Visual recipes for slicing and dicing data: teaching data wrangling using subgoal graphics

LOVISA SUNDIN, University of Glasgow, United Kingdom
NOURHAN SAKR, American University in Cairo, Egypt
JUHO LEINONEN, Aalto University, Finland
SHERIF ALY, American University in Cairo, Egypt
QUINTIN CUTTS, University of Glasgow, United Kingdom

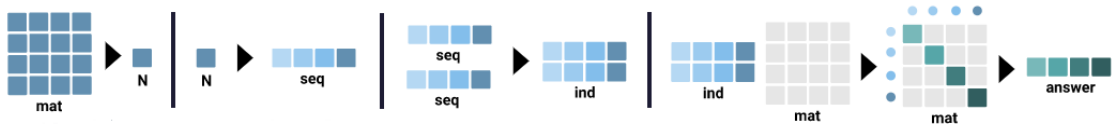


Fig. 1. An example of subgoal graphics visualizing how to access the matrix diagonal, here without subgoal labels.

The rising demand for data wrangling skills in today's global market poses new challenges for the programming education community. Non-majors often need to learn it quickly alongside their other subjects. Previous research suggests that subgoal labels offer a powerful scaffolding strategy to help novices decompose problems. Because data wrangling is inherently easy to represent graphically, we wonder whether such labels could be augmented with subgoal graphics. To test this idea, we developed an online tutorial that features subgoal graphics in both programmatic and non-programmatic data wrangling exercises. Following an RCT paradigm, a control group is only given subgoal labels, without any graphics. The platform collects learner activity in order to evaluate the pedagogical benefits. Participants were recruited from multiple institutions (N=197, 134). Our results did not show a significant difference in various learner performance metrics, however subjective feedback from our participants suggest that learners perceive the graphics to be very helpful. We discuss possible reasons for the apparent disparity between objective and subjective data.

CCS Concepts: • **Social and professional topics** → **Computer science education**; **Information science education**; *Computational thinking*; • **Human-centered computing** → *Visualization design and evaluation methods.*

Additional Key Words and Phrases: Data science; Data wrangling; Programming education; Visualization; Graphics; Subgoals

ACM Reference Format:
Lovisa Sundin, Nourhan Sakr, Juho Leinonen, Sherif Aly, and Quintin Cutts. 2021. Visual recipes for slicing and dicing data: teaching data wrangling using subgoal graphics. In *21st Koli Calling International Conference on Computing Education Research (Koli Calling '21), November 18–21, 2021, Joensuu, Finland.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3488042.3488063

1 INTRODUCTION

Across STEM and beyond, university students are increasingly expected to prepare and analyze their data using programming. This process includes *data wrangling*, i.e. reformatting and manipulating tabular data structures like

Manuscript submitted to ACM

vectors, matrices, and dataframes. Although point-and-click software and spreadsheets still retain their dominance in the social sciences, recent Google Scholar data suggest they are on decline in favor of programming languages like R and Python [36]. In the last few years, a multitude of papers called for introducing programming to non-CS majors [2, 12, 28] while a 2019 survey of psychology courses in Canada found that 19% of intermediate courses, and 40% of graduate courses already used programming [13]. Within such ancillary courses, programming instruction generally competes for time with that of statistics content knowledge, thus creating a steep learning curve in an environment where dedicated teaching hours (and student motivation) may already be scarce [10]. Therefore, we observe a clear need for pedagogical solutions that mitigate this content overload in a time-efficient, effective, enjoyable, and scalable way.

Existing literature on data-related programming for non-majors focuses on the larger-scale motivational aspects, such as media computation [21, 26] and dataset relevance [4]. While context matters to motivation, it is important to also investigate ways of cognitively restructuring data wrangling content to avoid overloading non-majors, for example using visualizations [25, 27]. This paper evaluates *subgoal graphics* as an approach to explaining data wrangling procedures. It extends previous work on *subgoal labels*, where worked examples are annotated with textual labels describing high-level, conceptual steps [7]. To the best of our knowledge, subgoals have not yet been explored in a data wrangling context, nor in combination with graphical aids. Since rectangular data structures are straightforward to visualize, we developed Slice N Dice, a special-built, online data wrangling tutorial which augments every subgoal label with graphical representations. Our app serves as a vehicle for a pre-registered, multi-institutional, randomized controlled trial (RCT) that presents subgoal graphics to a random sample of participants, and tracks user data to explore whether these add any pedagogical value.

2 BACKGROUND

Data can be wrangled using a variety of technologies, from spreadsheets and menu-driven software like SPSS, to automated approaches [16, 20] and block-based programming tools [3, 40, 48] - each with their own pedagogical trade-offs. In this paper, we assume that programmatic data wrangling is the core learning objective, and define our challenge as one of teaching this skill without the overhead of intermediate didactic technologies.

2.1 Data wrangling

Programming can be modeled as a search through the solution space of possible combinations of language primitives. In a traditional introductory programming course, this set of primitives is usually limited to a small set of language constructs, saving time to focus on algorithmic composition. By contrast, data wrangling makes extensive use of high-level library functions, such as functions of NumPy and Pandas in Python [22, 34], or tidyr and dplyr in R [51, 52]. In data wrangling, we may refer to such functional primitives on a language-neutral level as *operations*. An example of an operation is *Pivoting a dataframe from long to wide format*, which is pivot_wider() in dplyr and pivot_table() in Pandas. Some of these operations are relatively trivial (e.g. selecting a column, vectorized arithmetic) while others are more complex (e.g. nesting, merging, pivoting).

The functional primitives in data wrangling are often numerous, mutually redundant, and highly parameterized [37], leading to complex and ever-changing APIs. An interview study with novice data programmers suggests that this complexity is an important source of frustration [54]. Some operations are more common than others, for example dplyr is organized around key functions that roughly correspond to the basic SQL operations [45, 50], but outside of simple split-apply-combine transformations, the set of frequently used convenience functions is much larger. Thankfully, the combination of primitives is usually structurally simpler, with data flowing from function to function in a pipeline-like

fashion. We hypothesize that it is the *identification* of appropriate operations that incurs significant cognitive load on a novice data wrangler's working memory [47] and, therefore, forms a suitable target for guided instruction.

2.2 Subgoal labels

The Computer Science (CS) educational literature offers several scaffolding approaches for facilitating the identification of appropriate operations. One influential approach advocates breaking down worked examples into meaningful *steps* using instructor-provided, natural-language subgoal labels [30]. The underlying theory posits that such labels help constrain search for primitives and facilitates abstraction of high-level mental models likely to reappear in novel problems [30]. Subgoal labels are usually employed within learning materials [14, 31, 32], especially in worked examples [30, 35]. Studies have suggested that subgoal labels can help students get a deeper understanding of introductory programming materials [14, 30–32], including among poorly performing students who are at risk of dropping out from a course [33].

