Journal of Experimental Child Psychology 216 (2022) 105335

Contents lists available at ScienceDirect

Journal of Experimental Child Psychology

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

Peer tutoring of computer programming increases exploratory behavior in children

Diego P. de la Hera [a,b], María B. Zanoni [a,b], Mariano Sigman [a,b], Cecilia I. Calero [a,b,*,1]

[a] Laboratorio de Neurociencia, Universidad Torcuato Di Tella (UTDT), C1428 Buenos Aires, Argentina
[b] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), C1425 Buenos Aires, Argentina

ARTICLE INFO

Article history:
Received 21 January 2021
Revised 27 September 2021
Available online 30 December 2021

Keywords:
Computer programming
Teaching
Peer tutoring
Exploratory behavior
Learning by teaching
K-12 education

ABSTRACT

There is growing interest in teaching computer science and programming skills in schools. Here we investigated the efficacy of peer tutoring, which is known to be a useful educational resource in other domains but never before has been examined in such a core aspect of applied logical thinking in children. We compared (a) how children (N = 42, age range = 7 years 1 month to 8 years 4 months) learn computer programming from an adult versus learning from a peer and (b) the effect of teaching a peer versus simply revising what has been learned. Our results indicate that children taught by a peer showed comparable overall performance—a combination of accuracy and response times—to their classmates taught by an adult. However, there was a speed–accuracy trade-off, and peer-taught children showed more exploratory behavior, with shorter response times at the expense of lower accuracy. In contrast, no tutor effects (i.e., resulting from teaching a peer) were found. Thus, our results provide empirical evidence in support of peer tutoring as a way to help teach computer programming to children. This could contribute to the promotion of a widespread understanding of how computers operate and how to shape them, which is essential to our values of democracy, plurality, and freedom.

* Corresponding author.
E-mail address: ccalero@utdt.edu (C.I. Calero).
[1] Current address: Área de Educación, Escuela de Gobierno, Universidad Torcuato Di Tella (UTDT), C1428 Buenos Aires, Argentina.

https://doi.org/10.1016/j.jecp.2021.105335
0022-0965/

Introduction

Computer programming is of utmost relevance in our current digital world. Children, although digital natives, do not learn these skills by themselves, and schools are increasingly interested in teaching them. However, programming education has its difficulties, and alternatives are worth considering. In this study we examined peer tutoring as one such alternative to help teach programming languages to children.

Computer programming in a digital world

Today's world is flooded by digital technologies. Computers surround us in permanent and increasing communication among them. This ever growing computer network is deeply intertwined with our social network and underlies many aspects of our societies today (van Dijk, 2010; Fundación Sadosky, 2016; President's Council of Advisors on Science and Technology, 2010). For example, the internet, our largest computer network, which had 1.1 billion users in 2005, is currently used by more than 4 billion people, comprising more than half the world population (International Telecommunications Union [ITU], 2021). On the other hand, employment in the information technology sector increased 30% from 2015 to 2020 in Argentina (Observatorio Permanente de la Industria del Software y Servicios Informáticos [OPSSI], 2021) and increased from 2.6% of the country's total employment to 3.3% from 2010 to 2017 in Germany (International Labour Organization [ILO], 2020), for example. With computers at the foundations of our worldwide society, understanding how they operate and how to shape them turns us from passive consumers into active actors (Resnick et al., 2009) and is fundamental for freedom (Söderberg, 2012). Computer programming is the way we humans communicate with these machines and the networks they belong to and control what they do. It consists in designing and building sets of instructions interpreted by a computer to accomplish a specific task (Dalbey & Linn, 1985; Fundación Sadosky, 2016; Papert, 1980). There is a variety of languages for computer programming, and understanding them is becoming increasingly important (Fedorenko, Ivanova, Dhamala, & Bers, 2019; Fundación Sadosky, 2016). In addition, not only does learning to code improve computer programming skills, but there also is evidence from prekindergarten to university that it transfers to other general cognitive skills as well such as creativity, spatial cognition, mathematical skills, and reasoning (Fundación Sadosky, 2016; Resnick et al., 2009; Scherer, 2016; Scherer, Siddiq, & Sánchez Viveros, 2019).

Programming education

Children become software and hardware consumers at increasingly earlier ages (Rideout & Robb, 2020). However, being digital natives is not necessarily accompanied by learning how these technologies work (Martinez, Gomez, & Benotti, 2015). As a result, there is growing interest in integrating programming courses at all educational levels (Fessakis, Gouli, & Mavroudi, 2013; Gülbahar & Kalelioğlu, 2014). It has even been considered a new literacy just like reading or math (Fedorenko et al., 2019; O'Reilly, 1998; Voogt, Fisser, Good, Mishra, & Yadav, 2015; Wing, 2006), and many schools, districts, and countries have acknowledged the importance of including computer programming in their curricula (Balanskat & Engelhardt, 2015; Brinda, Puhlmann, & Schulte, 2009; Brown et al., 2013; Consejo Federal de Educación, 2018; Fundación Sadosky, 2016; Kalelioğlu, 2015). For example, in the United States, 20 states had established a requirement for all high schools to offer computer science as of 2020 compared with only 4 states in 2017 (Code.org, Computer Science Teachers Association [CSTA], & Expanding Computing Education Pathways [ECEP] Alliance, 2020). But teaching children this new literacy is not an easy endeavor. Recently, new programming languages have emerged that creatively managed to reduce the syntactic and semantic difficulties of traditional ones (Wyeth, 2008) to the point of becoming completely visual. These visual languages let users manipulate program elements graphically rather than specifying them textually and thus are particularly suitable for children (Maloney, Resnick, & Rusk, 2010; Resnick et al., 2009). However, teaching to code still poses some

difficulties on its own. A variability of teaching methods without a clear understanding of which are useful in what contexts, a lack of teaching standards and of trained teachers (Fundación Sadosky, 2016), and a long tradition of information technology classes that focus on soft computer skills such as word processors, spreadsheets, and other office tools (Benotti, Martínez, & Schapachnik, 2014; Fundación Sadosky, 2016; Resnick et al., 2009) are some examples of these difficulties. These may be especially relevant for girls given that many studies have described that girls may have lower performance and interest in technical domains than boys (Edwards, Coddington, & Caterina, 1997; Hill, Corbett, & St. Rose, 2010; Nourbakhsh, Hamner, Crowley, & Wilkinson, 2004).

Collaborative learning

There is no perfect recipe for learning a new skill. For example, although learning from a knowledgeable teacher in a traditional setting does promote rapid and efficient learning, it may also present some drawbacks such as limiting spontaneous exploration and discovery (Bonawitz et al., 2011). Hence, and considering the difficulties of teaching to code, it may be worth examining alternatives to traditional teaching practices. One such alternative is learning from a peer, or peer tutoring, which has been shown to benefit tutees as well as tutors (Cohen, Kulik, & Kulik, 1982; Roscoe & Chi, 2008; Topping, 2015) in a variety of knowledge domains (de la Hera, Sigman, & Calero, 2019; Topping, 2017), and it is among the most cost-effective teaching strategies (Levin, Glass, & Meister, 1987). However, research in peer tutoring is complex. Results are highly variable, and they depend on a multitude of factors such as the targeted subject matter, the age of the participants, and the availability and frequency of tutor training, among many others (Bowman-Perrott et al., 2013; Leung, 2015). For instance, a 2006 experimental study (Shamir, Tzuriel, & Rozen, 2006) assessed peer tutoring effects on mathematical skills for tutors and tutees. The study was cross-age (third-grade tutors and second-grade tutees) and compared tutor–tutee dyads where the tutor had received specific tutor training versus dyads where they had not. Gains were found for both tutors and tutees, but only when tutors had received tutor training. In addition, most studies have focused on math and reading, and although peer tutoring has proved to be useful across all educational levels, the largest effects are usually found in secondary school (Leung, 2015).

