 5x - 17 = 32$, having as one of the wrong transformations the equation $6k - 17 = 32 - x - 5$, and $6k - 17 = 32 - 5x$ as right choice.

In step S3 all participants took a pretest composed of multiple-choice questions (similar to the PLM questions) and problem-solving tasks (open-ended questions that asked students to solve basic algebra equations).

In step S4 there was no organization into experimental and control groups. All participants took the PLM.

Then, in step S5, all participants took the post-test, which is similar in design to the pretest described in step S3.

Finally, in step S6, data from pre- and post-tests were compared. Results indicate that the response time in the solving problems dropped from 26s (pretest) to 12s (post-test) to each problem, in average. Accuracy was high (around 80%) in both tests.

2.1.3 Linear Measurement PLM

[Kellman et al. 2010] explain that students have difficulties to conceive units of linear measurement as having an extent. Also, “*they do not make a clear distinction between position and distance, and they have great difficulty using fractions to represent subdivisions of units*” [Kellman et al. 2010].

In step S1, the authors defined the domain as “linear measurement”.

Then, in step S2, the PLM was designed as a web software that presents to students a graphic display showing a ball on top of a ruler and a billiard cue ready to strike it. The questions (trials) were designed to provide some data (start point, endpoint, distance, etc.) and then ask for a piece of missing information (e.g. the software provides the starting point and the distance, asking for the endpoint). After students entered the answer, the PLM software provided immediate feedback.

In step S3, all participants took a pretest composed of 44 open-ended questions to be solved with pencil and paper. Topics covered by the questions were related to linear measurement with integers and fractions; and adding and subtraction fractions.

Groups were organized in step S4 as follows: the control group was composed of seventh- and eight-grade students (N=86), whereas the experimental group consisted of sixth-grade students (N = 63). Only the participants of the experimental group took the PLM.

In step S5, all participants took the post-test in a similar way as described in step S3.

Finally, in step S6, data from the pre- and post-tests were analyzed. According to [Kellman et al. 2010], the PLM intervention had a positive impact on the experimental group as indicated by an ANOVA analysis comparing control and experimental participants ($F(2, 138) = 19.687, p < .001$).

2.2 MC102 – a CS1 course

The university in which the research was held has a *coordinated* CS1 course in Python, named MC102, mandatory for all STEM students. In the university, the term *coordinated* means that all classes of a given course follow a predefined plan and all students, regardless of section or instructor, are assessed in the same way – that is, take the same or similar exams. The CS1 courses for Computer Science and Computer Engineering are not coordinated.

The MC102 course comprises four weekly hours of classroom activities (mainly expository lectures) and two weekly hours of lab practice. In addition, the students have access to a learning platform (Moodle [Moodle 2013]), in which they carry out Conceptual Activities, multiple-choice questionnaires related to the topics seen in class. Students receive feedback after completing each questionnaire and are allowed to retake them if so desired. These activities are graded based solely on participation per questionnaire – that is, whether the student has attempted each questionnaire at least once – and not on their specific performance.

The number of Conceptual Activities may vary each semester according to overall student performance in the previous term and other external factors. Typically, around

10 Conceptual Activities are made available each semester, respectively covering the following topics: basic types, conditional commands, loops, lists, tuples, dictionaries, functions, and recursion.

MC102 is further described in [Wainer and Xavier, 2018]. An important characteristic is that it started as a C course and, at the time the intervention described herein was executed, the Python version still inherited some of the sequence of presentation of topics from the C version. For example, the topic of functions was only presented after the midterm, as in a C-language teaching context it must come after pointers, despite the fact that in a Python-based course it could have been presented earlier. The interventions explored in this research took place two weeks before the midterm exams for MC102 and the student's grades in the midterm exam were the evaluation metric for the interventions. The midterm exams in MC102 cover questions that involve single loops, lists, complex if-else expressions, and `print` and `read` commands. An example of a midterm can be found in [Gama et al. 2018].

3 Designing a PLM for MC102

Related to the methodological steps S1 and S2 (see Figure 1), for a period of about 3 months, a group of 4 specialists (2 CS1 professors, 1 postdoctoral researcher and 1 undergraduate computer science student) held brainstorming sessions [Lazar et al. 2017] focused on what the most useful learning patterns for the first half of the MC102 course could be.

The researchers first identified 3 main topics (T1 to T3): arrays/lists (T1); loops (T2); and conditionals (T3). In total, 8 learning patterns were associated to these topics ($Q1$), as described in Table 1.

ID	Learning Pattern Description
T1-P1	Iterate over the list until the first element to satisfy a certain property is located.
T1-P2	Append an element to a list.
T1-P3	Copy the contents of a list to another list.
T2-P1	Carry out a preset number of iterations.
T2-P2	Carry out iterations while a certain property is true.
T2-P3	Carry out iterations until a certain situation is identified.
T3-P1	Check whether a number is in a certain interval.
T3-P2	Choose one alternative from a set of options.

Table 1: Learning patterns related to the 3 selected CS1 topics: T1 (arrays/lists); T2 (loops); and T3 (conditionals). The ID indicates the pattern number in determined topic, following the format TY-PX, in which X is a sequential number and Y relates to the topic.

For example, the topic T1 has 3 patterns, labeled T1-P1, T1-P2 and T1-P3.

Considering that the expected resolution time of each question would be around 1 minute, and that the total resolution time of the questionnaire should be around 1 hour, it was defined that the PLM should consist of around 60 questions.

An important aspect of learning the patterns is the syntactic variation of code to achieve the same goal. For instance, a loop that scans a list until a particular condition

is found (T2-P3) could be implemented as a `while` loop, as a `for` loop iterating on an index of the list, a `for` loop iterating on the list's values, and so on. So, many of these syntactic variations for the same patterns were developed.