The subgoal-related research in CS education has mainly focused on imperative programming [15] and block-based app development [30]. Data wrangling problems are well-suited for subgoal labeling, due to their pipeline-like nature that readily decomposes into steps. For example, the need for pivoting a time series dataframe into wide format may be hinted at via a subgoal label *Reshape the dataframe so that each timepoint is its own column.* However, the efficacy of subgoal labels likely depends on domain [35], and therefore is not guaranteed.

2.3 Software visualization and subgoal graphics

Another addition to the subgoal label design space - that may or may not be worthwhile - is visualizing each subgoal graphically. Early work in software visualization suggests that graphical representations of programs might not be inherently valuable. For example, Green and Petre in [19] found that visual programs are harder to understand than textual ones, and in [38] that novices have a harder time compared to more experienced people at understanding visual aids. Thus, they suggest that effective use of visualizations might require training. Similarly, Hundhausen et al. [23] found in their meta-study of algorithm visualizations that "passive viewer" software visualizations are ineffective, compared with systems where visualization is a tool for self-guided problem exploration. More recently, Sorva et al. [43] suggested that software visualization systems have a generally positive impact on learning. However, we should exercise due caution before drawing general conclusions regarding the impact of "graphics" in general, as their efficacy likely depends on programming domain and design.

In our context of data wrangling, which primarily concerns tabular transformations, operations are highly visualizable. Iconic representations can graphically demonstrate the pre- and post-operation shape, and utilize visual cues (e.g. color hue and saturation) to convey abstract relations like selection, correspondences, and relative magnitude at a glance. Similar design principles have already been used in RStudio's cheat sheets for various data science APIs [42] and to some extent by creators of SQL query visualization tools [8, 11, 18]. However, little in the way of pedagogical evaluations of such graphics is published. We propose using graphics alongside subgoal labels, which we call *subgoal graphics*. Figure 2 shows an exercise from our current tool, where we leverage subgoal graphics for exercises involving toy datasets. The same method is applicable to snippets of more authentic datasets.

We argue that subgoal graphics may help students by providing hints on the necessary structural changes to an array or dataframe. The added utility of graphics is not obvious, since it could be either redundant or counter-productive, by for example making learners overestimate the extent of their understanding [24]. However, it potentially provides a low-cost method for having novices quickly intuit how to achieve certain data wrangling goals, and therefore merits investigation.

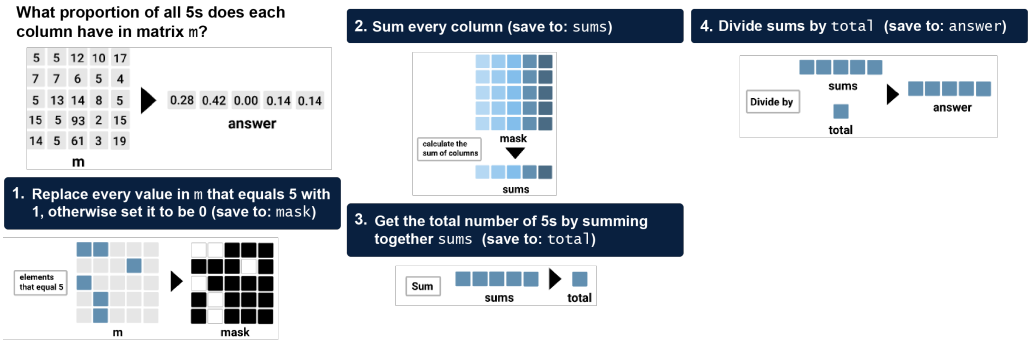


Fig. 2. An example of a data wrangling problem decomposed using subgoal labels and accompanied by subgoal graphics, styled the way it appears in our application. This particular exercise assumes that the participant doesn't already know that Boolean values are equivalent to 1 and 0.

3 INSTRUCTIONAL APPROACH

To test the effects of embedding author-provided subgoals and subgoal graphics, we designed a web app tutorial for teaching novices data wrangling. To understand its structure, it is important to briefly detail the underlying pedagogical model of novice data wrangling that motivates it.

3.1 A model for data wrangling learning

The model sorts requisite skills into two categories: one that concerns skills specific to data wrangling but separable from programming itself, and one that concerns programming but is non-specific to data wrangling. Cutting across these two categories is a division roughly translating to the semantic knowledge of primitives, and the procedural skill of combining the primitives appropriately [49].

3.1.1 Data wrangling. The data wrangling-related skills involve gaining a semantic understanding of tabular data structures, and the operations available for manipulating them. The precise subset of available operations is largely at the discretion of the teacher. Additionally, the novice needs procedural training in breaking down problems into functional steps (*problem decomposition*), and mapping those steps into the data wrangling operations they are familiar with (*plan composition*)[17]. For example, they must understand that the prompt "Out of the students majoring in French, find the tallest" calls for a row filter operation, an aggregation, and an additional filter operation. This is where subgoal labels and graphics are intended to help. Note how this decomposition process may be accomplished non-programmatically, without recourse to textual syntax.

3.1.2 Programming. Assuming the learner is not already a programmer, they will require training in foundational programming concepts like variables, data types, functions, and logical conditions. Not all typical programming fundamentals are required; loops and conditionals are replaced by vectorized functions. Associated with these primitives are the procedural skills of using them, of debugging typos, reading error messages, effectively using the read-eval-print loop (REPL), but also to search through documentation and adapt code examples by making appropriate variable substitutions and parameter selections [5, 6].

3.1.3 Programmatic data wrangling. Once a learner is familiar with both data wrangling operations and general programming primitives, as well as how to solve problems using them, it remains for the learner to learn data wrangling-specific programming syntax. Depending on the size of the syllabus, memorizing all this syntax is often untenable and the learner is better served by training to look up syntax on demand.

3.2 Experimental structure

The structure of the experimental platform closely mirrors the underlying learning model, in how it is split into three parts, and outlined below:

Part 1 introduces students to a set of data wrangling operations and tests their ability to compose solutions using them via 9 non-programmatic drag-and-drop exercises.

Part 2 introduces general programming concepts via 10 practical exercises.

Part 3 is the main part of the study and asks of the student to solve 18 data wrangling exercises programmatically, three of which are "unscaffolded" (without subgoal labels or graphics for 10 minutes).