Collaborative programming

Collaborative approaches have also been considered by the software industry. Pair programming, for example, is a practice that involves two developers collaborating as a single individual on the design, coding, and testing of the same programming task. This practice has been shown to be productive and to render higher-quality code than either developer may produce alone (Lui & Chan, 2006; Nosek, 1998; Williams, 2001; Williams, Kessler, Cunningham, & Jeffries, 2000; Williams & Upchurch, 2001), especially with novice programmers or challenging programming problems (Lui & Chan, 2006), and positive effects have been found in educational contexts as well (Braught, Wahls, & Eby, 2011; Tunga & Tokel, 2018; Umapathy & Ritzhaupt, 2017). Another popular practice involves explaining line by line what a nonworking piece of code is supposed to do. This forces the programmer to clarify and structure the explanation, helping him or her to find where the cause of the problem is in a fashion similar to how explanation and teaching are proposed to foster learning and understanding (Duran, 2017; Legare & Lombrozo, 2014; Roscoe & Chi, 2008). This technique is usually referred to as *rubber duck debugging* given that it is known to work even when the code is explained to an inanimate agent such as a rubber duck (Hunt & Thomas, 1999; Phillips, Lockton, Baurley, & Silve, 2013).

A gap in the literature

Although previous studies have examined collaborative learning in programming education (Silva, Mendes, & Gomes, 2020), no study has addressed peer tutoring of programming languages in children and how it can help to tackle some of the difficulties of teaching to code. This is particularly surprising given the increasing importance attributed to learning to code, the growing interest in teaching computer programming from elementary school or even kindergarten, the recognized effectiveness of peer

tutoring in a variety of fields, and the experience from the software industry. Peer tutoring is particularly appealing in this field given that children are digital natives and sometimes even outperform some of their teachers in this subject matter (Benotti et al., 2014). Some studies have come close to the topic, but no study has made it its focus of attention. For example, some studies have addressed attainment of basic computer literacy by unsupervised groups of young children, but they have not covered computer programming (Mitra et al., 2005). On the other hand, there are a few studies about peer tutoring and peer assessment of computer programming, but they have focused on older (secondary school and college) students (Altintas, Gunes, & Sayan, 2016; Golding, Facey-Shaw, & Tennant, 2006) and some of them are exploratory rather than experimental (Edwards et al., 1997).

The current study

In this study, we addressed peer tutoring as a way to help teach programming languages to children. To this end, we used a custom adaptation of a programming platform designed for them. First, we studied whether children can teach other children to program by comparing coding skills and resolution strategies developed by (a) children who were taught by an adult and (b) children who learned from a peer. On the other hand, we also looked into whether children benefit from teaching others by comparing (a) children who taught a naive peer with (b) children who revised the instruction materials alone.

Based on previous research on peer tutoring and peer programming practices, we expected that children taught by their peers would perform better than children not taught at all and would perform as well as, or even better than, children taught by an adult in a group class. We also expected that children who taught their peers would perform better than children who revised the instruction materials alone on both immediate and delayed reevaluations. In addition, although there is some evidence that boys perform better than girls in STEM (science, technology, engineering, and mathematics), these differences would largely be due to social convention stereotypes (e.g., see Beilock, Gunderson, Ramirez, & Levine, 2010; Dar-Nimrod & Heine, 2006) and we expected that children in our study might be too young to have been affected by them. Performance was assessed using a combination of accuracy and response times, as is frequently done in school tests, but these variables were considered separately as well.

Method

Participants

A total of 44 second graders (28 female) participated in the study. All were authorized children from two second grades (whose teachers agreed to participate) in a medium- to high-SES (socioeconomic status) bilingual school in Buenos Aires, Argentina, with which we had worked in the past. Data from only 2 children (both female) needed to be eliminated (1 child because her test session needed to be interrupted and the other child because she had difficulties in understanding the game). Ages of the 42 remaining children (26 female) ranged from 7 years 1 month to 8 years 4 months, with a mean age of 7 years 10 months. Sessions took place outside of the classroom in a quiet room provided by the school for the purpose of this study.

All children's parents or legal guardians gave signed voluntary consent previously authorized by an ethical committee.

Programming platform

The LEGO Education WeDo set (Mayerová & Veselovská, 2014) was used, marketed for children from 7 years of age (Kazakoff, Sullivan, & Bers, 2013) and already used in previous studies with children around this age (Mayerové & Veselovská, 2017; Pinto-Llorente, Casillas-Martín, Cabezas-González, & García-Peñalvo, 2018). This programming platform comprises (a) a programming language, (b) a program editor, (c) an interpreter, and (d) a programmable environment. The programming

language is a visual icon-based language similar to the Massachusetts Institute of Technology's (MIT) Scratch Jr. (Flannery et al., 2013). It includes a series of blocks, each conveying a different command. In the program editor, blocks are dragged from the blocks' panel onto the editor's canvas and are snapped one next to the other to build block sequences or programs (Fig. 1A). These programs are read by the interpreter to operate on the platform's programmable environment, which is composed of (a) a virtual screen located near the top left corner of the computer's screen, (b) the computer's speakers, and (c) a LEGO-brick physical robot built by us with a rotating arm, resembling a clock facing upward.

The programming language can be understood as operating on five different variables that describe the state of the programmable environment at any one moment. First, the image (img) variable describes the image content of the environment's screen, and it can take values 1–20, each representing a different image, or 0 for a white/empty image. Second, the text (txt) variable describes the text content of the environment's screen, and it can take any string value given by the user. Third, the position (pos) variable describes the position of the robot's arm in degrees, and it can take values 0 to 360. Fourth, the rotation (rot) variable describes whether the robot's arm is rotating and in which direction, and it can take values +1, −1, or 0 for clockwise, counterclockwise, and no rotation, respectively. Fifth,

Fig. 1. Methods. (A) Screen capture. A screen capture shows the programming platform, with a block sequence built on the editor's canvas and the software environment in test mode on Trial 2, with the trial video paused at 3 s (see "Programming platform" and "User interface" sections for full details). The editor's canvas was shrunk to optimize presentation. (1) Program editor blocks' panel. (2) Program editor's canvas. (3) Programmable environment screen with img = 11 and txt="" (empty string). (4) Video player screen paused during playback; current trial number is shown at the top ("JUEGO 2": Trial 2). (5) Play/pause button showing as play because video is paused. (6) Reload button. (7) Next trial button. (8) Previous trial button. (9) Reset programmable environment button. (B) Example trial video. Top: Four frames from an example trial video are shown. The playback time (in seconds) is shown in the bottom left corner of each frame. Each frame represents a different state of the programmable environment, described by the value of five variables (see "Programming platform" and "Trial videos" sections for full details), shown here in a square in the bottom right corner of each frame (img, image; txt, text; pos, position; rot, rotation; snd, sound; see main text). The playback time and the variables' square do not appear in the real videos. Bottom: Beneath the video frames, the block sequence or program that recreates the series of events depicted in the video is shown. Duck clip art designed by brgfx/Freepik.

the sound (snd) variable describes the sound being played by the computer's speakers, and it can take values 1–20, each representing a different sound, or 0 for no sound.

The programmable environment is initialized with state [img = 0, txt="" (empty string), pos = 0, rot = 0, snd = 0] for all trials. A two-sided cardboard catalog was printed, with all images available and their corresponding values (including 0 for white/empty image) on one side and all sounds (represented with icons) and their values on the other side (see online Supplementary Methods 1).

Block sequences (programs) are built on the editor's canvas by attaching command blocks one next to the other. Eight command blocks were available in this study; some of them (transitive) accept an additional argument block attached below to further define the command (see Supplementary Methods 2). The interpreter reads block sequences from left to right and updates the state of the programmable environment accordingly (Fig. 1B). For example, the image block updates the img variable when read by the interpreter. Changes in the programmable environment's state are referred to as *events.*

User interface

For this study, we developed a user interface laying between the user and the WeDo programming platform. Its purpose was to avoid undesired interactions with the platform and to provide an adequate frame, including a series of exercises or trials to evaluate participants' learning of the programming language (Fig. 1A).

Regarding control of user interaction, to simplify the programming language, a limited subset of blocks from the editor's block panel was made available, whereas the rest of the blocks were concealed and disabled behind the overlay interface. In addition, other undesired user interactions with the programming platform and with the computer's operating system were blocked such as closing the program editor's block panel, the programmable environment's screen, or the programming platform altogether, adjusting the computer's volume, and turning the computer off, among many others.