The available choices in the PLM questions – in addition to the correct one, that is, the correct learning pattern – were initially defined as random possibilities from the available learning patterns (see Table 1). However, during the design process, we identified certain combinations of learning patterns where students might consider multiple options to be correct. For instance, a loop that iterates over an index (pattern T2-P1) is, by definition, constantly verifying whether a number is in a given interval (pattern T3-P1). This led to the definition of a constraints table (see Table 2), used as a guide in question design.

Table 3 shows an example of two PLM questions for the pattern (T2-P3) “Carry out iterations until a certain situation is identified.” The correct answer is c) for the first question and b) for the second. The other answers were randomly selected from the other patterns, with the exception of T1-P1, T2-P1, and T2-P2 – which are answers that cannot be alternatives to a T2-P3 question, as indicated in Table 2.

	T1-P1	T1-P2	T1-P3	T2-P1	T2-P2	T2-P3	T3-P1	T3-P2
T1-P1					X	X		
T1-P2				X	X			X
T1-P3								
T2-P1					X	X	X	
T2-P2						X	X	
T2-P3								
T3-P1								X
T3-P2								

Table 2: Constraints Table. “X” indicates that a given learning pattern is not allowed to be mapped as possible choice for questions of another pattern. For example, for all questions of the T1-P1 pattern, the patterns T2-P2 and T2-P3 are not allowed as possible choices. Diagonal gray cells indicate that all questions from determined pattern must map as correct choice in their own pattern (e.g. all questions from pattern T1-P1 must map as right choice to the pattern T1-P1). Information within light-gray cells was omitted as they mirror the upper white cells.

The full set of 64 questions and answers are available in [Caceffo et al. 2018].

4 Interventions

Two PLM interventions were administered to students. The first intervention was a nonrandom controlled experiment in which students were given the choice whether or not to answer the PLM questionnaire. Those who answered the questionnaire were considered the experimental group, while the control group consisted of students who did not (see Section 4.1). The second intervention was a randomized controlled experiment with a placebo, in which students were randomly assigned to perform either the PLM questionnaire or another a placebo activity (see Section 4.2).

Both interventions, although with differences in their design, share a similar pattern, as follows:

Example 1	Example 2
<p>The code below is an example of:</p> <pre>found = False i = 0 while i < n and not found: if array[i] % 2 == 0: found = True i += 1</pre> <p>(a) Appending an element to a list. (b) Checking whether a number is in a certain interval. (c) Carrying out iterations until a certain situation is identified. (d) Copying the contents of a list to another list.</p>	<p>The code below is an example of:</p> <pre>all_odd = True for a in array: if a % 2 == 0: all_odd = False break</pre> <p>(a) Appending an element to a list. (b) Carrying out iterations until a certain situation is identified. (c) Copying the contents of a list to another list. (d) Choosing one alternative from a set of options.</p>

Table 3: Two examples of the PLM questionnaire for the T2-P3 pattern.

- the PLM intervention was one of the Conceptual Activities, an activity performed using an online system. The activity counted towards the students' final grade, but only on a binary basis – whether or not the student performed the activity – and not related to the student's performance in the questionnaire;
- the PLM questionnaire was the same in both interventions;
- before the PLM intervention, participants were instructed to download and preview a PowerPoint presentation in which the learning patterns were briefly explained;
- the PLM questionnaire was configured to provide immediate feedback after each question was answered;
- the PLM intervention began two weeks before the midterm;
- the PLM questionnaire remained available for 3 days, after which it was closed and made inaccessible to students;
- we used the student's grade in the midterm as the students' outcome;

In terms of the steps described in Figure 1:

- step S1 relates to the learning patterns discussed in Section 3;

- in step S2, the PLM questionnaire designed is available to the readers as discussed in Section 3;
- there was no pretest (step S3);
- step S4, related to how the experimental and control groups were organized, is different in each intervention as discussed in the next sections;
- step S5, related to the post-test, was the midterm exam in both interventions. The exam versions, carefully designed to have a similar level of difficulty, were not exactly equal for all students in a same intervention, as there were slightly differences for the morning, afternoon, and evening CS1 sections to minimize cheating. As discussed in section 2.2, the exams are available to the readers;
- step S6 for each intervention is discussed in the following sections.

4.1 Intervention 1

This intervention was performed in seven coordinated MC102 classes ($N = 761$ students) on the second semester of 2018. Around 2 weeks before the midterm exam, a special Conceptual Activity containing the PLM questionnaire was made available at the online learning platform. All students were invited to participate. Students were informed that the participation was optional.

Two groups were formed: the experimental group, composed of students who took the PLM questionnaire, and the control group, of students who did not (step S4 in Figure 1).

Participants from the control group did not perform any special placebo activity.

4.1.1 Results and Analysis

From all students enrolled in the MC102 coordinate course ($N = 761$), 98 took the PLM Questionnaire, with 45 students answering all questions. We discarded responses from students that scored less than 50% in the PLM questionnaire. The reason for this was that we considered that these students had not learned from the PLM exercise, considering that 50% was also the approve/fail threshold of the MC102 course itself. Five students fell into that category, leaving a total of 40 students in the experimental group. Also, by excluding those students we hoped to exclude the students that had answered the PLM in too short a time, by (likely) guessing the answers and scoring around 25%.

The control group was composed of the students that did not take the PLM questionnaire ($N = 663$). Individuals who missed the midterm exam ($N = 34$) were discarded, and thus this group has a total of 629 students.

The average time registered to answer the PLM questionnaire by the experimental group was 37 minutes ($N = 37$, $sd = 22.1$). Students that saved the session and resumed the questionnaire later ($N = 3$) were not considered in this calculation, as the online system was not able to correctly log the elapsed time.