Theoretically these three knowledge domains - data wrangling, programming, and programmatic data wrangling - can be taught in a concurrent or interleaved fashion. However, we decided to split it into distinct parts in order to isolate the acquisition and measurement of data wrangling knowledge from that of programming competence. Part 1 effective serves as a breadth-first, advance organizer of data wrangling as a domain [1].

In designing the app, we reasoned that, overall, it would be more time-efficient to train students in how to retrieve syntax from documentation rather than explicitly walking them through the syntax for each operation. The web app is therefore structured around a sidebar menu that contains a *taxonomy* of 56 available data operations. In Part 2 and 3, this menu serves as a syntax reference, so Part 1 also serves to train students in navigating through it.

4 WALKTHROUGH OF APPARATUS

The web platform has already undergone several iterations, including a qualitative usability study (n=10) and an extensive pilot study (n=42) [46]. Furthermore, the graphical style of the subgoal graphics embedded in the app has been validated in a short survey (n=38) that showed that participants mostly understood their meaning.

Under the hood, the platform implements an RCT where 50% of all participants will receive subgoal graphics in Part 1 and 3 (a condition we call **SG**) while the rest will *only* receive textual subgoals, without graphics (¬**SG**). In Part 3, 15 exercises will have subgoals but three will be without subgoals for the first 10 minutes (we refer to these as "unscaffolded"), as a within-subjects factor. The participant can choose to do the tutorial in either Python or R - everything except syntax will be the same. In order to appeal to as many students as possible, it is technically possible to skip a part, but the app itself only explicitly recommends participants with previous data wrangling and programming experience to skip ahead to Part 3.

4.1 Part 1

To familiarize students with the operations, Part 1 is organized as a virtual deck of *operation cards*, as shown in Figure 3. Each card describes an operation, and the learner is instructed to click *Next* when they believe that they understood the operation. The sidebar taxonomy highlights the location of the current card's operation.

As they browse through the cards, occasionally one or two data wrangling exercises will appear. An example is shown in Figure 3. These exercises have already been broken down into steps, with instructor-provided subgoal labels,

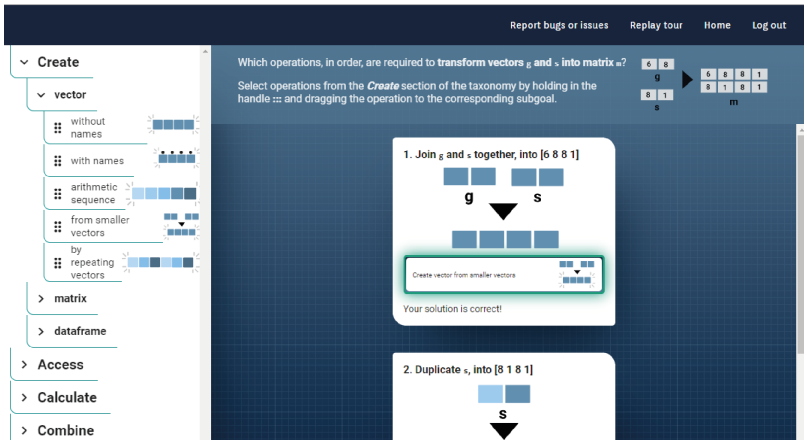


Fig. 3. Part 1 consists of drag-and-drop exercises where the participant selects the correct operation from a menu. The user gets immediate feedback and a hint if their answer is incorrect. In the control group, the graphics are absent. The very first exercise is shown.

as well as subgoal graphics for those in the SG condition. Using drag-and-drop, the students are instructed to, for each subgoal, select the corresponding operation from the sidebar taxonomy (out of the operations covered via operation cards thus far). Upon dropping an operation tile into a subgoal's drop zone, the user receives immediate feedback about its correctness. If correct, it will glow green with a success message. If incorrect, it will glow red and a hint will display. Only when all subgoals have been correctly matched with operations will the participant be able to proceed. There are 9 exercises in total.

4.2 Part 2

Part 2 features 10 programming exercises that are designed to teach complete novices programming fundamentals, including variables, data types, functions, basic indexing, logical conditions, and how to create vectors, matrices, and dataframes, as well as the previously described procedural skills. The solutions are mostly one-liners, and therefore do not feature subgoals or graphics. The taxonomy sidebar's operations are no longer draggable, but instead act as links for accessing corresponding documentation entries. The documentation has been created specifically for the experiment, and only exposes the most necessary details of each operator or function. Every exercise is loaded with the relevant documentation entry already visible in the sidebar.

The programming takes place inside an interactive programming widget. The widget[1] has one scripting area and one REPL area. As the learner types their solution into the scripting area, they have the option to run the program with or without without submission tests. The submission tests check for certain variable names and their contents, and output informative messages such as "Did you correctly specify variable *sums*?". Upon a correct submission, a success message appears and the participant is allowed to progress to the next exercise.

[1]The widget makes use of the DataCamp Light React API https://github.com/datacamp/datacamp-light

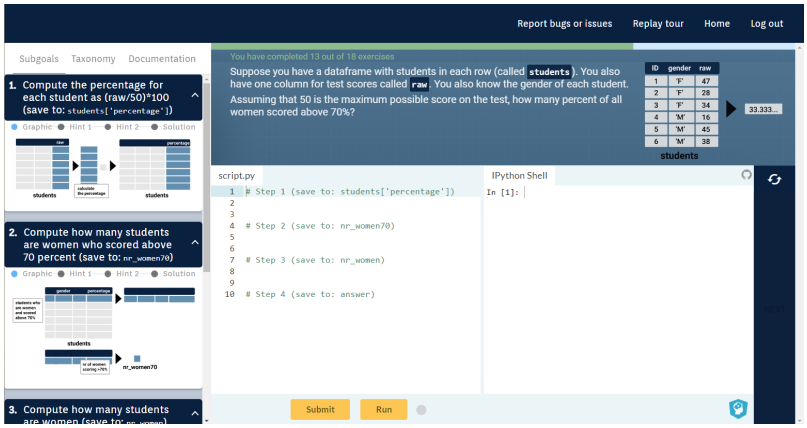


Fig. 4. Screenshot of a Part 3 exercise. The sidebar contains several tabs. By default, the pane is opened at the subgoal list, which for the SG condition holds graphics. The subgoal graphics, but not the textual labels themselves, are absent for ¬SG.

4.3 Part 3

Part 3 is the main part, and includes 18 data wrangling exercises, all of which require multiple (3-5) operations to be combined in a certain order. The exercises are of the toy dataset variety, since the focus is on imparting structural intuitions. In the sidebar pane a set of textual subgoal labels are visible, along with graphics in the SG condition (see Figure 4). The relevant documentation entry is no longer open by default - instead, the learner needs to deduce which operations are necessary, and proceed to look up the necessary syntax via the taxonomy. The submission tests work the same as before, with variable names suggested within the subgoal labels (e.g. "save to *sums*"). The learner is free to use another solution entirely, as long as the last variable contains the correct value.