On the other hand, to provide an adequate evaluation frame, we also included a video player and a trial navigator (Fig. 1A). The video player showed one video per trial, each presenting a series of events (changes in the programmable environment's state), which children needed to reproduce programmatically by building block sequences (see Fig. 1B and "Trial videos" section). The video player provided a play/pause button to start, pause, and resume playback as many times as desired and provided a reload button to restart playback from the beginning. The trial navigator included next and previous buttons to go back and forth through the trials as desired. Each time one trial was accessed through the trial navigator, it was counted as one attempt for that trial, and block sequences placed on the editor's canvas were remembered between attempts of the same trial. Participants could reinitialize the programmable environment's state at any time with a reset button (Fig. 1A).

Because participants could go to the next trial at any moment, regardless of their performance and without getting any feedback, a self-assessment screen was included to discourage compulsive "next clicking." For each trial's first attempt, when the next trial button was clicked, the user was presented with the self-assessment screen for that trial. It asked the user if he or she was able to successfully copy the video of the trial that had just been attempted and provided three buttons, each with a traffic light icon: "yes" (green light), "so so" (yellow light), and "no" (red light). If the user responded "yes," the self-assessment screen was closed and the next trial (if any) was presented. If the user responded "so so" or "no," the user was asked to confirm whether he or she wanted to proceed to the next trial or return to the previous trial instead.

The overlay user interface could be run in one of three modes: *introduction* and *training* modes, used in the instruction and peer-tutoring/self-revision stages; and *test* or *retest* modes, used in the test and retest stages (see "Procedure" section below; see also Supplementary Methods 2).

Trial videos

Trial videos show a series of *events,* each involving one or more of the img, txt, pos, rot, and snd variables, which participants needed to reproduce by building block sequences or programs (Fig. 1B). Trial videos contain (a) a copy of the programmable environment's screen in the top left cor-

ner to indicate changes in the img and txt variables; (b) an area in the top right corner where the icons that represent sounds may appear, accompanied by their corresponding sounds, to indicate changes in the snd variable; and (c) a top-view representation of the robot and its rotating arm to indicate changes in the pos and rot variables. All videos start with the programmable environment's initial state [img = 0, txt="", pos = 0, rot = 0, snd = 0]; the first event occurs at 1 s and the remaining events occur every 2 s (see Fig. 1B for an example; full videos are available at https://gitlab.com/diegodlh/lego-wedo/-/tree/master/videos).

In test and retest stages, 10 videos were presented in the same order to all participants, showing increasingly complex series of events (see Supplementary Methods 3). Videos in retest were isomorphic to videos in test, meaning that they showed events involving the same variables but with different values, to promote engagement (Smith et al., 2009). For example, if the second event in Test Video 3 involved changing img from 0 to 5 (image of the woods), the second event in Retest Video 3 involved a change in img as well but from 0 to 7 (image of a desert) instead.

Procedure

Children participated in three or five stages (see Supplementary Methods 4).

Instruction stage

First, two thirds of the participants in each class were assigned to the adult-taught condition (n = 27). In same-class mixed-gender groups of 4 or 5 children, they were given a 20-min tightly scripted lesson by an adult instructor. Children sat at the table one next to the other facing a notebook's screen, with the instructor standing to their left. The lesson consisted of two parts. In the first part, the basics of the programming language were taught following a 23-page instruction book (see Supplementary Methods 1) and using the user interface in *introduction* mode without video player or trial navigator. The robot remained out of sight to avoid distractions until it was revealed by the instructor when necessary. In the second part, the user interface was used in *training* mode, and two example trials were presented. The adult instructor paused the sample videos on each of the events and built the block sequence that reproduced them. At the end of each trial, the correct solution was shown in an examples book (see Supplementary Methods 1) and was checked against the sequence built. Mouse, keyboard, and instruction materials were under sole control of the adult instructor in both parts.

Test stage

Then, on average 4 days later (min = 2, max = 7), children taught by an adult were evaluated individually on 10 trials with the user interface in *test* mode. Following a tight script, participants were told that they needed to use the blocks from the editor's block panel to copy what appeared in the videos. The video player, trial navigator, and self-assessment buttons were explained as well. Then, the researcher gave participants the images and sounds catalog (see Supplementary Methods 1), opened Trial 1, and left the room. The researcher reentered the room when the session was over 20 min later and rapidly reviewed children's responses by simply congratulating them for their work in each trial without giving extra feedback about their performance.

Peer tutoring/self-revision stage

Next, adult-taught children were divided into two groups of equivalent performance: tutor (n = 15) and control (n = 12) groups. A third group included children who had not participated in the preceding *instruction* and *test* stages: peer-taught group (n = 15). Children in the tutor group were paired with same-gender classmates in the peer-taught group and were asked to teach their peers so that they could play as well (peer-tutoring condition; on average 7 days after test stage). The tutor was given the same instruction materials used by the adult instructor in the *instruction* stage, and up to 40 min were provided (see Supplementary Methods 2). On the other hand, children in the control group were asked to revise the instruction materials on their own so that they would play even better next time (self-revision condition; on average 7 days after test stage). The procedure ensured that the self-revision condition was as similar as possible to the peer-tutoring condition, with the only differ-

ences being (a) social (individual vs. paired) and (b) motivational (revising to improve performance vs. teaching to let a peer play as well). It is worth noting that to make sure that being taught by a peer was not the same as not being taught at all, a complementary study was conducted where the peer-taught group was replaced with a non-taught group (see Supplementary Discussion 1).

Final stage

In this stage, all children were evaluated. In the peer-taught children's test, peer-taught children were evaluated for the first time with the same test that their peers were given in the *test* stage. This test was compared with the adult-taught children's test above to assess the effect of learning conditions (see Results). This took place on average 5 days after being taught by a peer (min = 2, max = 7). In the adult-taught children's retest, on the other hand, controls and tutors were evaluated for the second time with the user interface in *retest* mode. Trial videos in this mode were different but isomorphic to videos in the *test* mode (see "Trial videos" section). This retest was used to assess the teaching effect (see Results). This took place on average 9 days after peer tutoring for tutors and 8 days after self-revision for controls.

Follow-up stage

To test whether teaching a peer had an effect on long-term retention, we evaluated all groups again 2 years later. Only the interface was explained to them again, but no further instructions about the programming language were provided (see Supplementary Discussion 2).

Accuracy

Accuracy was calculated for each trial by quantifying the difference between (a) the block sequence built by participants and (b) the correct sequence for that trial, that is, the shortest block sequence that recreated the series of events portrayed in the trial video (see Supplementary Methods 3). To do so, the Levenshtein distance was used, which counts the number of steps to go from one sequence to another, that is, the number of blocks that need to be removed or inserted (Levenshtein, 1966; Soukoreff & MacKenzie, 2001). In general, the sequence at the end of a trial's last attempt was taken as participants' response for that trial (see Supplementary Methods 2). If more than one sequence was available on the editor's canvas, the best matching sequence was used.

To easily measure the Levenshtein distance between sequences, command and argument blocks were translated into ASCII characters according to a translation table (see Supplementary Methods 5) such that one block = one character. Levenshtein distances can take values from 0 to the sum of the lengths of the sequences being compared. To work with values that are independent from sequence lengths, the following ratio r was used:

$$r = \frac{1 - d}{len(seq1) + len(seq2)}, \tag{1}$$

where d is the Levenshtein distance between sequences seq1 and seq2 and len(seq) is the length of sequence seq. Ratio r ranges from 0 if seq1 and seq2 share nothing in common to 1 if they match perfectly. See Supplementary Methods 2 for full details of the Levenshtein distance implementation used.

Trial time

Each time a trial was accessed through the trial navigator was counted as one attempt for that trial (see "User interface" section). Trial time is the sum of the time (in seconds) spent in all attempts of a trial. As it is usually the case with response times, trial time distribution was positively skewed (Ratcliff & McKoon, 2008), and therefore its logarithm was used instead.

Performance

A trial was marked as "finished" if the following two conditions were met: (a) at least one block was dragged onto the editor's canvas in at least one of the trial's attempts and (b) the next trial button was clicked at least once in that trial. Three measures of performance were defined: aggregate score, mean trial accuracy, and mean trial time. The aggregate score is the sum of all trial accuracies regardless of whether they were finished or not in the 20 min available per session; hence, it is a combination of trial accuracy and time.