We will report the difference between the control and experimental group using the effect size measure Cohen's d [Cohen 1988]. Cohen's d is a standardized difference of the means of the experimental and control groups. The standardization is performed by dividing the difference of the means by a "compound" or "pooled" standard deviation of both groups. Formally:

$$d = \frac{\bar{y}_e - \bar{y}_c}{sd_{pool}} \quad (1)$$

where \bar{y}_e is the mean of the outcome measured for the experimental group and \bar{y}_c the mean for the control group. sd_{pool} is the pooled standard deviation of both groups, and is calculated as:

$$sd_{pool} = \sqrt{\frac{(n_c - 1)s_c^2 + (n_e - 1)s_e^2}{n_c - 1 + n_e - 1}} \quad (2)$$

where s_c and s_e are the standard deviations for the control group and experimental group, respectively, and n_c and n_e are the respective sizes of the same groups.

Finally, [Hedges and Olkin 1985] proposed a correction to Cohen's d formula which compensates for an upward bias of d when the sample sizes are small. The corrected formula is usually called Hedge's g . In this paper, our calculations include Hedge's correction, but we will refer to the measurement throughout the paper as *Cohen's d* , which we believe to be the more commonly used term in the Educational literature, and not Hedge's g .

Instead of reporting the p -value of the Cohen's d , we will follow a more modern approach [Gardner and Altman 1986] and report the 95% confidence interval (abbreviated as 95% CI) for the g . If the 95% confidence interval includes the 0 then the reader may conclude that the results are not statistically significant at the 5% level, and therefore that the p -value is higher than 0.05. The computation of the confidence interval uses non-central Student- t distributions as described in [Cumming and Finch 2001]. The computations of the Cohen's d and their confidence intervals in this paper were performed using the function `cohen.d` from the R package `effsize` [Torchiano 2019].

Comparing the midterm grades for both the experimental and control groups in this first intervention yields Cohen's $d = 0.23$, with the 95% confidence interval from -0.08 to 0.55 . This and other results are summarized in Table 4.

The usual interpretation of Cohen's d was proposed by [Coe 2002], who states that effect sizes below 0.2 should be considered negligible, from 0.2 to 0.5 should be considered small, from 0.5 to 0.8 should be considered medium and above 0.8 should be considered large. But within Educational interventions, an effect size of 0.2 should be considered large and important. For example, [Wainer and Xavier, 2018] finds a d of 0.27 and 0.38 on the midterm and final exam outcomes of changing the introductory programming language from C (control) to Python (experimental) for a CS1 course. [Salleh et al. 2010] finds a d of 0.16 on exam grades when comparing pair programming (experimental) with solo programming (control) in a CS1 course; [Freeman et al. 2014][Fig 2.] finds an average d of 0.3 for 8 interventions on using active learning in Computer Science Education. Finally, [Lipsey et al. 2012](cf. Table 9 and Table 10) is a meta-analysis of Educational meta-analyses and reports mean and median effect sizes for educational interventions from elementary school to high school (unfortunately it does not include university-level interventions). All effect sizes reported are below 0.40.

We believe that a Cohen's d of 0.23 is, in fact, a surprisingly large effect size for such a limited intervention.

4.1.2 Dealing with the selection bias - matching

The design for this experiment has a clear threat to validity: the students that received the PLM intervention were self-selected, that is, they chose to receive the intervention,

and thus they could have some specific characteristic that was itself the reason for their better outcome in the midterm exam. The usual technique to analyze data from experiments where there may be a selection bias is to match the students from the control and experimental groups based on the characteristics that one believes are relevant to explain the bias.

We used the following student's data regarding to match each member on the experimental group with one on the control group:

- major – the student's major.
- in phase – whether the student was taking the class at the appropriate moment in their student trajectory - for most majors (but not all) the MC102 course should be taken in the second semester of the student's trajectory
- the year the student entered the University. For all students in phase, they entered the university the same year, but for students out of phase, the year may indicate how close to graduation is the student or how slowly they are progressing through their educational trajectory.
- Student's Standardized Grade Coefficient (SSGC) - is a given student's GPA normalised for their combination of major and admission year.

We used two procedures to match students from the experimental group with students from the control group: pairing on each of the independent variables and propensity score matching.

The **pairing** procedure tries to match each student in the experimental group with one in the control group, provided that both students have the same value for each of the categorical variables (major, in phase, and year) and the value of the numeric variable (SSGC) should be as close as possible. Therefore, each experimental student will be matched (if possible) with a control group student in the same major, who has the same in phase status, who entered the university in the same year, and who has the closest value for their SSGC.

As the procedure may not always find a corresponding match for an experimental group student, unmatched students are dropped out of the experimental group. In this case of this study, only 36 of the 40 students in the experimental group were matched.

The pairing procedure was computed by the `MatchBy` function of the `matching` R package [Sekhon 2011].

In terms of the statistical analysis, one can consider that the purpose of the pairing is to select or filter a subset of the control group that is most similar to the experimental group, in the hope of mitigating the selection bias of the design. In this sense, the statistical computations of the effect size are as described in Equation 1 – we have two sets of measurements (which happen to have the same number of elements) and the Cohen's d is computed as described.

A second alternative is to consider the pairing as a form of actually pairing a student in the experimental group with his or her correspondent in the control group. In this case, the two sets of measures are paired (and they are required to have the same number of elements). The computation of the effect size of paired sets of measurements is somewhat different than described in Equation 1 [Gibbons et al. 1993].

Unfortunately, as far as the authors are aware, there is no clear standard way of performing the effect size calculation after a pairing procedure, so we will report both calculations.

Using the non-paired calculation, the effect size after the pairing is $d = 0.26$, 95% CI $[-0.21, 0.73]$. For the paired calculation, $d = 0.25$, 95% CI $[-0.09, 0.59]$. The results for d are very close, but as expected the confidence interval for the paired calculation is narrower. But none of the results are significant (with 95% confidence).

