AI Enhanced Intelligent Texts and Learning Gains

Scott Crossley[1,*], Joon Suh Choi[1], Wesley Morris[1], Langdon Holmes[1], and David Joyner[2]

[1] *Vanderbilt University, 2201 West End Ave, Nashville, TN 37235, USA*
[2] *Georgia Institute of Technology, 225 North Avenue NW, Atlanta, GA 30313, USA*

Abstract
This study provides evidence that students in an introductory computer science class benefitted from the use of an intelligent text that enhanced the reading experience by making it more interactive through the use of large language models (LLMs) as compared to the use of static, digital textbook. Results indicate that higher performing students score higher in a post-test when reading an intelligent text. Secondary results show that learning gains were greater in the pre-test than the post-test for students using the intelligent text. Overall, the study indicates the AI enhanced texts can lead to learning gains in computer science classrooms.

Keywords
Intelligent texts, Natural Language Processing, Reading, Computer Science

1. Introduction

As the prevalence of computers continues to play an important role in business, education, and the arts, computational thinking remains a critical skill for solving important, real-world problems using complex solutions [1]. However, acquiring computational thinking is difficult and requires sustained efforts, diverse abilities, and specialized teaching environments. Unfortunately, there is a consensus that traditional textbooks are not as effective for computational thinking because it requires procedural knowledge that is difficult to demonstrate in static texts [2].

To assess whether students learned better from a traditional or intelligent text, Crossley et al. [3] examined the efficacy of an interactive intelligent text in a computer science class. The intelligent text was developed using Intelligent Texts for Enhanced Lifelong Learning (iTELL), which is a framework that simplifies the creation and deployment of intelligent texts with integrated interactive features. iTELL leverages Large Language Models (LLMs) to develop and deploy interactive content including constructed response items and summaries that are automatically scored. The LLMs simultaneously provide feedback to users to help with revisions.

ITELL was designed to provide students with interactive read-to-write tasks known to lead to increased learning gains [4-6] based on generation effect theories [7]. Read-to-write tasks require readers to extract and integrate text information into their writing allowing which helps them construct knowledge during the reading process [8-11]. Read-to-write tasks like constructed responses [6] and summaries [4-5] are effective learning strategies that can increase learning gains.

The iTELL text was deployed within a Python-based computer science course designed to help students understand and process information about computational thinking and programming. Outtake survey results indicated that students were satisfied with the intelligent text and that students felt the interactive tasks were easy to work with, were accurate, and improved learning. Comparisons between learning gains for the students that used the intelligent text as compared to the traditional digital text reported a small relationship. Specifically, students that used the intelligent

[1] *iTextbooks'25: Sixth Workshop on Intelligent Textbooks, July 26, 2025, Palermo, Italy*
[*] Corresponding author.

scott.crossley@vanderbilt.edu (S. Crossley); joon.suh.choi@vanderbilt.edu (J. Choi); wesley.g.morris @vanderbilt.edu, (W. Morris); langdon.holmes@vanderbilt.edu (L. Holmes); djoyner3@gatech.edu (D. Joyner)
0000-0002-5148-0273 (S. Crossley); 0000-0002-7732-0366 (J. Choi) ; 0000-0001-6316-6479 (W. Morris); 0000-0003-4338-4609 (L. Holmes); 0000-0003-0537-6229 (D. Joyner)

texts reported increased gains of ~5% between a pre-test and a post-test. Overall, the study found that intelligent texts seem to improve student learning through greater interactivity, student engagement, and dynamic assessments.

However, there were limitations to the original analysis conducted by Crossley et al. The study used a quasi-experimental design where students had the option to use a digital or an intelligent text. Students were also with provided extra credit to use the intelligent text, which may have introduced a self-selection bias [12]. Self-selection also led to fewer students in the intelligent text (n = 79) than in the digital text condition (n = 277). Lastly, the analysis included no co-variates of overall student performance in the class to account for student differences.

The goal of the current study is to provide a re-analysis of the data reported in Crossley et al. [3] by selecting participants for the intelligent text and digital text conditions using propensity score matching. Propensity score matching can be used to develop balanced samples of participants between experimental and control groups in observational studies. Specifically, propensity scores are used to match each individual in the experimental group to an individual in the control group based on the probability of participants being in the treatment group as calculated from a range of covariates including demographic and/or individual differences. This matching helps to remove overt bias caused by self-selection and outcomes from matched participants can be used to better estimate the effect of the experimental condition.