On the other hand, the mean trial accuracy and time are the average accuracy and time across trials, respectively. To make sure that these means unbiasedly represented children's intended responses, unfinished trials were excluded because time could have run out while children were still building their sequences or even before attempting the trial at all. However, because trials differ in complexity, it was desired to calculate them using a set of trials finished by most participants. Therefore, only trials in the 1–3 trial block were used because the mean proportion of finished trials in this block was above 90% for all groups. In addition, sessions with 1 or more unfinished trials in the 1–3 trial block were excluded from mean trial time calculation (sessions from 2 tutors and from 2 peer-taught children).

Data analysis

Data and statistical analysis and visualizations were run with Pandas (https://pandas.pydata.org), Seaborn (https://seaborn.pydata.org), StatsModels (https://www.statsmodels.org), SciPy (https://www.scipy.org), afex (https://afex.singmann.science), and other Python and R libraries using Jupyter Notebook (https://jupyter.org).

Assumptions were checked for all parametric tests, and nonparametric alternatives were used where these assumptions could not be verified. In some cases, nonsignificant effects of factors included in the models were not reported to improve readability.

Data availability

The data sets generated are available from the corresponding author upon request.

Ethical compliance

All children's parents or legal guardians gave signed voluntary consent previously authorized by an ethical committee, Comité de Ética de la Dirección de Investigación del Centro de Educación Médica e Investigación Clínica "Norberto Quirno" (CEMIC), Unidad Asociada del CONICET, Protocol No. 683.

Results

Effect of learning conditions

In this study, some children were taught by an adult instructor in a group lesson (adult-taught children), whereas others were taught by a peer in a one-to-one lesson (peer-taught children). First, the effects of these two learning conditions on learning were evaluated by comparing the test stages of peer-taught children versus adult-taught children. To do so, aggregate score in the test stage was compared between groups, that is, the sum of accuracies from all trials, regardless of whether they were attempted and finished or not, in the 20 min available. A two-way analysis of covariance (ANCOVA) was run, with group (i.e., adult-taught or peer-taught groups) and gender as factors and age as a covariate. Children who were taught by a peer in the peer-tutoring stage (peer-taught group; 68% confidence interval (CI) [3.80, 4.61]) performed as well as children taught by an adult instructor in the instruction stage (adult-taught group; 68% CI [3.62, 4.09]), $F(1, 37) = 0.510$, $p = .480$ (Fig. 2). In addition, a main effect of gender was found, with female participants

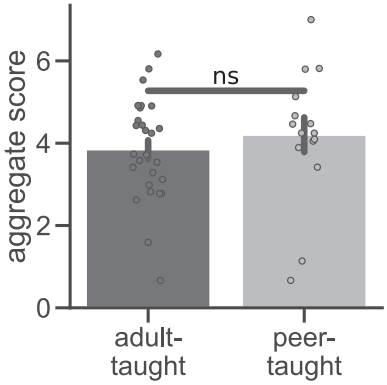

Fig. 2. Aggregate score. Aggregate score is the sum of accuracies from all trials, including unattempted (with accuracy = 0). Children who learned from a peer (peer-taught group) performed as well as children who learned from an adult (adult-taught group). ns, nonsignificant. Error bars represent 68% bootstrap confidence intervals.

showing a lower aggregate score (68% CI [3.39, 3.90]) than male participants (68% CI [4.20, 4.85]), F (1, 37) = 5.056, p = .031. No significant group by gender interaction effect was found, F(1, 37) = 0.256, p = .616.

Then, performance was analyzed trial by trial. In the 20 min available in the test stage, children could spend as long as they desired on any 1 of the 10 trials, thereby affecting the proportion of trials finished. In addition, the trials differed in complexity, with later trials being more complex, and trial complexity presumably would influence trial accuracy. Therefore, to compare the average trial accuracy between participants without bias, it was important to use only trials that were finished by most participants in all groups. Hence, trials were divided into three even trial blocks: 1–3, 4–7, and 8–10. Participants finished nearly all trials in the 1–3 trial block (on average 94% for adult-taught children and 98% for peer-taught children); thus, this block was selected to study performance on a per trial basis. In the 4–7 trial block, in contrast, not only was the average proportion of finished trials lower, but it was also significantly lower for adult-taught children (68% CI [0.33, 0.45]) than for peer-taught children (68% CI [0.57, 0.73]), as shown by a Kruskal–Wallis test, H(1) = 5.510, p = .019 (Fig. 3A, left). In fact, peer-taught children spent significantly less time per trial (in seconds) in the 1–3 trial block (n = 13; 68% CI [113.94, 138.24]) than adult-taught children (n = 25; 68% CI [172.43, 201.24]), as shown by a three-way ANCOVA, with group and gender as between-participant factors, trial as a within-participant factor, and age as a covariate, F(1, 33) = 7.601, p = .009 (Fig. 3A, right). The fact that peer-taught children spent less time per trial (∼60 s less) in the 1–3 trial block may explain why they finished more trials in the 4–7 trial block. In addition, female participants spent more time per trial (in seconds) (n = 23; 68% CI [166.05, 193.09]) than male participants (n = 15; 68% CI [124.99, 157.99]), F(1, 33) = 3.143, p = .085, although no significant gender effect on trial time was found. Finally, the average proportion of finished trials in the 4–7 trial block was significantly lower for female participants (68% CI [0.33, 0.44]) than for male participants (68% CI [0.58, 0.73]), as shown by a Kruskal–Wallis test, H (1) = 6.324, p = .012.

In contrast, peer-taught children were significantly less accurate per trial (trial accuracy; 68% CI [0.62, 0.69]) than adult-taught children (68% CI [0.75, 0.79]) in the 1–3 trial block, as shown by a two-way ANCOVA, with group and gender as factors and age as a covariate, F(1, 36) = 9.065, p = .005. It is worth noting that the peer-taught condition was not merely a non-taught condition, as shown in a complementary study (see Supplementary Discussion 1). This speed–accuracy trade-off, with lower trial times (which result in more finished trials [Fig. 3A]) countered by lower trial accuracy (Fig. 3B), may explain why peer-taught children's aggregate score was indistinguishable from that of adult-taught children (Fig. 2). On the other hand, no main effect of gender on mean trial accuracy was found, which was not significantly different between female participants (68% CI [0.69, 0.75]) and male participants (68% CI [0.72, 0.77]), F(1, 36) = 0.510, p = .480.

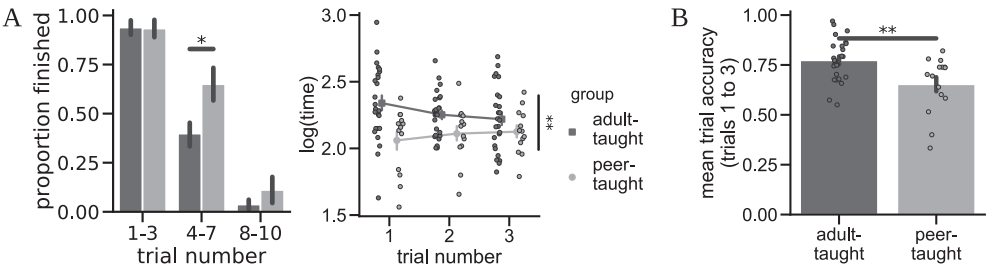

Fig. 3. Performance on a per trial basis. (A) Left: Proportion of finished trials. Children who learned from a peer finished more trials in the test stage than children who learned from an adult. Right: Time spent per trial. Children who learned from a peer spent less time per trial than children who learned from an adult. Trial times (in seconds) follow a log-normal distribution, and thus a logarithmic transformation was applied to them. Participants who did not finish all 1–3 trials were excluded. (B) Trial accuracy. Children who learned from a peer had lower accuracy per trial than children who learned from an adult. One participant who did not finish any 1–3 trials was excluded. $^{*}p < .05$; $^{**}p < .01$. Error bars represent 68% bootstrap confidence intervals.