4.3.1 Hints. Another feature of Part 3 is the availability of *hints*. Each subgoal is associated with three different hints. The first specifies *where* in the taxonomy to look for syntax, the second adds syntactic details, for example suggesting parameter values, and the final hint supplies the actual line of code corresponding to that subgoal. The hints are meant as a substitute for a human tutor, to prevent the user from getting completely stuck and dropping out due to frustration. The participants were instructed to use as few hints as possible.

4.3.2 Unscaffolded exercises. Three exercises (the same for all participants) spaced evenly within the exercise stack, are special in that they are designed to test learners' ability to cope without any subgoals. For the first 10 minutes, the sidebar will only contain the documentation, after which the subgoals (and subgoal graphics for SG) will appear. The participant is made aware of this. These exercises were interleaved among the scaffolded exercises, and the same for everyone. In either case, the learner is free to proceed to the next exercise as soon as the submission is correct. We chose to remove subgoals for 10 minutes only in order to minimize the extent to which students were tested as opposed to trained, as we anticipated this would lead unmotivated students to drop out.

4.4 Metrics

Pedagogical interventions can be beneficial in multiple ways. For a fuller picture, we will investigate subgoal graphics with respect to three variables: time on task in seconds, the number of incorrect attempts before solving an exercise,

and the number of completed Part 3 exercises. We argue that both time on task and number of attempts are necessary, since a participant could be fast by being haphazard, or slow by being cautious. A higher number of completed exercises could indicate that the condition is more motivating and enjoyable. Additionally, we gather demographic data from a survey administered at the beginning, and subjective feedback from an evaluation form given at the end of each part. Finally, to indicate how helpful they were, the SG group has the options to give a thumbs-up, thumbs-down or thumbs-sideways rating (i.e. neutral) for an exercise's subgoal graphics after completing the exercise.

4.5 Hypotheses

We hypothesize that subgoal graphics, compared with subgoal labels alone, are associated with:

$H_{1.1}$: shorter total time on task in Part 1
$H_{1.2}$: shorter total time on task in Part 3
$H_{2.1}$: fewer incorrect attempts in Part 1
$H_{2.2}$: fewer incorrect attempts in Part 3
$H_{3.1}$: better progress in unscaffolded exercises in Part 3
$H_{4.1}$: higher number of completed exercises in Part 3

These hypotheses were pre-registered at the Open Science Framework while data collection was still underway, and prior to any inspection of the data [44].

5 EXPERIMENTAL SETUP

5.1 Recruitment

Following ethics approval, the app was embedded into four courses. One was a scientific methods course open to students of any major, based at a UK research university and mostly composed of students without prior programming experience. Two courses, at a Pakistani research institution, were an introduction to quantitative reasoning module for non-majors with some procedural Python experience, and a sophomore CS class with more Python experience (but none in data wrangling). The fourth course was an introductory data science course aimed at teachers and based at a US university. We also advertised the tool at another introductory data science course at the UK university aimed at humanities students, as well as through a data science online community in Egypt, and a MOOC platform in Finland.

Although the app supports both Python and R, when advertised to a course, we asked participants to do it in Python, in order to remove language as a source of variance once we realized it was about the become the majority choice anyway.

5.2 Procedure

Delivery mode. In all instances, the study was taught completely online and asynchronously. Teachers in charge of the respective courses were given answer keys and, in the event of a participant getting completely stuck, were free to help them out over chat or teleconferencing software. Data relating to such external interactions were not collected.

Randomization protocol. Participants are automatically allocated to conditions based on the order in which they signed up, such that every second participant will receive subgoal graphics. This was to ensure balanced group sizes and that any recruited cohort is not oversampled in any group. Since the experiment was done remotely, we expect

participants to be blind to their assignment, though it is theoretically possible that participants show their sessions to each other and notice the lack of graphics.

Reward. To sign up, participants were asked to read through the consent form to give their informed consent, which was built into the app itself. Completion was non-credit bearing and participants were offered to have their data withdrawn from the research data set. Upon completing Part 3, every participant received an e-book containing a set of custom-made cheat sheets featuring all syntax covered by the taxonomy, which served as both a reward and to mitigate any differences in effect that could have resulted from the experimental manipulation. The students at the Pakistani university additionally received certificates as a token of gratitude and completion.

5.3 Inactivity data

We expected participants to be relatively distracted and intermittent in their app interactions, given its online and asynchronous nature. To control for this, we record the time spent in a different browser tab, logged out, or if a participant is inactive in terms of keystrokes and mouse movements for more than 60 seconds. For all time on task estimates that we use, such inactive time periods have already been subtracted as part of the data cleaning process.

6 ANALYSIS

6.1 Demographic characteristics

Out of 288 unique participants who engaged with at least one exercise in Part 1, 197 completed all Part 1 exercises (SG=100, ¬SG=97). 55% were male and 43% were female. In terms of degree choice, most studied CS (43%), but also common were biology (24%), chemistry (7%), engineering (5%), and economics (5%). The SG and ¬SG samples were had similar ratios in terms of gender, degree, and experience. Most students were from the UK science method course (34%), with 30% from the courses at the Pakistani institute, and 6% from the US data science course. 19% were unspecified guests.

It Part 3, the demographic composition is slightly different, since students could drop out prior to it, or optionally skip Part 1. Out of 204 participants who completed at least one Part 3 exercise, only 88 completed all 18 exercises. However, by excluding the last eight exercises, this number is increased to 134 (SG=66, ¬SG=68). This decision served to boost our sample size, at the cost of data loss with respect to exercises. This was a minor departure from our pre-registered protocol. Participants who completed at least 10 Part 3 exercises were 58% male, 39% female. 93% solved the exercises in Python. The most common majors were again CS (61%), biology (13%), chemistry (4%), and engineering (4%). CS-students, therefore, appear to be somewhat more likely to complete the tutorial. This might be because, already possessing a programming background, they are less likely to be overwhelmed by the content. However, we could partially rule out any attrition biases relating to experienced programmers being more likely to persist, since self-reported experience (in Python, R, and Excel) among those who started versus those who completed it were similar. Ratings for Excel were clustered around 3 (*Beginner's level*), ratings for Python were near uniformly distributed between 1 (*None at all*) and 4 (*Intermediate*), while ratings for R were mostly 1.