The **propensity score** matching [Guo and Fraser 2014] technique computes a single number for each student in both groups, and tries to match each student in the experimental group with the one in the control group based only on that number. This single number is the propensity score of the student and it is the result of modeling the probability that the student belongs to the experimental group as a logistic regression of the other independent variables (in this case major, in phase, year, and SSGC). Thus, the propensity score is a measure of how likely the student is to belong to the experimental group based on the other independent variables. The matching finds, for each student in the experimental group, the student in the control group with the closest value for the propensity score.

The propensity score matching was computed using the `matchit` function of the `Matchit` R package [Ho et al. 2007].

As with the pairing procedure, the computations for the effect size may either consider both sets as not-paired or as paired. We will report both results.

For the non-paired computation: $d = 0.34$, 95% CI $[-0.10, 0.78]$; for the paired computation: $d = 0.33$, 95% CI $[-0.06, 0.58]$. Again, both results are similar and the confidence interval for the paired computation is narrower.

4.1.3 Discussion

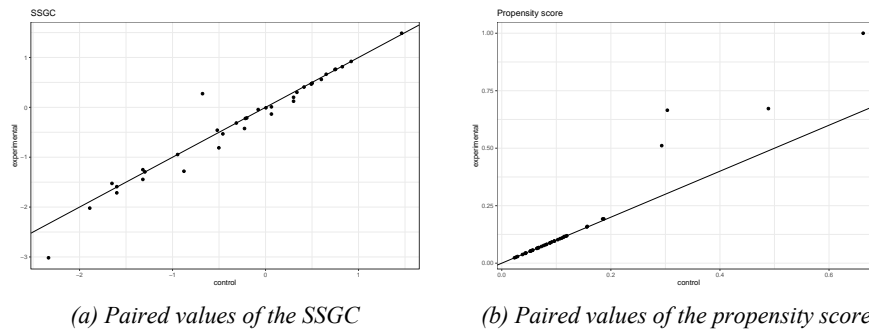
Table 4 contains the values of d for the different analyses discussed above. The table also contains the result for the second intervention discussed below.

Interv.	Experimental Group (N)	Control Group (N)	Random	Matching	d	Paired Analysis	95% CI	Significant
1	40	629	No	No	0.23	No	$-0.08, 0.55$	No
1	36	36	No	Pairing	0.26	No	$-0.21, 0.73$	No
1	36	36	No	Pairing	0.25	Yes	$-0.09, 0.59$	No
1	40	40	No	Propensity	0.34	No	$-0.10, 0.78$	No
1	40	40	No	Propensity	0.33	Yes	$-0.06, 0.58$	No
2	53	52	Yes	No	-0.15	No	$-0.56, 0.26$	No

Table 4: PLM interventions quantitative results.

We considered the results of the first experiment to be promising. As we were aware of the selection bias of the design, we used two different statistical techniques (pairing and propensity score matching) that could compensate for it, and the five results (the raw analysis, the pairing and the propensity score matching, and the analysis using the non-paired and paired calculations for the effect size) were consistent – the effect size was between 0.25 and 0.35, which we consider a large enough effect, but the results were not statistically significant.

There are two alternative interpretations for the non-significant results: either the effect was “real” but the number of subjects was not enough to show significance, or that the effect was not “real” and the figures computed are just “noise”. In other words, a non-significant result may count against the theory, or it may just indicate data insensitivity. Given that two forms of statistical treatment of the selection bias of the experiment yield



(a) Paired values of the SSGC

(b) Paired values of the propensity score

Figure 2: Diagnostics of the matching procedures

similar and positive results, we opted to believe that the effect was “real”, but a different experiment was needed to show it.

4.1.4 Threats to validity

As discussed, the design of Intervention 1 suffers from selection bias, inasmuch as the experimental group’s comparatively better outcome could have been due to characteristics of the group itself, rather than owing to the PLM intervention.

The matching procedures used are an *attempt* to compensate for the selection bias. Despite the fact that is a standard and well-used technique, it is nonetheless not guaranteed that the matching will create a control group that is similar to the experimental group in a way that is relevant to the study. The two groups are similar a) in the values of the categorical independent variables and to close values to the nominal independent variable (for the pairing procedure) or b) on the value of the propensity score which combines all independent variables to predict whether the student is or not in the experimental group for the propensity score matching.

Nonetheless, we can show that the procedure did successfully match the students on those independent variables. Figure 2 contains two scatter plots where the control group (horizontal axis) is compared to the experimental group (vertical axis), each with a $y = x$ graph for reference. The data points on the left-hand pane correspond to the respective SSGC scores for each of 36 pairs of students, whereas those on the right-hand pane show the propensity scores for 40 pairs of students.

The left-hand pane shows that the values of the SSGC are reasonably paired in the sense that they concentrate along the $y = x$ line. However, that claim seems less strong for the propensity score on the right-hand pane – the four largest values of the propensity score are not well-matched among the experimental and control groups. If we remove those 4 pairs from the analysis, the resulting d is 0.12 (95% CI $[-0.35, 0.58]$) for the non-paired computation and 0.12 (95% CI $[-0.24, 0.47]$) for the paired one. In this case, the effect size is smaller than the one including those pairs, but the results are compatible with the previously stated.

Besides the selection bias, this intervention also suffers from a lack of placebo in the control branch. After all, each student in the experimental branch undertook an additional learning activity of about 30 minutes, while those in the control branch were not offered an alternative activity of the same duration. If one group of students devoted half an hour more to preparing for the midterms than the other group, it could be claimed that

any suitable learning activity could produce an improved outcome for the experimental group, regardless of whether it is PLM-based.