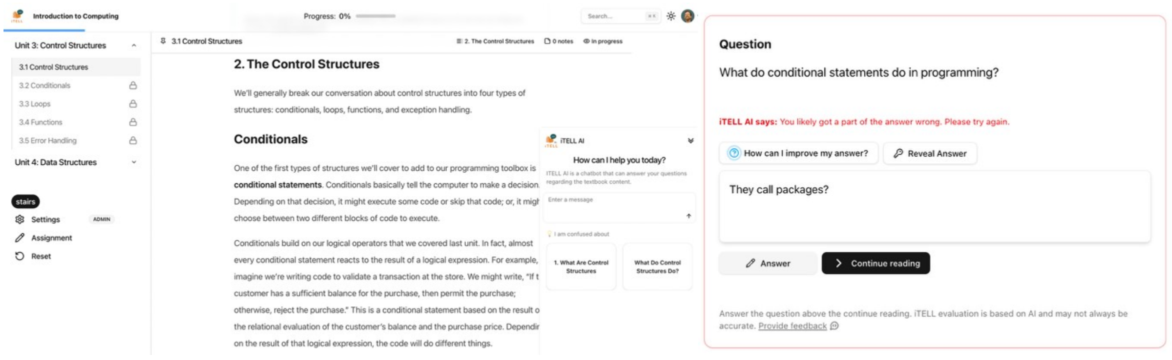
2. Method

2.1. iTELL

iTELL transforms conventional educational resources into interactive texts utilizing an sophisticated content authoring system. Within the content authoring system, instructional materials are segmented into pages and smaller sections known as chunks. A chunk typically lies under a single sub-header and includes 1-3 paragraphs of text or one instructional video. iTELL then creates learning activities for these divisions, leveraging reading comprehension theories to help users develop a deeper understanding of the material through constructed response items (CRIs) and summary writing tasks.

For each chunk, a CRI is crafted using GPT-3.5-turbo with human oversight. In any given iTELL volume, each chunk has a 1/3 probability of generating a constructed response item for the user, ensuring at least one CRI per page. Users must complete at least one CRI before advancing to subsequent chunks. The submitted CRIs are evaluated for accuracy, and feedback is provided by two distinct fine-tuned language models [3].

A summary must be completed at the conclusion of each page, which is then screened automatically for plagiarism, relevance, length requirements, offensive language, and appropriate language use. Summaries that pass these criteria are evaluated by a single language model for content accuracy, assessing whether they capture the essential elements of the original material. Additionally, iTELL incorporates an AI-driven chatbot based on Llama 3, enhanced with retrieval augmented generation (RAG) to ensure responses remain relevant. The chatbot also include safeguards against academic dishonesty and misuse. The chatbot is accessible anytime to assist users with inquiries regarding text-related content or the iTELL platform [3]. See Figure 1 for screenshots of the iTELL interface.



2. The Control Structures

We'll generally break our conversation about control structures into four types of structures: conditionals, loops, functions, and exception handling.

Conditionals

One of the first types of structures we'll cover to add to our programming toolbox is **conditional statements**. Conditionals basically tell the computer to make a decision. Depending on that decision, it might execute some code or skip that code; or, it migh choose between two different blocks of code to execute.

Conditionals build on our logical operators that we covered last unit. In fact, almost every conditional statement reacts to the result of a logical expression. For example, imagine we're writing code to validate a transaction at the store. We might write, "If t customer has a sufficient balance for the purchase, then permit the purchase; otherwise, reject the purchase." This is a conditional statement based on the result o the relational evaluation of the customer's balance and the purchase price. Dependir on the result of that logical expression, the code will do different things.

How can I help you today?

iTELL AI is a chatbot that can answer your questions regarding the textbook content.

Question

What do conditional statements do in programming?

iTELL AI says: You likely got a part of the answer wrong. Please try again.

They call packages?

Answer the question above the continue reading. iTELL evaluation is based on AI and may not always be accurate. Provide feedback

Figure 1. Screen shots for iTELL. The left screenshot is the reading interface with the available iTELL AI chatbot (which can be collapsed). The right screen shot is the CRI interface with an incorrect answer provided.