In summary, from the perspective of aggregate score (i.e., the sum of the accuracy from all trials), children who were taught by a peer were indistinguishable from children who were taught by an adult in a group lesson. However, a closer look reveals that in fact peer-taught children were faster, spending less time per trial and thus finishing more trials per session, although less accurate, with lower trial accuracy. On the other hand, although female participants had lower aggregate scores than male participants, this difference was not because of lower trial accuracy but rather because they were slower and finished fewer trials.

Teaching effect

Our second hypothesis argued that children would benefit from their role as tutors. To examine this, after evaluating them for the first time in the test stage, some of the children (tutors) who had learned from an adult were asked to teach a naive classmate, whereas others were asked to revise the instruction materials alone (controls). Finally, both groups were evaluated again in the retest stage (see Method). To compare aggregate scores between stages (test and retest) and groups (controls and tutors), a repeated-measures three-way ANCOVA was run, with group and gender as between-participant factors, stage as a within-participant factor, and age as a covariate. A significant stage effect was found, with aggregate score increasing from test to retest for both controls (test: 68% CI [3.83, 4.25]; retest: 68% CI [5.44, 6.04]) and tutors (test: 68% CI [3.33, 4.09]; retest: 68% CI [4.84, 5.68]), $F(1, 22) = 54.978$, $p = 2.04 \times 10^{-7}$). However, we found no significant interaction between stage and group, $F(1, 22) = 0.207$, $p = .654$ (Fig. 4). Hence, the increase in aggregate score does not seem to have emerged as a result of teaching a peer but rather seems to be due to a retest effect and/or an effect of revising the instruction materials (either in pairs or alone). In addition, no group effect was found in the follow-up stage 2 years later, disputing the long-term tutor effect on performance hypothesis as well (see Supplementary Discussion 2).

A detailed analysis revealed that this increase in aggregate score in both groups not only was due to a small increase in mean trial accuracy but also was mainly due to a decrease in total trial times, which led to more finished trials (see Supplementary Results 1).

In short, although children improved from test to retest, this improvement was not tied to teaching a peer and thus was the same for controls and tutors.

Discussion

In this study, we approached the problem of teaching programming languages to children by considering the potential role of peer tutoring. We studied both whether children could learn a program-

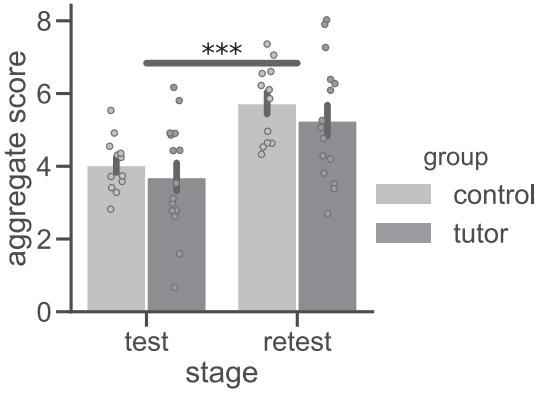

Fig. 4. Teaching effect. Both controls and tutors improved in retest, but no evidence was found that teaching provided an extra benefit (or detriment) to tutors on performance. ***$p < .001$. Error bars represent 68% bootstrap confidence intervals.

ming language from a peer and whether children would benefit from teaching others. To do so, we examined their performance in a series of programming tasks.

Learning from a peer

To test whether children could learn from a peer, we first compared adult-taught and peer-taught children on their aggregate scores. Aggregate scores are regularly used in school tests, where children are given a certain number of questions and have a fixed amount of time to answer all of them. At the end, the question scores are summed up so that it not only matters how well children responded to each question but also matters how many questions they managed to respond to. In this scenario, a question left blank due to lack of time gets 0 points, just like a question left unanswered due to lack of the necessary knowledge. Analogously, our first approach to measure performance involved aggregating accuracy scores along 10 trials of a programming task irrespective of whether children managed to try answering them or not in the 20 min they had available. Remarkably, according to this first coarse measure of performance, children who were taught by a peer performed as well as children who were taught by an adult.

But this result does not necessarily mean that adult-taught and peer-taught children are indistinguishable. Therefore, a finer-grained approach to performance was taken, examining trial accuracy and trial time separately. With this approach, children were found to differ in their speed–accuracy trade-off; peer-taught children seem to have adopted a less conservative strategy, with faster response times than their adult-taught classmates but at the expense of lower trial accuracy. This finding is surprisingly similar to a previous study showing that, when facing a stronger opponent, chess players adopt a more conservative strategy (Slezak & Sigman, 2012). According to the regulatory focus theory (Higgins, 1997), people can be set in a *promotion mode,* where the focus is on seeking to approach a desired state (e.g., winning a chess game or, in our case, successfully solving a programming task), or in a *prevention mode,* where the focus is on avoiding undesired states instead (e.g., making mistakes). In the promotion mode, people would be eager to achieve the desired state, increasing speed and decreasing accuracy (Slezak & Sigman, 2012). By analogy with how instruction can change exploratory policies in children, limiting spontaneous exploration and discovery (Bonawitz et al., 2011), our results could be interpreted using the regulatory focus theory. Under this framework, adult-taught children, who learned from a much more knowledgeable and authoritative figure, could have adopted the prevention mode, reflecting a cautious way of playing and avoiding errors, whereas peer-taught children could have been set in the promotion mode instead, taking more risks through trial and error. We speculate, then, that these results may suggest that children taught by a peer may develop a different understanding of how to use new knowledge, which could be some kind of interaction between peer tutoring and children's epistemic beliefs. This may have resulted from dif-

ferent expectations across conditions as to why children were learning first and why they were playing later. For example, children taught by an adult may have expected to be evaluated, whereas children taught by a peer may have expected that they would just play. Future studies manipulating children's expectations may help to further understand these results.

Although hasty decisions often lead to poor choices, accurate decisions are ineffective if they take too long (Standage, Wang, Heitz, & Simen, 2015). For example, adopting a conservative strategy against a stronger opponent actually reduces the chances to win in chess (Slezak & Sigman, 2012). So, what is the best speed–accuracy trade-off? It would probably depend on the situation at hand whether to prefer teaching methods that promote the more conservative prevention mode, with a focus on avoiding mistakes, or methods that promote the less conservative promotion mode, focusing on results instead and allowing some trial and error.

In addition, it is worth considering some limitations of our experimental approach. First, because children could have learned the basics of the language while they were taking the test, it is possible that the peer-taught condition was merely a non-adult-taught condition indistinguishable from children not taught at all. To evaluate this possibility, we conducted a complementary study where we compared two conditions, peer-taught children versus non-taught children, and successfully found that children taught by a peer were significantly more accurate than children not taught at all (see Supplementary Discussion 1). On the other hand, our experimental procedure was aimed at making the peer-taught and adult-taught conditions as comparable as possible. For example, children taught their peers using the exact same instruction materials used by the adult instructor. However, it is worth noting that conditions differed not only in whether children were taught by an adult or by a peer but also in whether classes were in groups or one on one, respectively. It is possible that one-on-one instruction would increase trial accuracy in adult-taught children (Bloom, 1984), hence widening the difference with peer-taught children.

Another aspect we wanted to examine was whether children exhibited different behaviors or strategies depending on who had taught them irrespective of their performance. For example, all trial videos required children to clear any image or text in the programmable environment's virtual screen at the end of their programs. To do so, children needed to use the corresponding command blocks with an empty argument (see Method above and Supplementary Methods 3). This was demonstrated by the adult instructor in the adult-taught condition. Interestingly, whereas half the children taught by an adult tried to reset the virtual screen in at least one of the programs they built, no children taught by a peer did this. This is compatible with tutors not simply imitating their adult instructors but rather digesting the information provided to them and then elaborating their own teaching (Bridgers, Jara-Ettinger, & Gweon, 2020). Further analysis of the peer interactions should be conducted to support this hypothesis.

In summary, our study shows that children can learn computer programming from a peer. Peer-taught and adult-taught children were indistinguishable using aggregate scores such as those frequently used in school tests. Nonetheless, children who learned from a peer showed a less conservative speed–accuracy trade-off, with shorter response times, at the expense of lower trial accuracy.