Hence, we concluded that even if the PLM intervention did have a real effect on the students' performances, the experiment as carried out using Intervention 1 was not the appropriate means to demonstrate that (Section 4.1.3). In order to account for the lack of placebo – as well as for the aforementioned selection bias – we designed a new intervention, as discussed below.

4.2 Intervention 2

This intervention was carried out in the first semester of 2019, with six coordinated MC102 classes (N = 574 students). Around two weeks before the midterm exam, students were divided into two groups (step S4 in Figure 1): the experimental group, composed of students with an *even* last digit in their student identification numbers and the control group, composed of students with an *odd* final digit in their ID. Each group consisted of 287 students.

All students were invited to participate through a Special Conceptual Activity available on the online platform. This activity was configured in such a way that students from the experimental group had access to the PLM questionnaire, while the control group to a revision activity, composed of a selection of 32 questions from previous versions of the same activity. The placebo questionnaire was designed to take a similar time to the PLM, around 40 minutes. Also, taking the Special Conceptual Activity counted towards the final average of students in either group.

From the 287 students of the experimental group, 124 took the questionnaire, with 62 participants answering all questions. As for the the control group, 122 of 287 students took the questionnaire, with 61 answering all questions.

Experimental group participants were instructed, before taking the questionnaire, to download a PowerPoint presentation in which the learning patterns were briefly presented. The PLM questionnaire was configured to provide immediate feedback after each question was answered. Control group participants only received feedback after completing the placebo activity. In both groups, we removed from the analysis the students that did not achieve 50% of correct answers in their respective questionnaires. The remaining experimental group had 51 students, and the remaining control group had also 51 students. Please note that the fact that the final groups had the same number of students is merely coincidental.

Because of the random assignment of students to the experimental and control group, there is no selection bias in this experimental design, and the two groups can be compared directly. The result is Cohen's $d = -0.15$, 95% CI $[-0.56, 0.26]$ – that is, an effect **against** the PLM intervention. Despite both groups having the same size, there is no meaningful pairing among the students in each group, and thus there is no meaningful paired computation for the effect size.

5 Contributions

As explained earlier in the text (see Section 1 and 2), the PLM has been successfully used in different areas, such as Radiology and Electrocardiography [Thai et al. 2015, Guerlain et al. 2004], flight training [Kellman and Kaiser 1994], language learning [Tallal et al. 1998, Chen et al. 2014] and mathematics [Kellman et al. 2010].

Part of these studies uses as methodology a non-random participant selection model. For example, in the Flight Training study [Kellman and Kaiser 1994] participants were divided into the experimental and control groups, being respectively composed of volunteers and airplane pilots. In the Linear Measurement study [Kellman et al. 2010], participants in the experimental group were students in the 6th grade and the control group was composed of students from the 7th and 8th grades. Both groups, as they represented students from the same school, shared similar socioeconomic criteria.

On the other hand, a randomized selection approach is also possible, being used for example in the Math MultiRep PLM [Kellman et al. 2010], in which participants were divided into control and experiment groups. A third option, as the one employed in the Algebraic Transformations PLM [Kellman et al. 2010] study, does not divide participants into groups, instead comparing the performance of all participants before and after the PLM intervention.

In this way, one of the contributions of this study, concerning the methodology related to PLM interventions, is the use of procedures to combine students from the experimental group with students from the control group to compensate for selection bias in non-randomized controlled experiments.

We described and used *pairing procedures*, which try to match each student in the experimental group with one in the control group, considering for this the profile and background of each student (for our study we used variables like “major”, “in phase”, “year” and the SSGC). We also present and discuss two ways to use pairing procedures: selecting a subset of the control group that is most similar to the experimental group and; pairing a student in the experimental group with his or her correspondent in the control group, discarding students that were not able to be paired.

Also, another pairing procedure adopted was through the *propensity score*. This technique computes a single number – the propensity score – for each student, and then tries to individually match students from a group to a student in the other group, considering how close the propensity score numbers are.

We believe these pairing techniques are a useful way to deal with this specific threat to validity (selection bias), and therefore it can serve as a reference to other researchers when designing further studies in the PLM area.

Another methodological contribution relates to the assessment of the PLM interventions. In the literature, a typical assessment is performed through pre- and post-tests, exams specifically designed to measure the impact of the interventions, like the approaches used in the MultiRep PLM [Kellman et al. 2010] and Algebraic transformations [Kellman et al. 2010] studies. Another possible approach, as the one employed in the Flight Training [Kellman and Kaiser 1994] and Linear Measurement [Kellman et al. 2010] interventions, used the own PLM as a measurement instrument, considering, for example, variables such as speed and response accuracy.

In this way, one difference of our research – and its consequent contribution to the literature – is the fact that the assessment instrument was not designed by the authors, but rather was a real and independent CS1 course exam, in which students should understand and write programming codes. This is a hint that the use of PLM may have a smaller impact in real situations than in the controlled/laboratory environment reported in other literature studies, although further research need to be performed, in the various PLM application areas, to validate this hypothesis.

Finally, as far as we know, this is the first study on automatic recognition of patterns in Introductory Programming (CS1) courses. Although the results point out that the use of PLM does not impact students’ understanding and ability to code, it can be a starting point for the design of other PLM studies related to CS1 and programming. For

example, the learning patterns presented on this study could be expanded, and even used to complement the teaching material in CS1 courses. Also, the intervention itself could be revised, having a longer duration (and possible a greater impact) than the currently presented.

6 Discussion and Conclusion

In this paper, we described two interventions designed to ascertain whether or not the use of Perceptual Learning Modules in CS1 courses has an effect on students' program comprehension, as well as the magnitude of that effect if any exists. The interventions had different experimental designs, were both carried out taking the appropriate precautions to mitigate threats to validity, and yielded opposite results – one with a reasonable positive effect in favor of the use of PLM, the other with a somewhat smaller negative effect, that is, against the use of PLM. Formally, the first intervention was a non-randomized controlled trial with no placebo, while the second was a randomized controlled trial with a placebo. None of the results were significant at the 95% confidence level, but some of them (in particular the results for Intervention 1 that used the paired computation for the effect size) had confidence intervals that almost did not include the 0, and thus were “almost significant”.