2.2. Class and textbook

Data was collected from an Introduction to Computing class taught at a large technology university in the southeastern United States. The course covered the basics of computing, presupposing no prior programming ability. The course began with the basics of procedural programming, moved through control structures and data structures, and concluded with object-oriented programming and algorithm development. After each unit, students were required to take a unit test. The tests accounted for 40% of students' grades in the class, 10% for each of four tests.

Demographic and individual difference information was collected through surveys at the beginning of the class. The survey collected demographic data such as age, gender, race or ethnicity, country of origin, and first language background. Students also provided information related to programming experience (from no experience to successful completion of programming classes) and their year of study (freshman, junior, sophomore, or senior).

Data was collected from the third unit of the course textbook [13], which covered Control Structures. This unit was ingested into an iTELL volume that comprised an overview page introducing iTELL and 5 additional pages with each page referencing a chapter from the unit. Each page faithfully represented the original text. However, screenshots of integrated development environments (IDEs) in the textbook used to demonstrate Python code and output were replaced with a Python interactive sandbox. Student test scores from the second unit (Procedural Programming) were used as a pre-test, and student test scores from the Control Structure unit (the third unit) were used as post-test scores.

2.3. Participants

We used the same 476 students enrolled in the class as reported by Crossley et al [3]. Most of the students were freshmen or juniors. Of these, 121 students self-selected to use the iTELL volume and 356 used the traditional digital version of the text (i.e., a pdf version). The students that used iTELL were given 1% extra credit to their overall course grade. Of the 121 students that elected to use iTELL, 79 students completed all requirements for inclusion in the analyses including age and consent requirements, and completion of the pre-test, and the post-test. Of the 79 students in the iTELL condition, 72 had complete demographic and individual difference survey data while 238 of the 356 students in the digital text condition had complete survey data. All these students completed the class and thus received a final class score, which indicated their overall performance in the class.

2.4. Propensity matching

We used the MatchIt package [14] in R for propensity matching for the students that had complete data. The package implements the suggestions of Ho, Imai, King, and Stuart [14] for preprocessing data using semi-parametric and non-parametric matching methods. The purpose of matching data is to reduce bias in observational or quasi-experimental studies by simulating the balance one would expect to find in a randomized design. The end goal of matching is to create a dataset that has covariate balance among participants where the distribution of covariates in groups is similar to that which would be found in a randomized control trial. Covariate balance allows for increased confidence and robustness in subsequent statistical analyses. Matching has three key steps: 1) planning, 2) matching the data, 3) examining the quality of matches [15].

2.4.1. Planning

Planning involves the selection of an outcome variable and selection of covariates that need to be balanced to increase the likelihood of an unbiased estimate of the treatment effect. Here, the outcome variables are performance on the class tests from Unit 2 (pre-test) and Unit 3 (post-test). The treatment effect is whether students read Chapter 3 text in either digital or iTELL format.

Our selection of covariates included both demographic variables and individual differences. Demographic variables were selected to ensure that students' experiences across conditions were strongly matched. The selected demographic variables were gender, age, country of birth, and race/ethnicity. To control for background knowledge, we selected programming experience and years of study as individual differences.

2.4.2. Matching

We selected a distance parameter that used a generalized linear model to calculate propensity scores. We used the "nearest" method parameter to select nearest matches and selected a one-to-one ratio for matching so that the number of matches was the same between the digital and iTELL text conditions. A one-to-one ratio helps to ensure stronger matches, but it does dis-card more potential matches. These matching parameters led to the selection of 72 participants from the digital text condition (i.e., the control condition) that matched the 72 participants from the iTELL condition (i.e., the treatment condition).

2.4.3. Examine quality of matching

The MatchIt package reports the empirical cumulative distribution function (eCDF) mean as a measure of balance between the selected participants in the control and treatment groups. The eCDF mean is the mean difference in the cumulative distribution functions of the propensity scores where a small value indicates a better balance (and an eCDF mean of 0 representing perfect balance). The eCDF mean for the propensity matches between the digital and iTELL conditions reported a M = .023, which indicates strong balance [14]. This distribution is plotted in Figure 2 below.