The effect of teaching a peer

Peer tutoring has been found to benefit not only tutees but also tutors (Cohen et al., 1982; Duran, 2017; Goodlad & Hirst, 1989; Roscoe & Chi, 2008). However, there is no clear consensus about the conditions under which this tutor effect is evident, and some studies have failed to find it (de la Hera et al., 2019; Rohrbeck, Ginsburg-Block, Fantuzzo, & Miller, 2003; Roscoe & Chi, 2007). Then, the effect of teaching on performance was examined. Our design was aimed at making the peer-tutoring and self-revision conditions as comparable as possible, with the only two differences being *social* (in one case children were interacting with a peer, whereas in the self-revision condition they were alone) and *motivational* (in one case the goal was to teach a classmate so that the classmate could play too, whereas in the other it was to revise the material to perform better the next time). Our results showed that children improved their aggregate score the second time they were tested. This increase not only was due to a small increase in mean trial accuracy but also was mainly due to shorter response times in retest, which in turn led to more trials finished (Supplementary Results 1). However, this improve-

ment was not significantly greater for tutors than for controls, thereby failing to support the tutor effect hypothesis on learning.

A variety of reasons may underlie our failure to identify a tutor effect on performance. First, tutoring a peer has been hypothesized to help tutors metacognitively reflect on their own expertise and comprehension through explanation and questioning (Roscoe & Chi, 2007). However, although this better understanding of their own knowledge could undermine participants' confidence, it might not immediately translate into learning benefits until participants are given the chance to identify or construct a correct answer. Therefore, tutors in our study may have become aware of gaps in their knowledge while teaching without having a chance to fill them before retest. Further studies including an additional revision instance between peer-tutoring and retest stages could help to test this hypothesis. Second, some authors suggest that cognitive benefits of teaching actually result from expecting to teach, rather than from teaching itself, and the associated greater effort to select the relevant elements and organize them into a meaningful representation (Bargh & Schul, 1980; Benware & Deci, 1984; Duran, 2017). Prompting this expectation to teach before even learning may help to test this hypothesis. Finally, two aspects of our study may have biased tutors to unproductive knowledge telling (Roscoe & Chi, 2007). On the one hand, although there is evidence that tutor training is not necessary (Cohen et al., 1982; Leung, 2015), with some authors even proposing that teaching is a natural cognitive skill (Calero, Goldin, & Sigman, 2018; Strauss, Calero, & Sigman, 2014; Strauss & Ziv, 2012), previous studies have shown that peer tutoring works best if tutor training is provided (Topping, 2017). The fact that our study implied no tutor training on teaching may have led to unproductive knowledge-telling bias (Roscoe & Chi, 2007). On the other hand, even though the availability of instruction materials makes our study a case of structured peer tutoring, which is known to show greater benefits than unstructured tutoring (Topping, 2017), the presence of teaching materials may have biased tutors to knowledge telling as well, preventing them from learning by retrieval (Koh, Lee, & Lim, 2018).

In addition, because teaching may have an effect on long-term retention (Ellis, Semb, & Cole, 1998; Fiorella & Mayer, 2013; Hermida et al., 2021; Roscoe & Chi, 2007; Sabol & Wisher, 2001; Semb, Ellis, & Araujo, 1993), we evaluated all groups again 2 years later. However, although most children reported that they remembered the game, the performance was the same across groups and was indistinguishable from children who were not taught at all (see Supplementary Discussion 2). Future studies in the field with shorter retention times may help to test this hypothesis as well.

In short, although our study failed to find beneficial effects of teaching on performance, no learning impairment in tutors was found either. Future studies may help to identify conditions under which the effect of teaching on performance is evident as well as other aspects to which it may contribute.

Can we talk about gender differences?

Another aspect that emerged from our study was gender differences. Many studies have described differences in performance and interest in technical domains between male- and female-gender participants (Edwards et al., 1997; Hill et al., 2010; Nourbakhsh et al., 2004). Most (if not all) of these differences are the by-product of social convention stereotypes (e.g., see Beilock et al., 2010; Dar-Nimrod & Heine, 2006). In this study, we expected that children may be free from these biases given their young age and what we hoped to be a limited cultural indoctrination regarding gender stereotypes. Nevertheless, we found that girls had lower aggregate scores than boys. Similar results were found by Sullivan and Bers (2013) in a robotics program involving kindergarten children. However, a closer look at our results revealed that girls' lower aggregate score was not because of lower trial accuracy but rather because they were slower and finished fewer trials. This is in line with previous findings that girls are less likely than boys to take risks to achieve a goal when programming with LOGO, exhibiting a more cautious or careful approach to task solution (Yelland, 1993). Gender differences in regulatory control (McClelland, Geldhof, Cameron, & Wanless, 2015) could be related to this. These differences may in part be an effect of gender stereotypes on children's cognition (Halpern, 2012; Rippon, 2019), which some studies report might emerge as early as 6 years (Bian, Leslie, & Cimpian, 2017).

Briefly, although we did find performance differences between boys and girls, these may have been due to different attitudes toward risk and not due to differences in knowledge or skill. Therefore, this finding raises a sign of warning that apparent lower performance may hide equivalent accuracy.

Conclusions

Peer tutoring has been found to be useful in a wide variety of settings. However, results are highly variable, and they depend on a multitude of factors such as the targeted subject matter, the age of the participants, and the availability and frequency of tutor training, among many others (Leung, 2015). Given the growing interest in teaching computer science in schools, it is surprising that peer tutoring of computer programming in children has not been experimentally studied so far. Children are digital natives, and it should not be denied that most of the time they outperform their teachers in this field (Benotti et al., 2014). This makes peer tutoring of programming languages among children even more interesting to complement traditional education in schools.

To the best of our knowledge, our study is the first experimental study to focus on peer tutoring of programming languages in children. It provides empirical results favoring peer tutoring as a complementary means of teaching programming languages, bringing further evidence to how peer tutoring can help promote learning in fields where learning is difficult to achieve (de la Hera et al., 2019). Interestingly, not only does learning to code improve computer programming skills, but there is evidence that it also transfers to other general cognitive skills (Fundación Sadosky, 2016; Resnick et al., 2009; Scherer, 2016; Scherer et al., 2019; Voogt et al., 2015), some of which are collectively referred to as *computational thinking* (Grover & Pea, 2013; Wing, 2017). Therefore, our study opens new possibilities in this research field as well, which is far from settled and is still an ongoing effort (Scherer, 2016). In addition, teaching computer programming through peer tutoring may also result in benefits specific to peer tutoring such as metacognitive benefits (Topping, 2017), advantages in social and communication skills (Xu, Gelfer, Sileo, Filler, & Perkins, 2008), and advantages in affective functioning (Miller, Topping, & Thurston, 2010).

Computers and computer networks lie at the foundations of our worldwide society. All in all, our study contributes to the widespread understanding of how these foundations operate and how to shape them, both of which are essential to the values of democracy and plurality, as a way of freeing future generations from the infertile grounds of technological dependency and leading them toward technological sovereignty instead.

Acknowledgments

This work was supported by Argentina's National Agency for the Promotion of Science and Technology (ANPCyT PICT 2013 1653), Argentina's National Scientific and Technical Research Council (CONICET), and the James S. McDonnell Foundation. LEGO WeDo software and materials were kindly donated by Educación Tecnológica, an official partner of Lego Education. We thank Alejandra Lupiz, Luciana Lucchina, Paloma Rodríguez Ferrante, and Tamara Niella for their assistance with data collection; the children and adults who helped us pilot the study; and Alexandra Elbakyan, without whose brave commitment to open science this work may not have been possible. Special acknowledgments go to the children who participated, their parents, and the school's authorities and teachers for their interest and helpfulness. LEGO and the Lego minifigure (shown in Figure 1) are trademarks and/or copyrights of the LEGO Group. ©2020 The LEGO Group. Used here with special permission.

Author contributions

Diego P. de la Hera designed the study; developed the methodology, including the software; collected, curated, and analyzed the data; discussed the results; and wrote the manuscript. María B. Zanoni discussed the methodology; helped with data collection; and discussed the results. Mariano Sigman acquired the funds; designed the study; discussed the results; and wrote the manuscript. Cecilia I. Calero provided the conceptualization; designed the study; discussed the results; and wrote the manuscript.