We will present two interpretations for the set of results yielded by the experiments described in this paper – one that is more cautious and we deem to be more likely; another that provides for a more consistent analysis of both sets of results, but that we consider to be less probable.

The first and more cautious interpretation is as follows: This paper presents an objective result that the use of Perceptual Learning Modules in CS1 courses as tools for improving program comprehension likely has no effect. Unfortunately, we cannot make any stronger statements than this. Moreover, under this interpretation, we assume that both the large positive and the less large negative results were “noise” from an “underpowered” experiment. The term *underpowered* is used here in quotes, because if we were to assume that the effect is null, then there would be no experiment, no number of subjects (N), that could detect a positive effect!

Notwithstanding, this negative conclusion, as all negative conclusions, must be understood in its specificity. The null result is for our particular PLM intervention, which included the following characteristics:

- consisting of a particular set of patterns, *i.e.*, the perceptual modules,
- carried out as a short half-hour, 60-question intervention,
- conducted as part of a CS1 course that did not place any pedagogical emphasis on program comprehension besides the PLM intervention itself.

It is the sum of these characteristics and statements that we believe to have shown not to have any effect.

The second interpretation is one that allows for a consistent view of both experiments, but which we find less likely than the one put forth above. It hinges on whether the activity given to the control group in Intervention 2 was a placebo at all. Considering that the activity was a compilation of questions taken from previous review activities, it could only serve as a placebo if the students had already solved those questions on a previous occasion. On the other hand, if the questions were new to the students, the

activity served as a meaningful learning activity, and therefore was responsible for the control group's better outcome. Hence, the gain of the PLM in the first intervention was due to the fact that the experimental group spent an extra 30 minutes performing a learning activity while the control group did not, and that gain was lost in the second intervention when the control group did perform a meaningful learning activity of the same duration as that of the experimental group.

We find this interpretation less likely because it does not explain the fact that the difference in learning gains between the experimental group and control group increased between Intervention 1 and Intervention 2, rather than decrease. In this view, 30 minutes of a learning activity against no activity at all would result in a gain of 0.2 to 0.3 standard deviations – as observed in the first intervention – but the review/placebo was such a meaningful activity (assuming that the students had not performed the review activities before) that it changed the effect by 0.3 to 0.4 standard deviations against the PLM in the second intervention. For that reason, we believe our first interpretation as stated above to be more likely.

Furthermore, there is a second aspect of this study that we believe is of interest to a researcher in computer education in general – the methodological decisions and techniques used.

The use of propensity score matching, or other forms of matching, to compensate for selection bias on non-randomized controlled experiments is uncommon in Computer Science Education (CSE) papers. We know of only three other papers that used the technique [Peteranetz et al. 2018, Peteranetz et al. 2018a, Stout et al. 2018], two of them from the same research group. Since non-randomized experiments are a very common experimental design in Educational interventions – particularly in experiments in CSE – the community should use matching techniques, such as pairing and propensity score matching, more frequently. In particular, the two R packages mentioned above, `Matchit` [Ho et al. 2007] and `matching` [Sekhon 2011], contain a powerful set of functions to apply matching techniques and to perform the diagnostics of the resulting match.

Regarding randomized experiments in CSE, although there has been an increase of such experiments in recent years ([Cao and Porter 2017, Patistas et al. 2013, Denny 2015, Vihavainen et al. 2015, Bezakova et al. 2014, Leinonen et al. 2019, Yuan and Cao 2019, Nelson et al. 2017], to cite a few recent papers within the CS1/CS2 context), randomized experiments are still not very common within CSE. We note that using an online tool as part of the standard activities in a CS1 course is an unparalleled opportunity for randomized experiments.

Finally, the methodological warning mentioned in the title of this paper refers to two questions we had to face in this research and that we believe other researchers may face in different lines of research. The first question is what to do with a non-significant intermediary result. Does the positive, non-significant effect indicate that one should drop the research line because the positive result is noise generated by the non-significance? Or should the researcher believe that the effect is real and that the non-significant result is just an accident of the low number of subjects in the experiment? While on the one hand, the significance tests were designed so that researchers and readers would not be misled by noisy results, on the other hand, the threshold of significance ($p \leq 0.05$) is arbitrary and thus an unknown number of true results may never been published because of this.

Should the researcher decide to repeat or redesign the experiment, they may also face the second methodological problem: what to conclude if both experiments indicate conflicting effects. Although randomized experiments are considered “a better design” than non-randomized experiments, we believe that there is no reason to only use its

results as the final conclusion of this line of research. There has been some empirical literature in the health field that show that despite the theoretical disadvantages of non-randomized experiments, the results of comparing them with randomized experiments are well correlated [Benson and Hartz 2000, Concato et al. 2000]. There is no evidence to believe that our second intervention was so more “valid” than the first that one should disregard the evidence of the latter. Thus, we believe that one should just report the contradiction of the results. In our case, the non-significance of all the intermediary results suggest that the effect is indeed null, and although perhaps less exciting, null results should be published [Rosenthal 1979] nonetheless.

Future work involves the adoption of PLMs in conjunction with Concept Inventories (CIs), assisting in mitigating misconceptions [Caceffo et al. 2016, Caceffo et al. 2019] as well as joint adoption of active learning techniques in CS1 courses [Caceffo et al. 2018a].

We gratefully acknowledge the supporters of this research: São Paulo Research Foundation (FAPESP), grants #2013/08293-7 and #2014/07502-4; Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES); the National Council of Technological and Scientific Development (CNPq); and the State University of Campinas (UNICAMP).