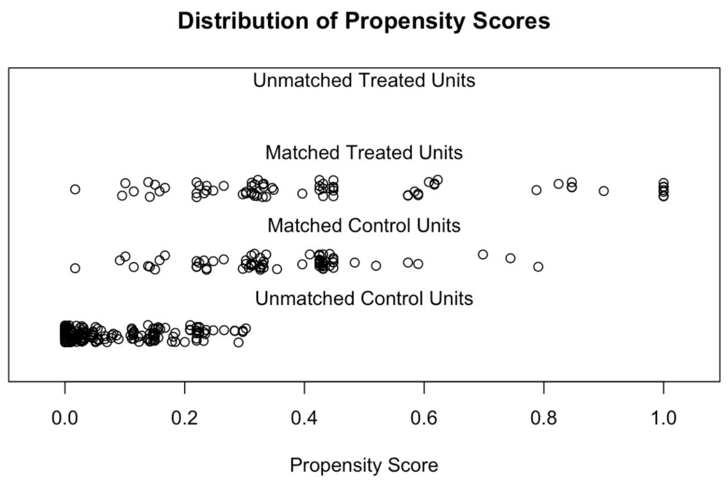


Figure 2. Distribution of propensity scores across matched and unmatched treatment and control conditions.

2.5. Statistical Analyses

Our first analysis reported descriptive statistics for the two conditions across tests scores. The second analysis examined difference between pre-tests and post-tests across conditions using a linear mixed effects (LME) model. For this analysis, we used R [16] and the lme4 package [17]. The outcome variable was all test scores, with the timing of the test administration (pre or post) coded as a fixed effect (Time). Other fixed effects were Condition (Digital or iTELL text) and Final Class Score. We included final class score as a measure of performance to control for potential effects of background knowledge, motivation, and/or ability during data collection. We did not include Class Score as a variable during propensity matching because we wanted to examine potential interaction effects during learning (specifically, was learning differential based on class performance). All numeric variables were scaled. Participants were included as a random effect. We developed an LME using a maximum three-way interaction that included Time, Condition, and Final Class Score.

3. Results

Descriptive statistics for test scores by pre-test and post-test for the matched participants in the digital and iTELL conditions indicated that students in the digital text condition (M = .869, SD = .157) scored higher than students in the iTELL condition in the pre-test (M = .783, SD = .189). This trend held in the post-test, but students in the iTELL condition (M = .834, SD = .263) showed learning gains (Delta = .051) while students in the digital text condition (M = .867, SD = .215) did not (Delta = -.002).

Results from the linear mixed effects models are presented in Table 1. The model indicated a significant three-way interaction between Test, Condition, and Final Class Score (see Figure 1 for interaction plot). The plot shows that students with high Final Class Score showed greater learning gains from reading in the iTELL condition as compared to the Digital condition. The model also yielded a two-way interaction between Time and Final Class Scores. The interaction indicates that students with a higher final class scores showed greater gains in the post-test. The model also showed a significant two-way interaction between Test and Condition. The interaction indicates that students in the iTELL condition showed greater learning gains between the pre- and post-test than students in the Digital condition. Lastly, the results showed a suppression effect for pre- and post-test where the estimate is negative (whereas mean scores show it should be positive). The main effect for Final Class Score approached significance and indicated that students with higher Final Class Scores showed higher test scores in general.

Table 1
Linear Mixed Effect model for test scores between digital and ITELL text conditions

Predictor	Estimate	S.E.	t	p
(Intercept)	0.397	0.209	1.897	0.059
Time (Post-test)	-1.442	0.270	-5.341	< .001
Condition (ITELL)	0.395	0.277	1.426	0.155
Final class score	0.421	0.277	1.854	0.065
Post-test*ITELL	0.882	0.358	2.465	< .050
Post-test*Final Class Score	1.630	0.293	5.559	< .001
ITELL*Final Class Score	-0.336	0.305	-1.101	0.272
Post-test*ITELL*Final Class Score	-1.003	0.393	-2.552	< .050