Appendix A. Supplementary material

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jecp.2021.105335.

References

Altintas, T., Gunes, A., & Sayan, H. (2016). A peer-assisted learning experience in computer programming language learning and developing computer programming skills. *Innovations in Education and Teaching International, 53*, 329–337.

Balanskat, A., & Engelhardt, K. (2015). *Computing our future: Computer programming and coding—Priorities, school curricula and initiatives across Europe.* Technical report, European Schoolnet. http://www.eun.org/resources/detail?publicationID=661.

Bargh, J. A., & Schul, Y. (1980). On the cognitive benefits of teaching. *Journal of Educational Psychology, 72*, 593–604.

Beilock, S. L., Gunderson, E. A., Ramirez, G., & Levine, S. C. (2010). Female teachers' math anxiety affects girls' math achievement. *Proceedings of the National Academy of Sciences of the United States of America, 107*, 1860–1863.

Benotti, L., Martínez, M. C., & Schapachnik, F. (2014). Engaging high school students using chatbots. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education—ITiCSE '14* (pp. 63–68). doi:10.1145/2591708.2591728.

Benware, C. A., & Deci, E. L. (1984). Quality of learning with an active versus passive motivational set. *American Educational Research Journal, 21*, 755–765.

Bian, L., Leslie, S.-J., & Cimpian, A. (2017). Gender stereotypes about intellectual ability emerge early and influence children's interests. *Science, 355*, 389–391.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher, 13*(6), 4–16.

Bonawitz, E., Shafto, P., Gweon, H., Goodman, N. D., Spelke, E., & Schulz, L. (2011). The double-edged sword of pedagogy: Instruction limits spontaneous exploration and discovery. *Cognition, 120*, 322–330.

Bowman-Perrott, L., Davis, H., Vannest, K., Williams, L., Greenwood, C., & Parker, R. (2013). Academic benefits of peer tutoring: A meta-analytic review of single-case research. *School Psychology Review, 42*, 39–55.

Braught, G., Wahls, T., & Eby, L. M. (2011). The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education, 11*, 1–21.

Bridgers, S., Jara-Ettinger, J., & Gweon, H. (2020). Young children consider the expected utility of others' learning to decide what to teach. *Nature Human Behaviour, 4*, 144–152.

Brinda, T., Puhlmann, H., & Schulte, C. (2009). Bridging ICT and CS: Educational standards for computer science in lower secondary education. *ACM SIGCSE Bulletin, 41*, 288–292. https://doi.org/10.1145/1562877.1562965.

Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). Bringing computer science back into schools: Lessons from the UK. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (pp. 269–274). https://doi.org/10.1145/2445196.2445277.

Calero, C. I., Goldin, A. P., & Sigman, M. (2018). The teaching instinct. *Review of Philosophy and Psychology, 9*, 819–830.

Code.org, Computer Science Teachers Association, & Expanding Computing Education Pathways Alliance. (2020). *2020 state of computer science education: Illuminating disparities.* https://advocacy.code.org/stateofcs.

Cohen, P. A., Kulik, J. A., & Kulik, C.-L.-C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal, 19*, 237–248.

Consejo Federal de Educación (2018). *Resolución CFE No 343/18: Aprobación e implementación de "Núcleos de Aprendizaje Prioritarios para Educación Digital, Programación y Robótica" (NAP).* Buenos Aires, Argentina: Ministerio de Educación https://www.argentina.gob.ar/sites/default/files/res_cfe_343_18_0.pdf.

Dalbey, J., & Linn, M. C. (1985). The demands and requirements of computer programming: A literature review. *Journal of Educational Computing Research, 1*, 253–274.

Dar-Nimrod, I., & Heine, S. J. (2006). Exposure to scientific theories affects women's math performance. *Science, 314*, 435.

de la Hera, D. P., Sigman, M., & Calero, C. I. (2019). Social interaction and conceptual change pave the way away from children's misconceptions about the Earth. *npj Science of Learning, 4*(1), 12.

Duran, D. (2017). Learning-by-teaching: Evidence and implications as a pedagogical mechanism. *Innovations in Education and Teaching International, 54*, 476–484.

Edwards, L. D., Coddington, A., & Caterina, D. (1997). Girls teach themselves, and boys too: Peer learning in a computer-based design and construction activity. *Computers & Education, 29*, 33–48.

Ellis, J. A., Semb, G. B., & Cole, B. (1998). Very long-term memory for information taught in school. *Contemporary Educational Psychology, 23*, 419–433.

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

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97.

Fiorella, L., & Mayer, R. E. (2013). The relative benefits of learning by teaching and teaching expectancy. *Contemporary Educational Psychology, 38*, 281–288.

Flannery, L. P., Kazakoff, E. R., Bontá, P., Silverman, B., Bers, M. U., & Resnick, M. (2013). Designing Scratch Jr: Support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)* (pp. 271–274). doi:10.1145/2485760.2485785.

Fundación Sadosky (2016). *Una propuesta para refundar la enseñanza de la computación en las escuelas Argentinas (CC-2016).* Buenos Aires, Argentina: Author.

Golding, P., Facey-Shaw, L., & Tennant, V. (2006). Effects of peer tutoring, attitude and personality on academic performance of first year introductory programming students. In *Proceedings of Frontiers in Education 36th Annual Conference* (pp. 7–12). https://doi.org/10.1109/FIE.2006.322662.