References

- [Benson and Hartz 2000] Kjell Benson and Arthur J Hartz. A comparison of observational studies and randomized, controlled trials. *New England Journal of Medicine*, 342(25):1878–1886, 2000.
- [Bezakova et al. 2014] Ivona Bezakova, James Heliotis, and Sean Strout. On the efficacy of board game strategy development as a first-year cs project. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 283–288. ACM, 2014.
- [Caceffo et al. 2016] Ricardo Caceffo, Steve Wolfman, Kellogg S. Booth, and Rodolfo Azevedo. Developing a computer science concept inventory for introductory programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE ’16*, page 364–369, New York, NY, USA, 2016. Association for Computing Machinery.
- [Caceffo et al. 2018] Ricardo Caceffo, Guilherme Gama, Jacques Wainer, Islene Garcia and Rodolfo Azevedo (2018). A questionnaire of perceptual learning modules (PLM) for introductory programming (CS1) courses in Python. Technical Report 18-11, Institute of Computing, University of Campinas, SP, Brazil, 46 pages.
- [Caceffo et al. 2018a] Ricardo Caceffo, Guilherme Gama, and Rodolfo Azevedo. Exploring active learning approaches to computer science classes. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE ’18*, pages 922–927, New York, NY, USA, 2018. ACM.
- [Caceffo et al. 2019] Ricardo Caceffo, Pablo Frank-Bolton, Renan Souza, and Rodolfo Azevedo. Identifying and validating java misconceptions toward a cs1 concept inventory. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE ’19*, page 23–29, New York, NY, USA, 2019. Association for Computing Machinery.
- [Cao and Porter 2017] Yingjun Cao and Leo Porter. Evaluating student learning from collaborative group tests in introductory computing. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 99–104. ACM, 2017.
- [Carey 2015] Benedict Carey. Learning to see data. In *New York Times*, 2015.
- [Chase and Simon 1973] William G. Chase and Herbert A. Simon. Perception in chess. *Cognitive Psychology*, 4(1):55 – 81, 1973.

- [Chen et al. 2014] Jiawei Chen, Yan Liu, Xiaomeng Li, and Liujun Chen. Perceptual learning model on recognizing chinese characters. In Zhigang Zeng, Yangmin Li, and Irwin King, editors, *Advances in Neural Networks – ISNN 2014*, pages 243–251, Cham, 2014. Springer International Publishing.
- [Coe 2002] Robert Coe. It’s the effect size, stupid: What effect size is and why it is important, 09 2002.
- [Cohen 1988] Jacob Cohen. *Statistical power analysis for the social sciences*. Erlbaum, 1988.
- [Concato et al. 2000] John Concato, Nirav Shah, and Ralph I Horwitz. Randomized, controlled trials, observational studies, and the hierarchy of research designs. *New England Journal of Medicine*, 342(25):1887–1892, 2000.
- [Cumming and Finch 2001] Geoff Cumming and Sue Finch. A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and psychological measurement*, 61(4):532–574, 2001.
- [Denny 2015] Paul Denny. Generating practice questions as a preparation strategy for introductory programming exams. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 278–283. ACM, 2015.
- [Freeman et al. 2014] Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23):8410–8415, 2014.
- [Gama et al. 2018] Guilherme Gama, Ricardo Caceffo, Renan Souza, Raysa Benati, Tania Aparecida, Islene Garcia and Rodolfo Azevedo (2018). An antipattern documentation about misconceptions related to an introductory programming course in Python. Technical Report 18-19, Institute of Computing, University of Campinas, SP, Brazil, 106 pages.
- [Gardner and Altman 1986] Martin J Gardner and Douglas G Altman. Confidence intervals rather than p values: estimation rather than hypothesis testing. *Br Med J (Clin Res Ed)*, 292(6522):746–750, 1986.
- [Gibbons et al. 1993] Robert D Gibbons, Donald R Hedeker, and John M Davis. Estimation of effect size from a series of experiments involving paired comparisons. *Journal of Educational Statistics*, 18(3):271–279, 1993.
- [Guerlain et al. 2004] S. Guerlain, Kristen Brook Green, M. LaFollette, T. C. Mersch, B. A. Mitchell, G. R. Poole, J. F. Calland, Jianhong Lv, and E. G. Chekan. Improving surgical pattern recognition through repetitive viewing of video clips. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(6):699–707, Nov 2004.
- [Guo and Fraser 2014] Shenyang Guo and Mark W Fraser. *Propensity score analysis: Statistical methods and applications*. SAGE publications, 2014.
- [Harrington and Cheng, 2018] Brian Harrington and Nick Cheng. Tracing vs. writing code: beyond the learning hierarchy. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 423–428, 2018.
- [Hedges and Olkin 1985] Larry V Hedges and Ingram Olkin. *Statistical methods for meta-analysis*. Academic Press, 1985.
- [Ho et al. 2007] Daniel E Ho, Kosuke Imai, Gary King, and Elizabeth A Stuart. Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political analysis*, 15(3):199–236, 2007.
- [Kellman et al. 2010] Christine M. Kellman, Philip J. an d Massey and Ji Y. Son. Perceptual learning modules in mathematics: Enhancing students’ pattern recognition, structure extraction, and fluency. *Topics in Cognitive Science*, 2(2):285–305, 2010.