6.2 Part 1

We expect participants in the graphical condition to take less time to solve the exercises (H1.1), and submit fewer attempts before getting their solution right (H2.1). To put all participants on equal footing, only those who solved all Part 1 exercises were included in our analysis.

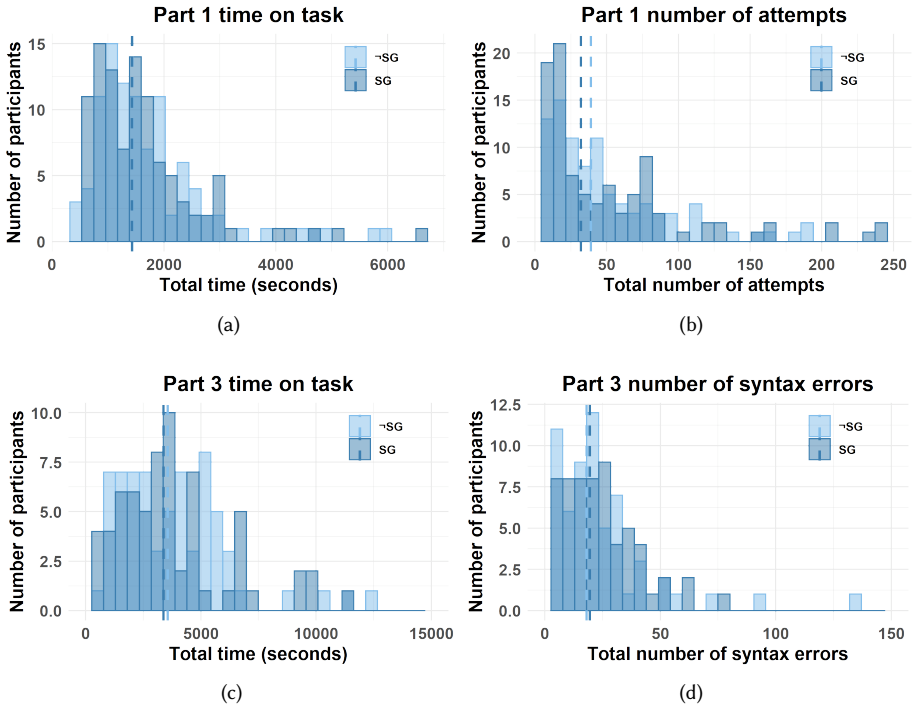


Fig. 5. Histograms for complete sets of performance data relating to Part 1 and 3. Dashed lines indicate medians.

6.2.1 Time on task. Times on tasks - recorded in seconds from the moment the exercise is presented up until a correct answer is submitted - are summed together across all exercises. The distributions do not indicate an effect (Figure 5a). The graphical condition exhibits a lower median time on task (N=100, 1432s, IQR=1007) compared with the control (N=97, 1438s, IQR=940), but only by 6 seconds. A two-tailed Wilcoxon rank sum test did not reveal a significant difference (W=5110, p=0.52). The null hypothesis for H1.1 cannot be rejected.

6.2.2 Number of attempts. All incorrect attempts are summed across the 9 exercises. The distributions, shown in Figure 5b, are positively skewed (likely a result of the lower, non-negative boundary) and do not indicate a discernible separation. No outliers were excluded. In line with our hypothesis, the SG group on average makes 7 fewer attempts (N=100, median=32,IQR=60.5) than -SG (N=97, median=39, IQR=55). A two-tailed Mann-Whitney U test did not reveal a significant difference (U=5286, p=0.28). The null hypothesis for H2.1 cannot be rejected.

6.3 Part 3

In Part 3, it was hypothesized that participants in the graphical condition persist for longer (H4.1), take shorter time (H1.2) and fewer attempts to complete the exercises (H2.2). Again, only complete observations were included for analysis (N=134).

6.3.1 Number of exercises completed. To explore whether participants in the experimental group persist for longer (H4.1), we plotted the distribution of completion data (Figure 6). The plot suggests that the control group (¬SG) drop

Fig. 6. Number of exercises completed (out of 18) for the two groups. Dashed lines indicate medians.

out in greater numbers just before the finishing lines, and that the SG group is over-represented among those who complete the set. In terms of median, SG complete 18 exercises (N=101, IQR=12) whereas ¬SG complete 16 (N=103, IQR=11.5). A two-tailed Mann-Whitney U test did not reveal this to be significant (U = 4768.5, p=0.28).

6.3.2 Time on task. Times on task were summed for all 12 scaffolded exercises, the results of which are plotted in Figure 5c. The descriptive data contradict the hypothesis, as the median duration is shorter for those in the graphical condition (N=66, median=3376s, IQR=2528s) compared with those without (N=68, median=3558s, IQR=3184s). This does not emerge as significant from a Mann-Whitney U test (U=2263,p=0.93). The null hypothesis of H1.2 cannot be rejected.

6.3.3 Number of attempts. When a participant clicks "Run" or "Submit" within the programming widget, it is registered as an attempt. The result is seen in Figure 5d. Participants in the graphical condition record more attempts (N=66, median=19.5,IQR=23.5) than the control (N=68, median=18, IQR=20.5). These are not significant in a Mann-Whitney U test (U=2103.5, p=0.53). The null hypothesis of H2.2 cannot be rejected.

6.3.4 Performance in exercises without subgoals. Recall that for 3 exercises out of the 18, neither SG nor ¬SG had any subgoals (or graphics) available to them for the first 10 minutes. This was to measure performance in "unscaffolded" circumstances. Since we are restricted to complete data, the sample size is 88 and no longer balanced (SG=52, ¬SG=36). In hypothesis H3.1, we predicted that the SG group would solve more subgoals within the first 10 minutes. We measure this by gathering all participants' code attempts in the first 10 minutes, and checking for syntax tokens and function names that indicate that the participant has identified the correct operation. This measure has a maximum of 10 points. We find these to have a skew towards the upper bound (Figure 7). The subgoal graphic group solved the same number of subgoals (N=52, median=8, IQR=5) than the control (N=36, median=8, IQR=2.25).

6.4 Subjective metrics

6.4.1 Evaluation form data. At the end of each part, participants are given an evaluation form with 5-point Likert-scale items that asks them how concentrated and motivated they felt during the part, and how effortful, enjoyable, and worthwhile they felt the task to be. A score of 1 indicated "Not at all" and a score of 5 indicated "Very much". The results of these surveys for both Part 1 and 3 are summarized in 9. No strong group differences are discernible, though in Part 3, it is worth noting that the ¬SG, a bigger proportion found the task *very effortful* and fewer found it *very worthwhile*.