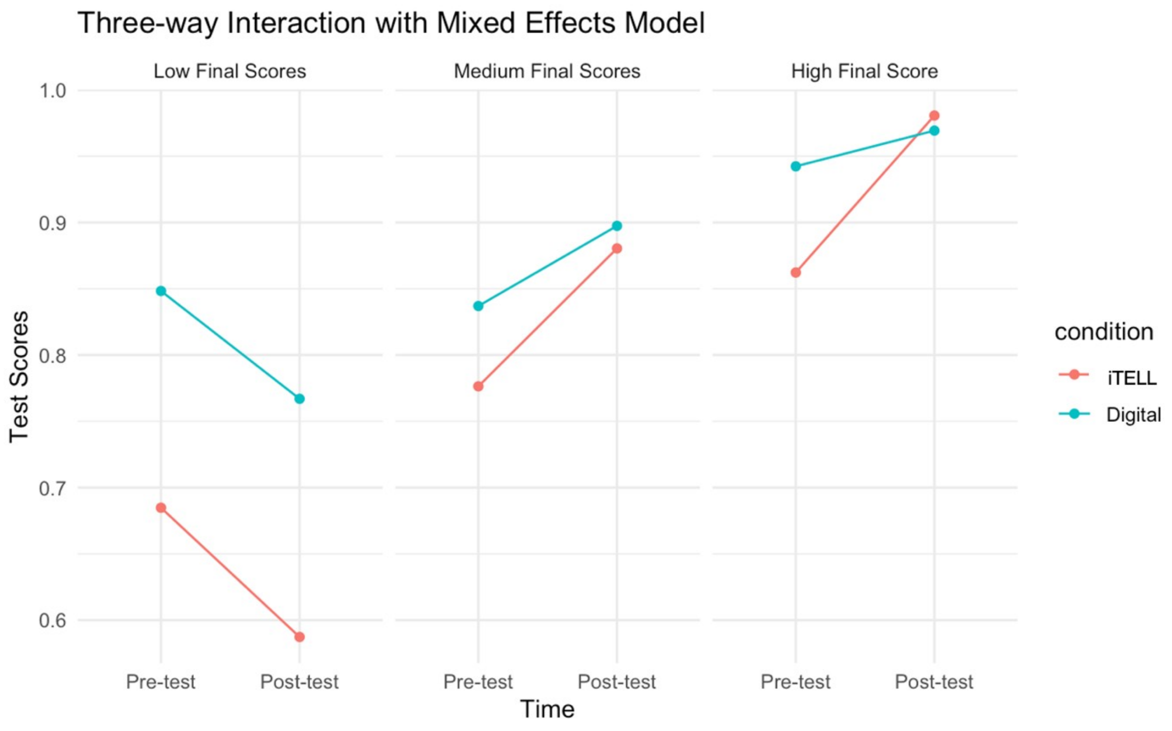


Figure 1: Interaction plot for pre- and post-test scores, condition, and final class score

4. Discussion

This study provides evidence that students in an introductory computer science class benefitted from the use of an intelligent textbook that enhanced the reading experience by making it more interactive. Our main result indicates that higher performing students (as indicated by final class scores) score higher in the post-test in the iTELL condition than in the digital text condition. Secondary results show that test scores were greater from the pre-test to the post-test for the iTELL condition compared to the digital text condition. From a pedagogical perspective, these findings show the intelligent texts may help beginning computer scientists develop computational thinking and computer programming skills. These skills may help students perform better in subsequent course and reduce the high failure and dropout rates [18] found in computer science programs.

Overall, the results indicate that intelligent texts that leverage LLMs to provide interactive reading environments that capitalize on read-to-write tasks [4-6] to increase the likelihood of generation effects in readers [7] lead to learning gains, at least in this setting. It is important to note that the three-way interaction showed that learning gains were greatest for students with higher class scores at the end of the semester, indicating that iTELL may work best for stronger students. There are a few plausible reasons that this may be the case. First, stronger students may be more determined and thus work through the text more thoroughly, including spending more time reading. Second, stronger students may be more diligent and attend to the text generation interventions more completely.