- [Kellman and Kaiser 1994] Philip J. Kellman and Mary K. Kaiser. Perceptual learning modules in flight training. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 38(18):1183–1187, 1994.
- [Lazar et al. 2017] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research Methods in Human Computer Interaction (Second Edition)*. Morgan Kaufmann, 2017.
- [Leinonen et al. 2019] Antti Leinonen, Henrik Nygren, Nea Pirttinen, Arto Hellas, and Juho Leinonen. Exploring the applicability of simple syntax writing practice for learning programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 84–90. ACM, 2019.
- [Lipsey et al. 2012] Mark W Lipsey, Kelly Puzio, Cathy Yun, Michael A Hebert, Kasia Steinka-Fry, Mikel W Cole, Megan Roberts, Karen S Anthony, and Matthew D Busick. *Translating the Statistical Representation of the Effects of Education Interventions into More Readily Interpretable Forms*. National Center for Special Education Research, 2012.
- [Lister et al. 2009] Raymond Lister, Colin Fidge, and Donna Teague. Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *ACM SIGCSE bulletin*, 41(3):161–165, 2009.
- [Lopez et al. 2008] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the fourth international workshop on computing education research*, pages 101–112, 2008.
- [Moodle 2013] Moodle System. Moodle: Open-source learning platform, 2013. Available at: <https://moodle.org/> Accessed: September 2022
- [Morrison et al. 2015] B. B. Morrison, L. E. Margulieux, and M. Guzdial. Subgoals, context, and worked examples in learning computing problem solving. In *ICER 2015 - Proceedings of the 2015 ACM Conference on International Computing Education Research*, pages 21–30, 2015.
- [Muller and Haberman 2008] Orna Muller and Bruria Haberman. Supporting abstraction processes in problem solving through pattern-oriented instruction. *Computer Science Education*, 18(3):187–212, 2008.
- [Murphy et al. 2012] Laurie Murphy, Sue Fitzgerald, Raymond Lister, and Renée McCauley. Ability to ‘explain in plain english’ linked to proficiency in computer-based programming. In *Proceedings of the ninth annual international conference on International computing education research*, pages 111–118, 2012.
- [Nelson et al. 2017] Greg L. Nelson, Benjamin Xie, and Andrew J. Ko. Comprehension first: Evaluating a novel pedagogy and tutoring system for program tracing in cs1. In *Proceedings of the 2017 ACM Conference on International Computing Education Research, ICER ’17*, page 2–11, New York, NY, USA, 2017. Association for Computing Machinery.
- [Patistas et al. 2013] Elizabeth Patistas, Michelle Craig, and Steve Easterbrook. Comparing and contrasting different algorithms leads to increased student learning. In *Proceedings of the ninth annual international ACM conference on International computing education research*, pages 145–152. ACM, 2013.
- [Peteranetz et al. 2018] Markeya S Peteranetz, Abraham E Flanigan, Duane F Shell, and Leen-Kiat Soh. Helping engineering students learn in introductory computer science (cs1) using computational creativity exercises (cces). *IEEE Transactions on Education*, 61(3):195–203, 2018.
- [Peteranetz et al. 2018a] Markeya S Peteranetz, Shiyuan Wang, Duane F Shell, Abraham E Flanigan, and Leen-Kiat Soh. Examining the impact of computational creativity exercises on college computer science students’ learning, achievement, self-efficacy, and creativity. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 155–160. ACM, 2018.
- [Rosenthal 1979] Robert Rosenthal. The file drawer problem and tolerance for null results. *Psychological bulletin*, 86(3):638, 1979.

- [Salleh et al. 2010] Norsaremah Salleh, Emilia Mendes, and John Grundy. Empirical studies of pair programming for cs/se teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering*, 37(4):509–525, 2010.
- [Sekhon 2011] Jasjeet S. Sekhon. Multivariate and propensity score matching software with automated balance optimization: The matching package for r. *Journal of Statistical Software*, 42(7):1–52, 2011.
- [Skudder and Luxton-Reilly 2014] Ben Skudder and Andrew Luxton-Reilly. Worked examples in computer science. In *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148*, pages 59–64, 2014.
- [Stout et al. 2018] Jane G Stout, N Burçin Tamer, and Christine J Alvarado. Formal research experiences for first year students: A key to greater diversity in computing? In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 693–698. ACM, 2018.
- [Tallal et al. 1998] Paula Tallal, Michael Merzenich, Steve Miller, and William Jenkins. Language learning impairment: Integrating research and remediation. *Scandinavian Journal of Psychology*, 39(3):197–199, 1998.
- [Thai et al. 2015] Khanh-Phuong Thai, Sally Krasne, and Philip J. Kellman. Adaptive perceptual learning in electrocardiography: The synergy of passive and active classification. In *CogSci*, 2015.
- [Torchiano 2019] Marco Torchiano. *effsize: Efficient Effect Size Computation*, 2019. R package version 0.7.6.
- [Van Merriënboer and Paas 1990] Jeroen JG Van Merriënboer and Fred GWC Paas. Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in human behavior*, 6(3):273–289, 1990.
- [Venables et al. 2009] Anne Venables, Grace Tan, and Raymond Lister. A closer look at tracing, explaining and code writing skills in the novice programmer. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop, ICER '09*, page 117–128, New York, NY, USA, 2009. Association for Computing Machinery.
- [Vihavainen et al. 2015] Arto Vihavainen, Craig S Miller, and Amber Settle. Benefits of self-explanation in introductory programming. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 284–289. ACM, 2015.
- [Wainer and Xavier, 2018] Jacques Wainer and Eduardo C Xavier. A controlled experiment on python vs c for an introductory programming course: Students’ outcomes. *ACM Transactions on Computing Education (TOCE)*, 18(3):12, 2018.
- [Xie et al. 2019] Benjamin Xie, Dastyni Loksa, Greg L Nelson, Matthew J Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, and Andrew J Ko. A theory of instruction for introductory programming skills. *Computer Science Education*, 29(2-3):205–253, 2019.
- [Yuan and Cao 2019] Hans Yuan and Yingjun Cao. Hybrid pair programming—a promising alternative to standard pair programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 1046–1052. ACM, 2019.