The same surveys also asked participants about how helpful they found the subgoals (see Figure 8a) and subgoal graphics (see Figure 8b). Only SG participants were asked about the graphics. Emerging from the histograms is a heavy

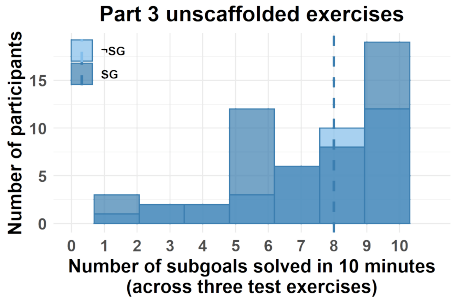
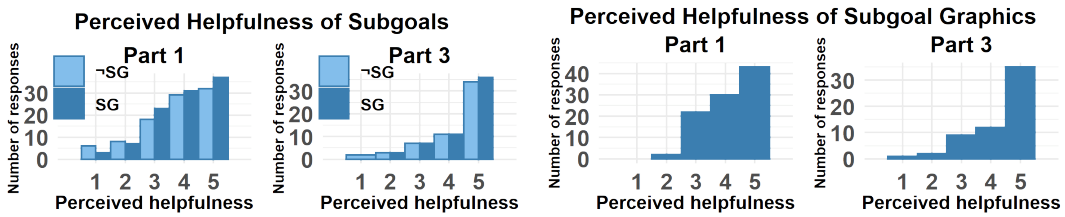


Fig. 7. Progress in unscaffolded exercises, measured by the number of subgoals solved in the first 10 minutes. The histograms are overlapping.



(a) (b)

Fig. 8. Evaluation data from Part 1 and 3, reflecting Likert-scale data (1="Not at all", 5="Very much"). Histograms are stacked, with SG as the darker color. Only the SG group were asked about subgoal graphics.

skew where most people (across both groups and parts) give it a rating of 5. The dominance of 5 is bigger in Part 3, where a majority found both subgoals and subgoal graphics *very helpful*.

6.4.2 Individual graphic ratings. Another source of evidence could be derived from the SG group's submission of subgoal graphic ratings. 17 participants submitted ratings for at least one exercise's subgoals. From exercises with at least one non-empty rating (N=985), 93% of all subgoal graphic ratings were positive and 2.6% negative.

7 DISCUSSION

Based on our literature review, we reasoned that subgoal graphics could improve the efficiency and efficacy with which novices learn problem decomposition in data wrangling tasks. Specifically, we predicted these benefits would manifest through shorter times on task and fewer incorrect attempts, in both non-programmatic and programmatic tasks, and better progress in unscaffolded programming exercises. The data did not show significant results for these hypotheses, and we are therefore unable to reject the possibility that subgoal graphics lack measurable performance benefits.

While disappointing, the lack of positive results should be interpreted in the context of the true discovery rate in educational research. A recent meta-review by Lortie-Forgues & Inglis of large-scale educational RCTs [29] suggests that their average effect size is around 0.06 *SD*s. Meanwhile Cheung & Slavin found that larger sample sizes and randomized designs tend to have smaller effect sizes [9]. While not exactly "large-scale", our study presented a reasonably sized, pre-registered RCT: this degree of rigor means that falsely positive results are less likely, and therefore that positive

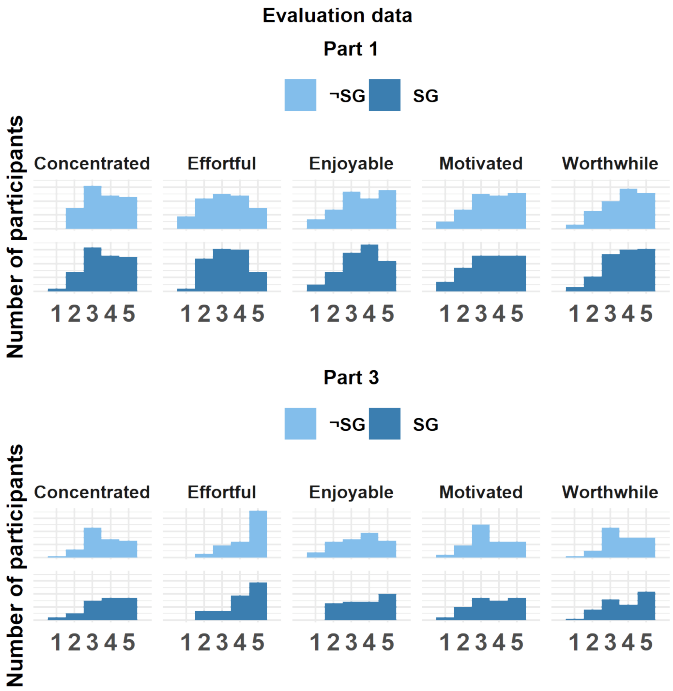


Fig. 9. Other evaluation data, asking participants on how enjoyable, effortful or worthwhile they found the task, as well as how concentrated and motivated they were (1="Not at all", 5="Very much")

effects in general are harder to obtain. Expectations should be adjusted accordingly, and not based on the positivity rate in the literature, which would over-estimate the prevalence of true educational effects [39].

If we interpret the results as true negatives, one possibility is that there are interactions at play, for example an expertise reversal effect [41], where the efficacy is moderated by the learner's prior expertise. Our participants included complete novices, students with some programming experience, and CS2 students, and any such interactions could have obscured a true effect. We did not deem the sample large enough to justify an interaction analysis, however we do believe it merits investigation in a more high-powered study.

One possibility is that the graphics themselves were unclear, such that subgoal graphics in principle could be helpful, but that our particular implementation of this idea was not. We do not find this plausible, given the graphic validation study we conducted previously, although that study recruited mostly participants with prior data science experience. However, we also obtained positive valuations of the graphics in this study, via the individual subgoal graphic ratings and the evaluation survey data reported here. We also do not think that lack of training, as Petre & Green [38] suggested would be necessary, is a problem, since Part 1 serves this training role.

What we instead offer as the most plausible explanation for the lack of effects is the distal nature of the dependent variables. Time on task and number of attempts depend on many factors, and compared with more theory-proximal variables such as "data wrangling plan comprehension" or "decomposition ability", the effect of graphics may be too diluted to measurably impact performance. It is also possible that the graphics were not paid attention to as much as

we presumed. Further analysis is needed to establish how often learners interacted with the graphics, and whether the graphics were ignored or not.