Goodlad, S., & Hirst, B. (1989). *Peer tutoring. A guide to learning by teaching*. Asbury, IA: Nichols.
Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.
Gülbahar, Y., & Kalelioğlu, F. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education, 13*, 33–50.
Halpern, D. F. (2012). *Sex differences in cognitive abilities* (4th ed.). New York: Psychology Press.
Hermida, M. J., Santangelo, A. P., Calero, C. I., Goizueta, C., Espinosa, M., & Sigman, M. (2021). Learning-by-teaching approach improves dengue knowledge in children and parents. *American Journal of Tropical Medicine and Hygiene*. https://doi.org/10.4269/ajtmh.21-0253.
Higgins, E. T. (1997). Beyond pleasure and pain. *American Psychologist, 52*, 1280–1300.
Hill, C., Corbett, C., & St. Rose, A. (2010). *Why so few? Women in science, technology, engineering, and mathematics*. Washington, DC: American Association of University Women.
Hunt, A., & Thomas, D. (1999). *The pragmatic programmer: From journeyman to master*. Boston: Addison Wesley.
International Labour Organization (2020). *Skills shortages and labour migration in the field of information and communication technology in Canada, China, Germany and Singapore*. Geneva, Switzerland: Author. https://www.ilo.org/sector/Resources/publications/WCMS_755663/lang–en/index.htm.
International Telecommunications Union (2021). *ITU-D ICT statistics*. Geneva, Switzerland: Author. https://www.itu.int/itu-d/sites/statistics.
Kalelioğlu, F. (2015). A new way of teaching programming skills to K–12 students: Code.org. *Computers in Human Behavior, 52*, 200–210.
Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*, 245–255.
Koh, A. W. L., Lee, S. C., & Lim, S. W. H. (2018). The learning benefits of teaching: A retrieval practice hypothesis. *Applied Cognitive Psychology, 32*, 401–410.
Legare, C. H., & Lombrozo, T. (2014). Selective effects of explanation on learning during early childhood. *Journal of Experimental Child Psychology, 126*, 198–212.
Leung, K. C. (2015). Preliminary empirical model of crucial determinants of best practice for peer tutoring on academic achievement. *Journal of Educational Psychology, 107*, 558–579.
Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady, 10*, 707–710.
Levin, H. M., Glass, G. V., & Meister, G. R. (1987). Cost-effectiveness of computer-assisted instruction. *Evaluation Review, 11*(1), 50–72.
Lui, K. M., & Chan, K. C. C. (2006). Pair programming productivity: Novice–novice vs. Expert–expert. *International Journal of Human Computer Studies, 64*, 915–925.
Maloney, J., Resnick, M., & Rusk, N. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education, 10*, 1–15.
Martinez, C., Gomez, M. J., & Benotti, L. (2015). A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 159–164). https://doi.org/10.1145/2729094.2742599.
Mayerová, K., & Veselovská, M. (2014). The programming environment for the LEGO WeDo Robotic Construction Set. In J. Kapounová & K. Kostolányová (Eds.), *Information and communication technology in education* (pp. 149–157). Ostrava, Czech Republic: University of Ostrava.
Mayerové, K., & Veselovská, M. (2017). How to teach with LEGO WeDo at primary school. In M. Merdan, W. Lepuschitz, G. Koppensteiner, & R. Balogh (Eds.). *Robotics in education* (Vol. 457, pp. 55–62). Cham, Switzerland: Springer International.
McClelland, M. M., Geldhof, G. J., Cameron, C. E., & Wanless, S. B. (2015). Development and self-regulation. In R. M. Lerner (Ed.), *Handbook of child psychology and developmental science* (pp. 1–43). Kennesaw, GA: American Cancer Society.
Miller, D., Topping, K., & Thurston, A. (2010). Peer tutoring in reading: The effects of role and organization on two dimensions of self-esteem. *British Journal of Educational Psychology, 80*(Pt 3), 417–433.
Mitra, S., Dangwal, R., Chatterjee, S., Jha, S., Bisht, R. S., & Kapur, P. (2005). Acquisition of computing literacy on shared public computers: Children and the "hole in the wall". *Australasian Journal of Educational Technology, 21*(3). https://doi.org/10.14742/ajet.1328.
Nosek, J. T. (1998). The case for collaborative programming. *Communications of the ACM, 41*(3), 105–108.
Nourbakhsh, I. R., Hamner, E., Crowley, K., & Wilkinson, K. (2004). Formal measures of learning in a secondary school mobile robotics course. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings—ICRA '04* (Vol. 2, pp. 1831–1836). doi:10.1109/ROBOT.2004.1308090.
Observatorio Permanente de la Industria del Software y Servicios Informáticos. (2021). *Coyuntura 2020 y expectativas 2021*. Cámara de la Industria Argentina del Software. https://www.cessi.org.ar/opssi-reportes-949/index.html.
O'Reilly, D. (1998). School programming as literacy: The case for BOXER. *Journal of Computer Assisted Learning, 14*(1), 51–58.
Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
Phillips, R., Lockton, D., Baurley, S., & Silve, S. (2013). Making instructions for others: Exploring mental models through a simple exercise. *Interactions, 20*(5), 74–79.
Pinto-Llorente, A. M., Casillas-Martín, S., Cabezas-González, M., & García-Peñalvo, F. J. (2018). Building, coding and programming 3D models via a visual programming environment. *Quality & Quantity, 52*, 2455–2468.
President's Council of Advisors on Science and Technology (2010). *Designing a digital future: Federally funded research and development in networking and information technology—Report to the President and Congress*. Washington, DC: Executive Office of the President. https://eric.ed.gov/?id=ED527261.
Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation, 20*, 873–922.
Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., ... Silver, J. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.

Rideout, V., & Robb, M. B. (2020). *The Common Sense census: Media use by kids age zero to eight*. San Francisco: Common Sense Media.
Rippon, G. (2019). Good girls don't. *Gender and our brains: How new neuroscience explodes the myths of the male and female minds*. New York: Pantheon (Chap. 12).
Rohrbeck, C. A., Ginsburg-Block, M. D., Fantuzzo, J. W., & Miller, T. R. (2003). Peer-assisted learning interventions with elementary school students: A meta-analytic review. *Journal of Educational Psychology, 95*, 240–257.
Roscoe, R. D., & Chi, M. T. H. (2007). Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions. *Review of Educational Research, 77*, 534–574.
Roscoe, R. D., & Chi, M. T. H. (2008). Tutor learning: The role of explaining and responding to questions. *Instructional Science, 36*, 321–350.
Sabol, M. A., & Wisher, R. A. (2001). Retention and reacquisition of military skills. *Military Operations Research, 6*(1), 59–80.
Scherer, R. (2016). Learning from the past—The need for empirical evidence on the transfer effects of computer programming skills. *Frontiers in Psychology, 7*. https://doi.org/10.3389/fpsyg.2016.01390.
Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology, 111*, 764–792.
Semb, G. B., Ellis, J. A., & Araujo, J. (1993). Long-term memory for knowledge learned in school. *Journal of Educational Psychology, 85*, 305–316.
Shamir, Adina, Tzuriel, David, & Rozen, Merav (2006). Peer Mediation: The Effects of Program Intervention, Maths Level, and Verbal Ability on Mediation Style and Improvement in Maths Problem Solving. *School Psychology International, 27*(2), 209–231. https://doi.org/10.1177/0143034306064548.
Silva, L., Mendes, A. J., & Gomes, A. (2020). Computer-supported collaborative learning in programming education: A systematic literature review. In *2020 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1086–1095). https://doi.org/10.1109/EDUCON45650.2020.9125237.
Slezak, D. F., & Sigman, M. (2012). Do not fear your opponent: Suboptimal changes of a prevention strategy when facing stronger opponents. *Journal of Experimental Psychology: General, 141*, 527–538.
Smith, M. K., Wood, W. B., Adams, W. K., Wieman, C., Knight, J. K., Guild, N., & Su, T. T. (2009). Why peer discussion improves student performance on in-class concept questions. *Science, 323*, 122–124.
Söderberg, J. (2012). *Hacking capitalism: The free and open source software movement*. New York: Routledge.
Soukoreff, R. W., & MacKenzie, I. S. (2001). Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. In *CHI EA '01: CHI '01 extended abstracts on human factors in computing systems* (pp. 319–320). doi:10.1145/634067.634256.
Standage, D., Wang, D.-H., Heitz, R. P., & Simen, P. (2015). Toward a unified view of the speed–accuracy trade-off. *Frontiers in Neuroscience, 9*. https://doi.org/10.3389/fnins.2015.00139.
Strauss, S., Calero, C. I., & Sigman, M. (2014). Teaching, naturally. *Trends in Neuroscience and Education, 3*(2), 38–43.
Strauss, S., & Ziv, M. (2012). Teaching is a natural cognitive ability for humans. *Mind, Brain, and Education, 6*, 186–196.
Sullivan, A., & Bers, M. U. (2013). Gender differences in kindergarteners' robotics and programming achievement. *International Journal of Technology and Design Education, 23*, 691–702.
Topping, K. (2015). Peer tutoring: Old method, new developments [Tutoría entre iguales: Método antiguo, nuevos avances]. *Infancia y Aprendizaje*, 1–29. https://doi.org/10.1080/02103702.2014.996407.
Topping, K. (2017). *Effective peer learning: From principles to practical implementation*. New York: Routledge, Taylor & Francis Group.
Tunga, Y., & Tokel, S. T. (2018). The use of pair programming in education: A systematic review. In *Paper presented at EDUCCON 2018 Education Conference, Ankara, Turkey*. https://open.metu.edu.tr/handle/11511/77773.
Umapathy, K., & Ritzhaupt, A. D. (2017). A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. *ACM Transactions on Computing Education, 17*(4), 1–13.
van Dijk, J. A. (2010). *The network society: Social aspects of new media* (2nd ed.). Thousand Oaks, CA: Sage.
Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*, 715–728.
Williams, L. (2001). Integrating pair programming into a software development process. In *Proceedings 14th Conference on Software Engineering Education and Training: "In search of a software engineering profession"* (pp. 27–36). https://doi.org/10.1109/CSEE.2001.913816.
Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software, 17*(4), 19–25.
Williams, L., & Upchurch, R. L. (2001). In support of student pair-programming. *SIGCSE Bulletin, 33*, 327–331. https://doi.org/10.1145/366413.364614.
Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
Wing, J. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology, 25*(2), 7–14. https://doi.org/10.17471/2499-4324/922.
Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *Journal of the Learning Sciences, 17*, 517–550.
Xu, Y., Gelfer, J. I., Sileo, N., Filler, J., & Perkins, P. G. (2008). Effects of peer tutoring on young children's social interactions. *Early Child Development and Care, 178*, 617–635.
Yelland, N. (1993). Young children learning with LOGO: An analysis of strategies and interactions. *Journal of Educational Computing Research, 9*, 465–486.