In all cases, we presume that iTELL helps readers construct better mental models of the text leading to greater learning gains, but more evidence is needed to support this assertion. Key to this is addressing many of the limitations found in this study. First, this study only focused on a single course within a single academic domain (computer science). More courses across a wider range of topics are needed. We also offered students an incentive to use iTELL, which may attract certain types of students over others. While propensity matching can control for some differences, it may not capture individual differences that explain selecting the iTELL condition. We do not know if students selecting to use the iTELL system wanted the extra credit because they were performing poorly, were struggling in other ways, or were highly motivated to succeed regardless of performance. Additionally, the pre-test (Procedural Programming) and post-test (Control Structures) assessed non-identical skills, so raw gains may partly reflect differences in quiz difficulty rather than learning. We tried to address this by co-varying class grade, but a better developed pre- and post-test design could provide stronger evidence of learning. Lastly, propensity matching removed around 40% of the original data leaving us with 72 students in each condition. While this sample is large enough for statistical analysis, it is unlikely to generalize past the current population. In general, we suggest that future studies focus on different topics areas and different student populations with larger sample sizes. Additionally, future studies should include multiple experimental conditions to disaggregate simple text production from text production that includes AI feedback and should include multiple measures of reading skill and background knowledge to help control for these differences during learning.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant 2112532. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We are thankful to suggestions made by Ken Koedinger about using propensity matching at Educational Data Mining, 2024.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] J. M. Wing. Computational thinking. *Communications of the ACM 49*, 3 (March 2006), 33–35. (2006)

[2] A. Gomes, A. J. Mendes. Learning to program difficulties and solutions. *In International Conference on Engineering Education–ICEE* (Vol. 7), (2007)

[3] S. A. Crossley, J. S. Choi, W. Morris, L. Holmes, D. Joyner, V. Gupta. Using Intelligent Texts in A Computer Science Classroom: Findings from an iTELL Deployment. Proceedings of 8th Educational Data Mining in Computer Science Education Workshop (CSEDM 2024) at the 17th International Conference on Educational Data Mining (EDM). Atlanta, GA (2024)

[4] A. M. Silva, R. Limongi. Writing to learn increases long-term memory consolidation: A mental-chronometry and computational-modeling study of "Epistemic writing". *Journal of Writing Research, 11*(1), 211-243, (2019)

[5] A. L. Brown, J. C. Campione, J. D. Day. Learning to learn: On training students to learn from texts. *Educational researcher, 10*(2), 14-21, (1981)

[6] M. Bensoussan, I. Kreindler. Improving advanced reading comprehension in a foreign language: Summaries vs. short-answer questions. *Journal of Research in Reading, 13*(1), 55-68, (1990)

[7] S. Bertsch, B. J. Pesta, R. Wiscott, M. A. McDaniel. The generation effect: A meta-analytic review. *Memory & Cognition. 35*(2) 201–210, (2007)

[8] Y. A. Delane. Investigating the reading-to-write construct. *Journal of English for academic purposes, 7*(3), 140-150, (2008).

[9] W. Grabe, F. L. Stoller. *Teaching and researching reading*. Routledge. (2019)

[10] N. Nelson, R. C. Calfee. The Reading-Writing Connection Viewed Historically. Teachers *College Record, 99*(6), 1-52, (1998)

[11] N. Nelson, J. R. King. Discourse synthesis: Textual transformations in writing from sources. *Reading and Writing, 36*(4), 769-808, (2023)

[12] G. Tripepi, K. J. Jager, F. W. Dekker, C. Zoccali. Selection bias and information bias in clinical research. *Nephron Clinical Practice, 115*(2), c94-c99, (2010)

[13] D. Joyner. *Introduction to Computing*. McGraw-Hill Education LLC. (2016)

[14] D. Ho, K. Imai, G. King, E. Stuart. MatchIt: Nonparametric Preprocessing for Parametric Causal Inference. *Journal of Statistical Software, 42*(8), 1–28 (2011)

[15] N. Greifer. MatchIt: Getting Started. https://cran.r-project.org/web/packages/MatchIt/vignettes/MatchIt.html, last accessed 2025/1/28

[16] R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria Available at: https://www.R-project.org/. (2021)

[17] D. Bates, M. Maechler, B. Bolker, S. Walker, R. H. B. Christensen, H. Singmann, B. Dai, F. Scheipl, G. Grothendieck, P. Green, J. Fox, A. Bauer, P. N. Krivitsky, E. Tanaka, M. Jagan. lme4: Linear mixed-effects models using 'Eigen' and S4 (Version 1.1-36) [R package], (2015)

[18] A. Robins, J. Rountree, N. Rountree. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172. (2003)