In the evaluation data, subgoal graphics were perceived to be *very helpful* by a plurality of the participants in Part 1, and a majority in Part 3. Demand characteristics may explain this partially, as a participant is unlikely to rate a pedagogical feature as directly unhelpful, but it is notable how *perceived* helpfulness contrasts with the lack of *measurable* helpfulness. This suggests a failure of meta-comprehension accuracy, where instructional features lead to an illusion of understanding [24, 53]. On the other hand, the perceived helpfulness could be intrinsically useful and serve a motivational role, for example helping people persist in their learning.

7.1 Threats to validity

The study has a number of methodological strengths, such as being randomized and controlled. Some aspects are both a strength and liability: the heterogeneous and multi-institutional sample was necessary for an adequate sample size and serves to boost its ecological validity, but this also inflated the variance in the data. Although we are primarily concerned with addressing the demand for data wrangling instruction among non-majors, it is the case that about half of our sample were CS majors, which attenuates the extent to which our findings can be generalized to our target audience. Since completion of the study was not directly credit-bearing, we also expect self-selection to play a part.

We also may have been overly cautious in not subjecting participants to too much testing, for fear of them dropping out. It is possible that if subgoals were completely removed from the unscaffolded exercises (rather than just for the first 10 minutes), there would have been a bigger effect.

There are also more subtle caveats to be aware of. For example, we operationalized performance through time on task and number of attempts before arriving at the correct solution. Given that participants were not penalized for taking time, using hints or making many attempts (through, for example, a reward system), they were free to pursue different strategies. Some may have used hints liberally or opted for a haphazard, trial-and-error-driven approach that is quick and with many attempts, others may have consciously tried to reduce their number of attempts, thus taking a longer time. This probably added variance to our data, and theoretically could be mitigated through gamification.

7.2 Future work

The null results leave open the possibility of interactions; in the future, we wish to recruit enough novices and prior programmers to conduct an interaction study. Additional measurements that could be introduced include the long-term retention of data skills and a more objective measure of prior experience, also encompassing their experience in databases and functional programming.

The methodology itself could serve as a useful template for future studies. By being both layered (in difficulty level) and self-contained (with minimal pre-requisites), the course could easily be slotted into a variety of data analytical courses around the world. We recommend that CS education researchers investigate whether their programming-related research question could be answered in a data wrangling context, or whether their findings transfer to them, since this opens up a far greater population of potential recruits and beneficiaries compared with CS-specific ones.

8 CONCLUSION

Previous research on subgoal labels and the inherent visualizability of data wrangling led us to explore the use of subgoal graphics in an e-learning experiment. We found that most participants perceive the subgoals and the subgoal graphics to be very helpful, which could serve to motivate students to complete the exercises. Indeed, among those

who completed all exercises, participants in the graphical condition were over-represented. However, we did not find any statistical differences between the times of task, attempt totals, or completion rates. It is possible that subgoal graphics do not carry further pedagogical benefits, but future research needs to ascertain whether this null result is due to interactions with prior expertise or the choice of dependent variables.

ACKNOWLEDGMENTS

Our data collection was possible with the help of Eric Yao, Junaid Akhtar, Briana Morrison, www.MOOC.fi, and Syed Waqar Nabi. The research is supported by an EPSRC doctoral grant (award ref 1946992).

REFERENCES

[1] David P Ausubel. 1978. In defense of advance organizers: A reply to the critics. *Review of Educational research* 48, 2 (1978), 251–257.
[2] Anna Bargagliotti, Wendy Binder, Lance Blakesley, Zaki Eusufzai, Ben Fitzpatrick, Maire Ford, Karen Huchting, Suzanne Larson, Natasha Miric, Robert Rovetti, et al. 2020. Undergraduate learning outcomes for achieving data acumen. *Journal of Statistics Education* 28, 2 (2020), 197–211.
[3] Austin Cory Bart, Javier Tibau, Eli Tilevich, Clifford A Shaffer, and Dennis Kafura. 2017. Blockpy: An open access data-science environment for introductory programmers. *Computer* 50, 5 (2017), 18–26.
[4] Austin Cory Bart, Ryan Whitcomb, Dennis Kafura, Clifford A Shaffer, and Eli Tilevich. 2017. Computing with corgis: Diverse, real-world datasets for introductory computing. *ACM Inroads* 8, 2 (2017), 66–72.
[5] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R Klemmer. 2010. Example-centric programming: integrating web search into the development environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 513–522.
[6] Joel Brandt, Philip J Guo, Joel Lewenstein, Mira Dontcheva, and Scott R Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1589–1598.
[7] Richard Catrambone. 1998. The subgoal learning model: Creating better examples so that students can solve novel problems. *Journal of experimental psychology: General* 127, 4 (1998), 355.
[8] Maurizio Cembalo, Alfredo De Santis, and Umberto Ferraro Petrillo. 2011. SAVI: a new system for advanced SQL visualization. In *Proceedings of the 2011 conference on Information technology education*. 165–170.
[9] Alan CK Cheung and Robert E Slavin. 2016. How methodological features affect effect sizes in education. *Educational Researcher* 45, 5 (2016), 283–292.
[10] Alyssa Counsell and Robert A Cribbie. 2020. Students' Attitudes toward Learning Statistics with R. *Psychology Teaching Review* 26, 2 (2020), 36–56.
[11] Jonathan Danaparamita and Wolfgang Gatterbauer. 2011. QueryViz: helping users understand SQL queries and their patterns. In *Proceedings of the 14th International Conference on Extending Database Technology*. 558–561.
[12] Andrew A David. 2021. Introducing Python Programming into Undergraduate Biology. *The American Biology Teacher* 83, 1 (2021), 33–41.
[13] H Davidson, K Peters, H Patton, F O'Hagan, and RA Cribbie. 2019. Statistical software in Canadian university psychology courses. *Teaching of Psychology* 20 (2019).
[14] Adrienne Decker, Lauren E Margulieux, and Briana B Morrison. 2019. Using the SOLO taxonomy to understand subgoal labels effect in CS1. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 209–217.
[15] Adrienne Decker, Briana B Morrison, and Lauren E Margulieux. 2019. Developing subgoal labels for imperative programming to improve student learning outcomes. In *ASEE Annual Conference proceedings*.
[16] Ian Drosos, Titus Barik, Philip J Guo, Robert DeLine, and Sumit Gulwani. 2020. Wrex: A Unified Programming-by-Example Interaction for Synthesizing Readable Code for Data Scientists. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–12.
[17] Kathi Fisler, Shriram Krishnamurthi, and Janet Siegmund. 2016. Modernizing plan-composition studies. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 211–216.
[18] Kristin Annabel Torjussen Folland. 2016. *viSQLizer: An interactive visualizer for learning SQL*. Master's thesis.
[19] Thomas RG Green and Marian Petre. 1992. When visual programs are harder to read than textual programs. In *Human-Computer Interaction: Tasks and Organisation, Proceedings of ECCE-6 (6th European Conference on Cognitive Ergonomics). GC van der Veer, MJ Tauber, S. Bagnarola and M. Antavolits. Rome, CUD*. Citeseer, 167–180.
[20] Sumit Gulwani, William R Harris, and Rishabh Singh. 2012. Spreadsheet data manipulation using examples. *Commun. ACM* 55, 8 (2012), 97–105.
[21] Mark Guzdial. 2007. *Introduction to media computation: a new CS1 approach aimed at non-majors and under-represented populations*. Technical Report. Georgia Institute of Technology.
[22] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362.
[23] Christopher D Hundhausen, Sarah A Douglas, and John T Stasko. 2002. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing* 13, 3 (2002), 259–290.

[24] Allison J Jaeger and Jennifer Wiley. 2014. Do illustrations help or harm metacomprehension accuracy? *Learning and Instruction* 34 (2014), 58–73.

[25] Byeongsu Kim, Taehun Kim, and Jonghoon Kim. 2013. Pen-and-Pencil Programming Strategy toward Computational Thinking for Non-Majors: Design Your Solution. *Journal of Educational Computing Research* 49, 4 (2013), 437–459.

[26] Cynthia Bailey Lee. 2013. Experience report: CS1 in MATLAB for non-majors, with media computation and peer instruction. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 35–40.

[27] Minjeong Lee. 2017. Exploring the Effect of SW Programming Curriculum and Content Development Model for Non-majors College Students: focusing on Visual Representation of SW Solutions. *Journal of Digital Contents Society* 18, 7 (2017), 1313–1321.

[28] Ruoxi Li. 2019. Teaching Undergraduates R in an Introductory Research Methods Course: A Step-by-Step Approach. *Journal of Political Science Education* (2019), 1–19.

[29] Hugues Lortie-Forgues and Matthew Inglis. 2019. Rigorous large-scale educational RCTs are often uninformative: Should we be concerned? *Educational Researcher* 48, 3 (2019), 158–166.

[30] Lauren E Margulieux, Richard Catrambone, and Mark Guzdial. 2016. Employing subgoals in computer programming education. *Computer Science Education* 26, 1 (2016), 44–67.

[31] Lauren E Margulieux, Mark Guzdial, and Richard Catrambone. 2012. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. In *Proceedings of the ninth annual international conference on International computing education research*. 71–78.

[32] Lauren E Margulieux, Briana B Morrison, and Adrienne Decker. 2019. Design and pilot testing of subgoal labeled worked examples for five core concepts in CS1. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 548–554.

[33] Lauren E Margulieux, Briana B Morrison, and Adrienne Decker. 2020. Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples. *International Journal of STEM Education* 7 (2020), 1–16.

[34] Wes McKinney et al. 2011. pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing* 14, 9 (2011), 1–9.

[35] Briana B Morrison, Lauren E Margulieux, and Mark Guzdial. 2015. Subgoals, context, and worked examples in learning computing problem solving. In *Proceedings of the eleventh annual international conference on international computing education research*. 21–29.

[36] Robert Muenchen. 2019. The Popularity of Data Science Software. https://www.r-bloggers.com/2019/04/data-science-software-used-in-journals-stat-packages-declining-including-r-ai-ml-software-growing/. [Online; accessed 14-March-2021].

[37] Andrew M Olney and Scott D Fleming. 2019. A Cognitive Load Perspective on the Design of Blocks Languages for Data Science. In *2019 IEEE Blocks and Beyond Workshop (B&B)*. IEEE, 95–97.

[38] Marian Petre and Thomas R. G. Green. 1993. Learning to read graphics: Some evidence that'seeing'an information display is an acquired skill. *Journal of Visual Languages & Computing* 4, 1 (1993), 55–70.

[39] Justus J Randolph and Roman Bednarik. 2008. Publication Bias in the Computer Science Education Research Literature. *J. Univers. Comput. Sci.* 14, 4 (2008), 575–589.

[40] Arjun Rao, Ayush Bihani, and Mydhili Nair. 2018. Milo: A visual programming environment for data science education. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, Vol. 2018-October. IEEE Computer Society, 211–215. https://doi.org/10.1109/VLHCC.2018.8506504

[41] Günter Daniel Rey and Florian Buchwald. 2011. The expertise reversal effect: cognitive load and motivational explanations. *Journal of Experimental Psychology: Applied* 17, 1 (2011), 33.

[42] RStudio. 2021. RStudio Cheat Sheets. https://github.com/rstudio/cheatsheets. [Online; accessed 21-Feb-2021].

[43] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)* 13, 4 (2013), 1–64.

[44] Lovisa Sundin, Sherif Aly, Nourhan Sakr, Juho Leinonen, and Quintin Cutts. 2021. Slice N Dice. https://doi.org/10.17605/OSF.IO/6Q8UT

[45] Lovisa Sundin and Quintin Cutts. 2019. Is it feasible to teach query programming in three different languages in a single session? A study on a pattern-oriented tutorial and cheat sheets. In *Proceedings of the 1st UK & Ireland Computing Education Research Conference*. 1–7.

[46] Lovisa Sundin and Quintin Cutts. 2021. Introducing Data Wrangling using Graphical Subgoals-Findings from an e-Learning Study. In *Proceedings of the Eighth ACM Conference on Learning@ Scale*. 267–270.

[47] John Sweller. 2011. Cognitive load theory. In *Psychology of learning and motivation*. Vol. 55. Elsevier, 37–76.

[48] tidyblocks.tech. 2021. TidyBlocks. https://github.com/tidyblocks/tidyblocks. [Online; accessed 21-Feb-2021].

[49] Endel Tulving. 1985. How many memory systems are there? *American psychologist* 40, 4 (1985), 385.

[50] Hadley Wickham. 2016. Data transformation. In *ggplot2*. Springer, 203–220.

[51] Hadley Wickham, R Francois, L Henry, and K Müller. 2014. Dplyr. In *useR! Conference*.

[52] Hadley Wickham and Maintainer Hadley Wickham. 2017. Package 'tidyr'. *Easily Tidy Data with'spread'and'gather ()'Functions* (2017).

[53] Jennifer Wiley, Allison J Jaeger, Andrew R Taylor, and Thomas D Griffin. 2018. When analogies harm: The effects of analogies on metacomprehension. *Learning and Instruction* 55 (2018), 113–123.

[54] Victoria Woodard and Hollylynne Lee. 2020. How Students Use Statistical Computing in Problem Solving. *Journal of Statistics Education* (2020), 1–18.